



Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

**Precificação em Computação em Nuvem para
Instâncias Permanentes e Transientes: Modelagem e
Previsão**

Gustavo Jardim Portella

Tese apresentada como requisito parcial para
conclusão do Doutorado em Informática

Orientador

Prof.a Dr.a Alba Cristina Magalhães Alves de Melo

Brasília
2021

Ficha Catalográfica de Teses e Dissertações

Esta página existe apenas para indicar onde a ficha catalográfica gerada para dissertações de mestrado e teses de doutorado defendidas na UnB. A Biblioteca Central é responsável pela ficha, mais informações nos sítios:

<http://www.bce.unb.br>

<http://www.bce.unb.br/elaboracao-de-fichas-catalograficas-de-teses-e-dissertacoes>

Esta página não deve ser incluída na versão final do texto.



Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

**Precificação em Computação em Nuvem para
Instâncias Permanentes e Transientes: Modelagem e
Previsão**

Gustavo Jardim Portella

Tese apresentada como requisito parcial para
conclusão do Doutorado em Informática

Prof.a Dr.a Alba Cristina Magalhães Alves de Melo (Orientador)
Dep. de Ciência da Computação/CIC-UnB

Prof.a Dr.a Lúcia Maria de A. Drummond Prof.a Dr.a Aletéia Patrícia F. de Araújo
Dep. de Ciência da Computação/IC-UFF Dep. de Ciência da Computação/CIC-UnB

Dr. Luiz DeRose
AWS/USA

Prof.a Dr.a Genáina Nunes Rodrigues
Coordenadora do Programa de Pós-graduação em Informática

Brasília, 4 de Março de 2021

Dedicatória

Dedico esse trabalho à minha esposa Luciana e aos meus filhos Lucas e Clara.

Agradecimentos

Agradeço à minha esposa Luciana e aos meus filhos Lucas e Clara, que sempre me apoiaram na realização desse sonho. Obrigado pelo amor, carinho e compreensão, que foram fundamentais durante toda a minha caminhada.

Agradeço à minha mãe (em memória) e ao meu pai por acreditarem na minha capacidade e, especialmente, pelo provimento de uma educação plena e de excelência. O meu agradecimento carinhoso aos meus familiares, aos familiares da minha esposa e aos meus amigos que, direta ou indiretamente, sempre me encorajaram e apoiaram.

À Professora Alba, agradeço pela confiança, amizade e, principalmente, por me guiar nessa incrível jornada até a fronteira do conhecimento. O meu sincero agradecimento aos docentes do Departamento de Ciência da Computação (CIC) da Universidade de Brasília (UnB), pela dedicação e competência nas atividades acadêmicas e científicas. O meu agradecimento especial à Professora Genáina Rodrigues, do CIC/UnB, e ao Professor Eduardo Nakano, do Departamento de Estatística (EST) da UnB, pelas incríveis contribuições à minha pesquisa. Agradeço ao Professor Maurício Bugarin, do Departamento de Economia da FACE/UnB, pelas contribuições na qualificação desta Tese. Agradeço também ao Professor Manish Parashar, por ter me recebido em uma visita científica de curta duração na Universidade de Rutgers, EUA.

Agradeço aos colegas e amigos da CAPES pelo valioso apoio. Ao Professor Sérgio Côrtes, agradeço pela amizade, sabedoria e, especialmente, pelo incentivo que me fez dar o passo inicial no doutorado. Agradeço aos amigos da Diretoria de Tecnologia da Informação (DTI) pelo apoio e encorajamento. O meu agradecimento também à equipe da Coordenação-Geral do Portal de Periódicos da CAPES, por viabilizar o acesso amplo e igualitário ao conteúdo científico de excelência, a nível nacional e internacional.

Registro também o meu agradecimento aos coordenadores dos projetos CNPq/AWS nº 440014/2020-4 e FAP-DF, chamada nº 01/2019 e concessão nº 23106.014035/2020-03, cujos recursos viabilizaram a execução de importantes experimentos citados nesta Tese.

Finalmente, agradeço a Deus pela oportunidade de viver, capacidade de pensar e disposição para aprender, e assim trilhar o meu caminho na pesquisa científica e contribuir para o avanço tecnológico da nossa sociedade.

Resumo

A computação em nuvem se consolidou como um modelo de computação distribuída em larga escala, onde provedores oferecem instâncias de máquinas virtuais utilizando diferentes modelos de precificação. Na precificação *on demand*, o preço cobrado do usuário é fixo por período de utilização. A precificação *spot* da AWS surgiu para o melhor aproveitamento da infraestrutura, sendo que os preços são variáveis e, em geral, menores do que os preços *on demand*, podendo haver revogação das instâncias. Assim, a modelagem da precificação *spot*, visando redução de custo e aumento de disponibilidade, consiste em um importante desafio, abordado de forma específica em diversos trabalhos da literatura. No entanto, uma abordagem que consiga avaliar variações de preço a curto prazo, bem como identificar tendências a longo prazo, permanece um problema em aberto. O principal objetivo desta Tese consiste na modelagem da precificação de instâncias permanentes e transitórias, de modo que os desafios de redução de custo e aumento de disponibilidade sejam superados. Para isso, inicialmente foi analisado o comportamento dos preços das instâncias *spot*, visando oferecer ao usuário uma estimativa de lance a curto prazo, alcançando 25% do preço *on demand*. O próximo passo consistiu na determinação uma função de utilidade que considera tanto o preço quanto a disponibilidade das instâncias *spot*, oferecendo lances a curto prazo que balanceiam esses dois objetivos conflitantes, atingindo 28,82% do preço *on demand* com disponibilidade prevista de 98,19%. Em seguida, foi projetado um arcabouço flexível com base em redes neurais LSTM (*Long Short Term Memory*), capaz de determinar tendências de preço *spot* a longo prazo, com baixo erro. O arcabouço LSTM e a função de utilidade foram combinados em um mecanismo, de forma a selecionar a instância *spot* de acordo com a tendência de preços e definir o valor lance mais apropriado. Finalmente, o mecanismo combinado foi utilizado por uma aplicação para execução de 218.179 comparações de sequências do SARS-CoV-2, fornecendo boas opções de instâncias *spot*, com baixo custo e alta disponibilidade. O custo médio observado foi de 40% do preço *on demand*, mantendo a instância por 8 horas, com menor custo por comparação e maior número de sequências comparadas do que outras abordagens.

Palavras-chave: computação em nuvem, modelos de precificação *on demand* e *spot*, análise estatística, função de utilidade, redes neurais LSTM

Abstract

Cloud computing has consolidated itself as a large-scale distributed computing model, where providers offer instances of virtual machines using different pricing models. In the on-demand pricing, the price charged to the user is fixed by period of use. AWS spot pricing emerged for a better use of the infrastructure, with prices being variable and, in general, lower than on demand prices, with the possibility of instance revocation. Thus, the modeling of spot pricing, aiming to reduce costs and increase availability, is an important challenge, addressed specifically in several works in the literature. However, an approach that can assess short-term price changes, as well as identify long-term trends, remains an open problem. The main objective of this PhD Thesis is to model the pricing of permanent and transient instances, so that the challenges of reducing costs and increasing availability are overcome. For this purpose, the price behavior of the spot instances was initially modeled, aiming to offer the user a short-term bid estimate, reaching 25% of the price on demand. The next step was to determine a utility function that considers both price and availability, offering short-term bids for the spot maximum price that balance these two conflicting objectives, reaching up to 28.82% of the on demand price with an expected availability of 98.19%. Then, a flexible framework was designed based on Long Short Term Memory (LSTM) neural networks, capable of determining long-term spot price trends, with low error. The LSTM framework and the utility function were combined in one mechanism, in order to select the spot instance according to the price trend and define the most appropriate spot maximum price value. Finally, the combined mechanism was used by an application that performs 218,179 SARS-CoV-2 sequence comparisons, providing good spot instances options, with low cost and high availability. The average cost observed was 40% of the on demand price, maintaining the instance for 8 hours, with lower cost per comparison and a greater number of sequences compared than other approaches.

Keywords: cloud computing, on demand and spot pricing models, statistical analysis, utility function, LSTM neural networks

Sumário

| | | |
|----------|--|-----------|
| 1 | Introdução | 1 |
| 1.1 | Motivação | 2 |
| 1.2 | Objetivos | 3 |
| 1.3 | Contribuições | 3 |
| 1.4 | Sumário da Tese | 5 |
| I | <i>Background/Contextualização</i> | 6 |
| 2 | Computação em Nuvem | 7 |
| 2.1 | Evolução Histórica | 8 |
| 2.2 | Características da Computação em Nuvem | 10 |
| 2.3 | Modelo de Serviços | 11 |
| 2.4 | Modelo de Implantação | 13 |
| 2.5 | Arquitetura de Computação em Nuvem | 13 |
| 2.5.1 | Modelo em Camadas | 14 |
| 2.5.2 | Utilização de Nuvem IaaS | 15 |
| 3 | Precificação em Computação em Nuvem | 17 |
| 3.1 | Tipos de Precificação | 17 |
| 3.1.1 | Precificação Estática | 18 |
| 3.1.2 | Precificação Dinâmica | 19 |
| 3.1.3 | Precificação Transiente | 19 |
| 3.2 | Amazon Web Services (AWS) | 20 |
| 3.2.1 | Amazon EC2 <i>on demand</i> | 20 |
| 3.2.2 | Amazon EC2 <i>reserved</i> | 21 |
| 3.2.3 | Amazon EC2 <i>spot</i> | 22 |
| 3.3 | Google Cloud Platform (GCP) | 25 |
| 3.3.1 | Google GCP <i>predefined</i> | 25 |
| 3.3.2 | Google GCP <i>custom</i> | 26 |

| | | |
|----------|--|-----------|
| 3.3.3 | Google GCP <i>reserved</i> | 26 |
| 3.3.4 | Google GCP <i>preemptible</i> | 27 |
| 3.4 | Microsoft Azure | 27 |
| 3.4.1 | Azure <i>pay-as-you-go</i> | 28 |
| 3.4.2 | Azure <i>reserved instances</i> | 28 |
| 3.5 | Comparação entre Provedores | 29 |
| 3.5.1 | Comparação dos Preços de Instâncias <i>on demand</i> | 29 |
| 3.5.2 | Comparação dos Modelos de Precificação | 30 |
| 4 | Séries Temporais, Teoria de Jogos e Leilões aplicados à Utilidade | 32 |
| 4.1 | Função de Utilidade | 33 |
| 4.2 | Séries Temporais | 35 |
| 4.3 | Média Móvel com Janela Deslizante | 38 |
| 4.4 | Teoria de Jogos | 40 |
| 4.5 | Leilões | 43 |
| 4.6 | Desenho de Mecanismos | 45 |
| 5 | Redes Neurais Artificiais | 46 |
| 5.1 | Redes Neurais e <i>Machine Learning</i> | 46 |
| 5.1.1 | <i>Perceptron</i> | 46 |
| 5.1.2 | Arquitetura da Rede Neural | 47 |
| 5.1.3 | Retropropagação dos Erros | 49 |
| 5.1.4 | Algoritmo Gradiente Descendente | 51 |
| 5.1.5 | Variações do Gradiente Descendente | 53 |
| 5.1.6 | Entendimento Geométrico do Gradiente Descendente | 55 |
| 5.1.7 | Inicialização e Parâmetros de Treinamento da Rede | 56 |
| 5.2 | Algoritmos de Otimização em Redes Neurais | 58 |
| 5.2.1 | Taxa de Aprendizado com Decaimento Dinâmico | 58 |
| 5.2.2 | Momento Clássico | 59 |
| 5.2.3 | Momento Nesterov | 60 |
| 5.2.4 | AdaGrad | 60 |
| 5.2.5 | RMSProp | 61 |
| 5.2.6 | ADAM | 62 |
| 5.2.7 | NADAM | 63 |
| 5.3 | Redes Neurais Recorrentes | 63 |
| 5.3.1 | Arquitetura RNN | 64 |
| 5.3.2 | Redes LSTM | 66 |

| | | |
|-----------|---|-----------|
| 6 | Precificação Dinâmica em Computação em Nuvem | 69 |
| 6.1 | Estado da Arte em Precificação | 69 |
| 6.1.1 | Lee e co-autores (2011) | 69 |
| 6.1.2 | Zhang e co-autores (2012) - CloudRecommender | 70 |
| 6.1.3 | Ben-Yehuda e co-autores (2013) | 70 |
| 6.1.4 | Javadi e co-autores (2013) | 71 |
| 6.1.5 | Tanaka e Murakami (2014) | 72 |
| 6.1.6 | Huang e co-autores (2015) - Cumulon | 73 |
| 6.1.7 | Karunakaran e Sundarraj (2015) | 73 |
| 6.1.8 | Singh e co-autores (2015) | 73 |
| 6.1.9 | Lucas-Simarro e co-autores (2015) | 74 |
| 6.1.10 | Ouyang e co-autores (2016) - SpotLight | 74 |
| 6.1.11 | Li e co-autores (2016) | 75 |
| 6.1.12 | Agarwal e co-autores (2017) | 76 |
| 6.1.13 | Wolski e co-autores (2017) - DrAFTS | 76 |
| 6.1.14 | Al-Theiabat e co-autores (2018) | 77 |
| 6.1.15 | Baughman e co-autores (2018) | 77 |
| 6.1.16 | Sharma e co-autores (2018) | 77 |
| 6.1.17 | Shastri e Irwin (2018) | 78 |
| 6.1.18 | Wang e co-autores (2018) | 78 |
| 6.1.19 | Wolski (2018), Baughman e co-autores (2019) | 79 |
| 6.1.20 | Liu e co-autores (2019) | 79 |
| 6.1.21 | Lu e co-autores (2019) | 80 |
| 6.1.22 | Mishra e co-autores (2019) | 80 |
| 6.1.23 | Baig e co-autores (2020) | 80 |
| 6.1.24 | Khandelwal e co-autores (2020) | 81 |
| 6.2 | Análise Comparativa | 81 |
| | | |
| II | Contribuições | 85 |
| | | |
| 7 | Modelagem Estatística de Precificação <i>On demand</i> e <i>Spot</i> | 86 |
| 7.1 | Modelagem da Precificação <i>On demand</i> | 86 |
| 7.1.1 | Modelagem Estatística Aplicada | 87 |
| 7.1.2 | Dados Utilizados e Experimentos Realizados | 87 |
| 7.2 | Modelagem da Precificação <i>Spot</i> | 94 |
| 7.2.1 | Projeto de Modelagem Estatística | 94 |
| 7.2.2 | Análises Realizadas | 96 |

| | | |
|------------|---|------------|
| 7.3 | Considerações sobre as Modelagens Estatísticas | 102 |
| 8 | Mecanismo de Utilidade para Instâncias <i>Spot</i> da Amazon EC2 | 104 |
| 8.1 | Projeto do Mecanismo | 104 |
| 8.1.1 | Função de Utilidade | 105 |
| 8.1.2 | Algoritmo de Utilidade | 105 |
| 8.1.3 | Extensão do Algoritmo de Utilidade | 107 |
| 8.2 | Resultados dos Experimentos | 108 |
| 8.2.1 | Dados Utilizados | 108 |
| 8.2.2 | Implementação e Disponibilidade de Código | 108 |
| 8.2.3 | Experimentos com Mecanismo de Utilidade | 109 |
| 8.2.4 | Experimentos com Mecanismo Estendido para Dia da Semana . . . | 112 |
| 8.3 | Considerações sobre o Mecanismo de Utilidade | 114 |
| 9 | Mecanismo Combinado para Análise de Tendência e Previsão de Preços <i>Spot</i> da Amazon EC2 | 115 |
| 9.1 | Projeto do Mecanismo LSTM | 116 |
| 9.1.1 | Arquiteturas da Rede LSTM | 117 |
| 9.1.2 | Hiperparâmetros da Rede LSTM | 118 |
| 9.2 | Extensão do Mecanismo de Utilidade para Cálculo da Disponibilidade Real | 119 |
| 9.3 | Experimentos Realizados | 120 |
| 9.3.1 | Experimentos com Mecanismo LSTM | 121 |
| 9.3.2 | Experimentos com Mecanismo de Utilidade (Preços de 2020) | 127 |
| 9.4 | Metodologia para Uso Conjunto dos Mecanismos LSTM e Utilidade | 130 |
| 9.5 | Estudo de Caso: Comparação de Sequências Biológicas SARS-CoV-2 | 131 |
| 9.6 | Considerações sobre o Mecanismo Combinado | 135 |
| III | Conclusão | 138 |
| 10 | Conclusão e Trabalhos Futuros | 139 |
| 10.1 | Conclusão | 139 |
| 10.2 | Trabalhos Futuros | 141 |
| | Referências | 143 |
| | Anexo | 153 |
| I | Artigo Publicado no WSCAD 2016 (primeira página) | 154 |

| | | |
|-----|---|-----|
| II | Artigo Publicado no CCPE 2017 (primeira página) | 156 |
| III | Artigo Publicado no IEEE Cloud 2019 (primeira página) | 158 |
| IV | Artigo Submetido a Periódico (primeira página) | 160 |

Lista de Figuras

| | | |
|-----|--|----|
| 2.1 | Tecnologias que influenciaram o surgimento e a evolução da computação em nuvem nas últimas décadas, adaptado de [1]. | 9 |
| 2.2 | Arquitetura de Computação em Nuvem, adaptado de [2]. | 14 |
| 2.3 | Arquitetura de Computação em Nuvem no modelo de serviço <i>IaaS</i> , adaptado de [1]. | 16 |
| 3.1 | CrITÉrios para classificação de precificação em nuvem. | 18 |
| 3.2 | Evolução da precificação do serviço Amazon EC2, adaptado de [3]. | 24 |
| 4.1 | Exemplo da função de utilidade no jogo justo [4]. | 35 |
| 4.2 | SÉrie temporal com a variação de preços de instâncias <i>spot</i> da Amazon EC2 durante o primeiro semestre de 2020 na região US-East. | 37 |
| 4.3 | SÉrie temporal do tipo ruído branco ou <i>white noise</i> [5]. | 38 |
| 4.4 | MÉdia móvel com janela deslizante para uma sÉrie temporal discreta [6]. | 39 |
| 4.5 | Representação normal ou estratégica do dilema dos prisioneiros através de uma matriz de decisões. | 42 |
| 4.6 | Mecanismo VCG genérico, adaptado de [7]. | 45 |
| 5.1 | <i>Perceptron</i> ou unidade de processamento individual da rede neural [8]. | 47 |
| 5.2 | Arquitetura de rede neural em camadas ou <i>multilayer perceptron</i> [9]. | 48 |
| 5.3 | Função de erro e seus mínimos locais e global. | 49 |
| 5.4 | Ilustração da cadeia do cálculo diferencial em um grafo simples [9]. | 50 |
| 5.5 | Convergência do cálculo do gradiente descendente [11]. | 55 |
| 5.6 | Representação da arquitetura da rede RNN com conexões diretas e de retorno, adaptada de [12] e [13]. | 64 |
| 5.7 | Arquitetura da rede RNN desdobrada no tempo [14]. | 65 |
| 5.8 | Nó LSTM utilizado nas camadas ocultas da rede neural [15]. | 67 |
| 7.1 | Resumo do experimento <i>mlr</i> com dados da Amazon EC2 coletados em [16]. | 89 |
| 7.2 | Resumo do experimento <i>mlr</i> com dados do Google GCP coletados em [16]. | 90 |

| | | |
|------|---|-----|
| 7.3 | Dendograma de representação dos grupos em função da distância entre instâncias, após aplicação do algoritmo de clusterização <i>Minimal Spanning Tree</i> | 91 |
| 7.4 | Resumo do experimento <i>mlr-spt</i> com dados da Amazon EC2 coletados em [16], após aplicação do algoritmo de clusterização <i>Minimal Spanning Tree</i> | 91 |
| 7.5 | Resumo do experimento <i>mlr-ecu-restricted</i> com dados da Amazon EC2 de [17] com uso da métrica ECU. | 92 |
| 7.6 | Resumo do experimento <i>mlr-estricted-no-core</i> com dados da Amazon EC2 de [17], desconsiderando o fator <i>Virtual Cores</i> | 93 |
| 7.7 | Preços observados e estimados para a instância c3.8xlarge durante os meses de setembro, outubro e novembro de 2016. | 97 |
| 7.8 | Preços observados e estimados para a instância g2.2xlarge durante os meses de setembro, outubro e novembro de 2016. | 98 |
| 7.9 | Preços observados e estimados para a instância m4.10xlarge durante os meses de setembro, outubro e novembro de 2016. | 99 |
| 7.10 | Preços observados e estimados para a instância r3.2xlarge durante os meses de setembro, outubro e novembro de 2016. | 100 |
| 7.11 | Preços observados e estimados para a instância t1.micro durante os meses de setembro, outubro e novembro de 2016. | 101 |
| 8.1 | Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância c3.8xlarge ($\theta = 2$). | 110 |
| 8.2 | Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância t1.micro ($\theta = 1$). | 110 |
| 8.3 | Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância r3.2xlarge ($\theta = 3$). | 111 |
| 8.4 | Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância g2.2xlarge ($\theta = 2$). | 111 |
| 8.5 | Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância m4.10xlarge ($\theta = 2$). | 112 |
| 9.1 | <i>Layout</i> arquitetural básico (tipo 1). | 117 |
| 9.2 | <i>Layout</i> arquitetural intermediário (tipo 2). | 118 |
| 9.3 | <i>Layout</i> arquitetural sofisticado (tipo 3). | 118 |
| 9.4 | Análise futura de lance e disponibilidade. | 120 |
| 9.5 | Previsão LSTM para a instância r4.2xlarge em Janeiro de 2020. | 124 |
| 9.6 | Previsão LSTM para a instância r5.2xlarge em Fevereiro de 2020. | 124 |
| 9.7 | Previsão LSTM para a instância m5.4xlarge em Março de 2020. | 125 |

| | | |
|------|--|-----|
| 9.8 | Previsão LSTM para a instância c5n.2xlarge em Abril de 2020. | 125 |
| 9.9 | Previsão LSTM para a instância i3.8xlarge em Maio de 2020. | 126 |
| 9.10 | Previsão LSTM para a instância r5.2xlarge em Maio de 2020. | 126 |
| 9.11 | Lance e disponibilidade da instância c5n.9xlarge no 1º semestre de 2020. . . | 128 |
| 9.12 | Lance e disponibilidade da instância c5n.9xlarge no 2º semestre de 2020. . . | 129 |
| 9.13 | Lance e disponibilidade da instância r4.2xlarge no 1º semestre de 2020. . . | 129 |
| 9.14 | Lance e disponibilidade da instância r4.2xlarge no 2º semestre de 2020. . . | 130 |
| 9.15 | Tendência de preços do mecanismo LSTM para a instância m5.2xlarge. . . | 132 |
| 9.16 | Tendência de preços do mecanismo LSTM para a instância c5n.9xlarge. . . | 134 |
| 9.17 | Tendência de preços do mecanismo LSTM para a instância i3.2xlarge. . . | 134 |
| 9.18 | Tendência de preços do mecanismo LSTM para a instância r5.2xlarge. . . | 134 |
| 9.19 | Exemplo de alinhamento gerado pelo estudo de caso. | 135 |

Lista de Tabelas

| | | |
|-----|---|-----|
| 3.1 | Tipos de instâncias <i>on demand</i> disponíveis no serviço Amazon EC2 [17]. | 21 |
| 3.2 | Atendimento das requisições dependendo do preço do lance e da disponibilidade do tipo de instância. | 23 |
| 3.3 | Configurações pré-definidas do GCP sob demanda [18]. | 25 |
| 3.4 | Itens de personalização de instâncias do provedor GCP [18]. | 26 |
| 3.5 | Itens de personalização e configurações pré-definidas de instâncias do GCP no modelo de reserva antecipada [18]. | 26 |
| 3.6 | Instâncias Microsoft Azure no modelo pay-as-you-go [19]. | 28 |
| 3.7 | Comparação dos preços praticados pelos provedores AWS, GCP e Microsoft Azure no modelo de precificação <i>on demand</i> em 2017. | 29 |
| 3.8 | Comparação dos modelos de precificação dos provedores AWS, GCP e Microsoft Azure. | 30 |
| 5.1 | Hiperparâmetros mais comuns em redes neurais. | 57 |
| 6.1 | Comparação entre os trabalhos sobre precificação em computação em nuvem. | 84 |
| 7.1 | Experimentos com as instâncias <i>on demand</i> dos provedores AWS e GCP. | 88 |
| 7.2 | Grupos de Instâncias Amazon EC2 após aplicação do algoritmo de clusterização <i>Minimal Spanning Tree</i> | 90 |
| 7.3 | Aplicação da equação de regressão multilinear do experimento <i>mlr-ecu-restricted</i> em tipos de instância Amazon EC2 <i>memory optimized</i> | 94 |
| 7.4 | Tipos de instâncias Amazon EC2 analisadas. | 95 |
| 7.5 | Análise de custo e disponibilidade da instância c3.8xlarge. | 97 |
| 7.6 | Análise de custo e disponibilidade da instância g2.2xlarge. | 98 |
| 7.7 | Análise de custo e disponibilidade da instância m4.10xlarge. | 99 |
| 7.8 | Análise de custo e disponibilidade da instância r3.2xlarge. | 101 |
| 7.9 | Análise de custo e disponibilidade da instância t1.micro. | 102 |
| 8.1 | Tipos de instância, preços <i>on demand</i> e número de variações de preço <i>spot</i> de setembro a novembro de 2016. | 109 |

| | | |
|-----|--|-----|
| 8.2 | Percentuais de sugestões de dias da semana para t1.micro. | 113 |
| 8.3 | Percentuais de sugestões de dias da semana para c3.8xlarge. | 113 |
| 8.4 | Percentuais de sugestões de dias da semana para r3.2xlarge. | 114 |
| 9.1 | Valores possíveis dos hiperparâmetros de configuração. | 119 |
| 9.2 | Tipos de instância utilizados nos experimentos com o mecanismo combinado. | 121 |
| 9.3 | Análise de MSE e MSE-limite por configuração e tipo de rede LSTM. | 122 |
| 9.4 | Melhores resultados de MSE no período de teste por tipo de instância. | 123 |
| 9.5 | Piores resultados de MSE no período de teste abaixo do limite pré-estabelecido ($MSE \leq 0,0001$). | 123 |
| 9.6 | Resultados experimentais do mecanismo de utilidade durante o primeiro trimestre (Q1) de 2020. | 127 |
| 9.7 | Resultados experimentais do mecanismo de utilidade durante o segundo se- mestre (Q2) de 2020. | 128 |
| 9.8 | Resultados detalhados do estudo de caso. | 133 |
| 9.9 | Comparação entre os trabalhos sobre precificação em computação em nuvem. | 137 |

Capítulo 1

Introdução

A computação em nuvem surgiu no início dos anos 2000 como um modelo para utilização de recursos computacionais em larga escala. É uma tecnologia de computação distribuída orientada por aspectos econômicos, caracterizada pela utilização massiva de virtualização e com a finalidade de prover escalabilidade de recursos [20]. Por meio dela, a computação flexível e sob demanda é possível através do acesso a um conjunto de recursos computacionais configuráveis, como por exemplo redes, servidores, armazenamento, aplicativos e serviços [21].

Os provedores de computação em nuvem pública, como Amazon Web Services (AWS) [22], Google Cloud Platform (GCP) [23] e Microsoft Azure [19] oferecem seus serviços de provisionamento de infraestrutura computacional, utilizando diferentes modelos de negócio. A contratação sob demanda (*on demand*) de instâncias de máquinas virtuais foi o primeiro modelo a ser oferecido, no qual o provedor estabelece o custo a ser cobrado do usuário a partir de um preço fixo por unidade de tempo, a depender das características computacionais da instância provisionada. Esse é o modelo considerado tradicional, sendo que a sua desvantagem se evidencia no custo elevado para o usuário e na pouca previsibilidade quanto à alocação dos recursos do ponto de vista do provedor, o que possibilita períodos com grande quantidade de recursos ociosos.

Com a evolução dos mecanismos de precificação utilizados na computação em nuvem, surgiram alternativas que combinam baixo custo de contratação e diferentes garantias de disponibilidade dos recursos alocados. É o caso da precificação dinâmica *spot*, disponível no serviço Elastic Compute Cloud (EC2) [24] da AWS, que faz uso de um modelo de mercado para provisionamento de instâncias transientes. Nesse mecanismo de precificação, o custo pode cair significativamente em relação à precificação tradicional, mas sem a garantia de disponibilidade da instância. Até 2017, o provedor AWS fez uso de um modelo de mercado real, com intensa variação de preços das instâncias. Ao final do referido ano, a AWS passou a adotar um modelo de mercado suavizado, sem a obrigatoriedade de

estabelecimento de um preço máximo a ser pago pelo usuário [25].

Diante do exposto, a contratação de infraestrutura computacional por meio da computação em nuvem se tornou uma tarefa complexa para o usuário, na qual devem ser considerados não somente os aspectos relativos ao custo e disponibilidade do recurso, mas também seu correto dimensionamento de acordo com as necessidades reais da aplicação.

Este capítulo apresenta a introdução da Tese, que analisa os modelos de precificação de recursos em computação em nuvem pública e apresenta estratégias para a melhor tomada de decisão por parte do usuário. Na Seção 1.1 é detalhada a motivação desta pesquisa. Os objetivos principais e específicos são descritos na Seção 1.2. As contribuições são elencadas na Seção 1.3. Ao final, na Seção 1.4, é descrita a estrutura dos demais capítulos desta Tese.

1.1 Motivação

O modelo de precificação tradicional *on demand*, utilizado por diversos provedores de nuvem pública para provimento de infraestrutura como serviço [21], tem sido estudado em trabalhos como [26], [27] e [28]. Foi observado nestes trabalhos que as características de infraestrutura computacional (por exemplo processador e memória) influenciam na composição dos preços dos recursos oferecidos pelos provedores.

Os modelos de precificação dinâmica para instâncias transientes surgiram para oferecer uma alternativa mais barata aos usuários, assim como permitir que os provedores aproveitem melhor sua infraestrutura. A partir de 2013, foram publicados diversos estudos que utilizam técnicas de análise estatística e série temporal aplicadas em modelos de precificação dinâmica, como [29], [30], [28], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40] e [41]. Abordagens econômicas para modelagem desse tipo de mercado também têm sido empregadas, como nos trabalhos [42], [43] e [38]. Os objetivos desses estudos compreendem seleção de recursos, escalonamento de tarefas, redução de custos e aumento de disponibilidade a partir dos modelos de precificação dinâmica em computação em nuvem.

Em 2017, a utilização de redes neurais artificiais como técnica de previsão de preços dinâmicos em computação em nuvem se tornou predominante, como pode ser observado em trabalhos como [44], [25], [45] e [46]. O emprego desta técnica acompanhou a mudança do modelo de precificação *spot* da AWS, do mercado real para o suavizado, que ocorreu no final do referido ano.

Desse modo, a utilização de instâncias de máquinas virtuais com custos reduzidos e garantias de disponibilidade variáveis se apresenta como um desafio para os usuários de computação em nuvem. Entretanto, até o presente momento não foi encontrada pesquisa sobre estratégias de precificação em computação em nuvem, que considere custo e dispo-

nibilidade de forma balanceada a curto prazo, assim como preveja tendências de preços a longo prazo, de forma a proporcionar a tomada de decisão aprimorada por parte do usuário de computação em nuvem.

1.2 Objetivos

O objetivo geral da presente Tese de Doutorado consiste na proposição e investigação de estratégias de precificação estática e dinâmica para computação em nuvem, tanto para alocação de instâncias permanentes quanto transientes, que sejam válidas para os mercados real e suavizado, de modo que os desafios para os usuários de computação em nuvem sejam superados. Tal investigação aborda a variação de preços dinâmica, considerando redução de custos e aumento de disponibilidade das instâncias, proporcionando a execução satisfatória da aplicação do usuário e o melhor aproveitamento dos recursos computacionais provisionados.

Os objetivos específicos desta Tese compreendem:

- A análise e o entendimento aprofundado do modelo de precificação dinâmico *spot* do serviço Amazon EC2, proporcionando melhores condições de custo, disponibilidade e momento adequado para contratação dos recursos, sob o ponto de vista do usuário de computação em nuvem;
- A definição de estratégias para previsão de custo e disponibilidade a curto prazo de instâncias *spot* do serviço Amazon EC2, assim como para a identificação de tendências de preços a longo prazo, de forma a apoiar a tomada de decisão do usuário;
- A realização de estudo de caso para analisar as estratégias de identificação de tendências de preços e sugestão de lance de acordo com a estimativa de disponibilidade esperada de instâncias *spot* do serviço Amazon EC2.

A compreensão de como as características computacionais influenciam na composição dos preços das instâncias, considerando o modelo tradicional de precificação *on demand* utilizado pelos provedores de nuvem pública, é um objetivo específico secundário desta Tese.

1.3 Contribuições

Para o alcance do objetivo geral e dos objetivos específicos descritos na Seção 1.2, são elencadas à seguir as principais contribuições desta Tese:

- Proposta de um mecanismo baseado em utilidade com o objetivo de equilibrar a diminuição do custo de utilização do recurso e o aumento da sua disponibilidade, proporcionando um balanceamento desses dois objetivos conflitantes do ponto de vista do usuário de computação em nuvem;
- Proposta de um mecanismo que combina análise de utilidade a curto prazo, de forma a oferecer sugestão de lances para o usuário de computação em nuvem, e aprendizado de máquina para previsão de preço e identificação de tendências *spot* a longo prazo, utilizando rede neural recorrente *Long Short Term Memory* (LSTM) adaptada para previsão em séries temporais;
- Realização de estudo de caso composto pela execução de uma aplicação real de Bioinformática, de forma a comprovar a escolha satisfatória de instâncias com base na análise de tendências de preços, assim como da adoção apropriada de sugestões de lance de acordo com a disponibilidade esperada, considerando utilidade do ponto de vista do usuário.

As três contribuições listadas acima foram testadas no modelo de mercado *spot* mais atual da AWS (mercado suavizado), com dados de 2020 e 2021, obtendo resultados muito significativos.

Como contribuições secundárias desta Tese, temos:

- Modelagem estatística da precificação *on demand* utilizada pelos provedores AWS e GCP, com vistas a identificação de parâmetros que compõem os preços praticados. Com isso, os usuários podem compreender a formação de preços dos provedores e elaborar estratégias para contratação de recursos mais adequados às suas necessidades;
- Modelagem estatística com aplicação de técnicas para previsão de disponibilidade e preços no modelo *spot* do serviço Amazon EC2. Desse modo, o usuário tem a possibilidade de reduzir custos e aumentar a disponibilidade das instâncias contratadas neste modelo de precificação. Para aprimorar ainda mais os resultados obtidos, é proposta a extensão das análises estatísticas do modelo de precificação dinâmica com uso de séries temporais.

A segunda contribuição secundária desta Tese foi testada no modelo de mercado real para instâncias *spot*, com dados do ano de 2016, obtendo resultados significativos para esse modelo.

1.4 Sumário da Tese

A Tese está dividida em três partes. A primeira parte (*Background/Contextualização*) apresenta a fundamentação teórica e os conceitos abordados ao longo desta Tese. O Capítulo 2 detalha a computação em nuvem, seus principais conceitos e características. Os mecanismos de precificação de instâncias utilizados pelos principais provedores de computação em nuvem pública são detalhados no Capítulo 3. O Capítulo 4 aborda a fundamentação teórica sobre utilidade do ponto de vista do usuário de computação em nuvem, assim como técnicas estatísticas e modelagem de séries temporais utilizadas. Os principais aspectos sobre redes neurais artificiais e sua aplicação como técnica de *machine learning* são abordados no Capítulo 5. No Capítulo 6 é apresentada a pesquisa relacionada ao estado da arte sobre precificação de recursos computacionais aplicada a computação em nuvem, assim como uma análise comparativa dos trabalhos analisados.

A segunda parte da Tese (Contribuições) descreve de maneira cronológica as contribuições resultantes da pesquisa. Essa parte é composta por três capítulos. No Capítulo 7 são apresentadas as duas contribuições secundárias da Tese: a análise estatística realizada sobre o modelo de precificação *on demand* e a modelagem estatística da precificação *spot* do serviço Amazon EC2. A primeira contribuição principal (primária) da Tese, que consiste da proposta do mecanismo baseado em utilidade do ponto de vista do usuário de computação em nuvem é apresentada no Capítulo 8. O mecanismo que combina análise de utilidade a curto prazo, assim como a identificação de tendências de preços *spot* a longo prazo usando redes LSTM, é a segunda contribuição primária da Tese, proposto no Capítulo 9. Tanto a análise de utilidade como a identificação de tendências de preços *spot* são avaliados em separado com históricos de preços recentes, do ano 2020. Finalmente, os dois módulos são avaliados de maneira combinada em um estudo de caso real que compara milhares de sequências do vírus SARS-CoV-2 e foi executado na Amazon EC2 em janeiro de 2021.

Finalmente, as conclusões e as possibilidades de trabalhos futuros são discutidas na última parte da Tese. Além do Capítulo 10, essa última parte contém as referências bibliográficas e quatro anexos relativos à produção científica derivada da Tese. O Anexo I apresenta a primeira página do artigo [47] publicado no XVII Simpósio de Sistemas Computacionais de Alto Desempenho (WSCAD) em 2016. O Anexo II apresenta a primeira página do trabalho [48] publicado em 2019 no periódico internacional *Concurrency and Computation: Practice and Experience*. O Anexo III apresenta a primeira página do artigo [49] publicado na conferência internacional *12th IEEE International Conference on Cloud Computing*. A primeira página do artigo [50], submetido a periódico internacional em outubro de 2020, é apresentada no Anexo IV, sendo que o manuscrito revisado foi submetido em fevereiro de 2021.

Parte I

Background/Contextualização

Capítulo 2

Computação em Nuvem

A computação em nuvem (ou *cloud computing*) se caracteriza como um modelo de computação distribuída para utilização de recursos computacionais em larga escala. Este novo modelo possui características que o diferenciam consideravelmente das abordagens computacionais tradicionais, tais como o uso extensivo de soluções de virtualização, a possibilidade de fornecimento de novos recursos através de provisionamento dinâmico e a utilização de um modelo de cobrança em função dos tipos de recursos disponibilizados e do tempo de utilização [20].

A definição do *National Institute of Standards and Technology* (NIST) é uma das mais completas e estabelece que a computação em nuvem é [21]:

“Um modelo para permitir acesso ubíquo, conveniente e sob demanda a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo redes, servidores, armazenamento, aplicativos e serviços), que podem ser rapidamente provisionados e disponibilizados com o mínimo de esforço de gerenciamento ou de interação com o serviço.”

Luis Vaquero e co-autores, definem computação em nuvem da seguinte maneira [51]:

“A Nuvem é um grande reservatório de recursos virtualizados, facilmente utilizáveis e acessíveis. Esses recursos podem ser dinamicamente reconfigurados para ajustar à carga (escala) variável do sistema, permitindo o uso otimizado dos mesmos. Esse repositório de recursos é geralmente explorado por um modelo pay-per-use no qual as garantias são oferecidas por meio de SLAs (ou Service Level Agreements).”

Para Buyya e co-autores, o conceito de computação em nuvem abrange a seguinte definição [52]:

“Um tipo de sistema paralelo e distribuído que consiste de uma coleção de computadores interligados e virtualizados, que são provisionados dinamicamente e compreendidos como um ou mais recursos computacionais unificados com base em acordos de nível de serviço estabelecidos através da negociação entre o provedor de serviços e os consumidores.”

Por último, para Ian Foster e co-autores, a definição de computação em nuvem é a seguinte [20]:

“Um paradigma de computação distribuída em larga escala e orientado por aspectos econômicos, em que um conjunto abstrato de recursos virtualizados de computação, armazenamento, plataforma e serviços são dinamicamente escaláveis e entregues sob demanda para clientes externos através da Internet.”

A primeira definição de computação em nuvem, ou definição do NIST [21], enfatiza a comodidade quanto à utilização de recursos computacionais configuráveis sob demanda, sem necessidade de grande intervenção gerencial ou operacional. A segunda definição traz a característica de adaptabilidade dos recursos computacionais em relação à demanda variável [51], assim como a característica de pagamento pelo período de utilização dos recursos. A terceira definição [52], explora o conceito de transparência quanto ao provimento de um ou múltiplos recursos, agrupando-os em níveis comuns de qualidade de serviço. Por fim, a definição de Foster e co-autores [20] aborda o caráter econômico como fator determinante para orientar o provimento dinâmico dos recursos abstratos. Em todas as definições, o uso da virtualização como meio de abstrair detalhes específicos de hardware é um consenso.

Na Seção 2.1, é descrita a evolução histórica da tecnologia de computação em nuvem, desde o surgimento da arquitetura em *cluster*, passando por *grid computing* e *peer-to-peer*. A Seção 2.2 descreve as características comuns, como virtualização, utilização sob demanda, provisionamento dinâmico e elasticidade. A Seção 2.3 detalha os níveis de abstração em relação à infraestrutura de computação em nuvem. As fronteiras da computação em nuvem em relação às organizações são discutidas na Seção 2.4. Finalmente, na Seção 2.5 são detalhadas as principais arquiteturas para computação em nuvem.

2.1 Evolução Histórica

Nas décadas de 1980 e 1990, houve uma grande evolução da Internet devido à utilização do protocolo Transmission Control Protocol/Internet Protocol ou TCP/IP. A popularização na utilização de computadores pessoais tornou o modelo cliente-servidor bastante utilizado e transferências de arquivos entre sites começaram a ganhar importância. A capacidade de processamento dos grandes computadores evoluiu para o conceito de *cluster*. Ao final da década de 1990 surgiu o conceito de computação em grade (ou *grid computing*), tendo como foco o aproveitamento de ciclos de processamento ociosos em ambientes heterogêneos [53]. O conceito de P2P (*peer-to-peer*) também surgiu nessa década, com a ideia de um sistema onde nós se auto-organizavam em diferentes topologias, operando sem um controle central [54]. Outro importante conceito que ganhou impulso

na época foi o de virtualização, que mais tarde se tornaria um dos pilares para a computação em nuvem. A ideia de ubiquidade [55], ou computação pervasiva, começa a ser bastante difundida, sendo que a facilidade em alocação e distribuição de recursos a qualquer momento, de forma transparente em relação à localização física do usuário, só fizeram crescer a popularidade desse conceito. Outra tendência importante que surge nessa época é o de processamento em larga escala de dados utilizando a técnica de MapReduce [56], popularizada por ferramentas como o Apache Hadoop [57].

O objetivo da computação em grade ou *grid computing* consistia em estabelecer uma agregação de recursos em um ambiente dinâmico, abstraindo a complexidade das fontes computacionais [53]. Para isso, a computação em grade dependia da utilização de protocolos padrão de rede e de *middleware* para mediar o acesso a essas fontes heterogêneas. Três pontos importantes são observados na arquitetura em *grid*: coordenação de recursos que não estão sujeitos a um controle centralizado (podem estar em diferentes domínios), uso de interfaces e protocolos abertos e a entrega de qualidade de serviço atendendo às necessidades do usuário. Um dos maiores desafios neste modelo de computação distribuída consistia no escalonamento de tarefas de forma eficiente [58]. A arquitetura mais difundida e padronizada foi proposta pela *Open Grid Service Architecture* (OGSA), criando o conceito de *grid* de serviços. Esse modelo evoluiu para o modelo Globus Toolkit 5, ao qual foram incorporadas diversas evoluções [59].

Grids e sistemas *peer-to-peer* são sistemas computacionais distribuídos dirigidos pela necessidade de compartilhamento de recurso, mas com propósitos diferentes. Enquanto o foco principal da computação em *grid* foi a comunidade científica, envolvendo aplicações complexas, as quais requerem garantias de tempo e qualidade de serviço, o alvo dos sistemas P2P é, em geral, o usuário comum. Há diferenças significativas entre o P2P e sistemas em *grid*, entretanto, combinando-se os dois, obtém-se vantagens como a alta escalabilidade e a diminuição nos custos de manutenção. Um ponto a ser destacado no sistema de P2P é que, como sua organização é composta por nós e esses não possuem um comando central, a ideia de quem é servidor ou cliente não existe [60].

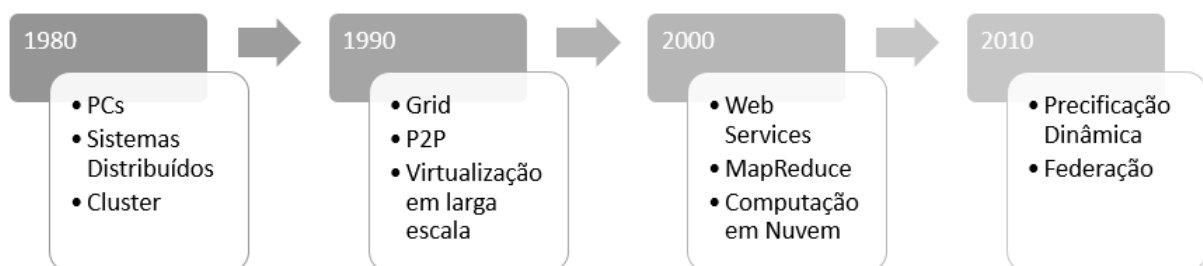


Figura 2.1: Tecnologias que influenciaram o surgimento e a evolução da computação em nuvem nas últimas décadas, adaptado de [1].

Os fundamentos do conceito de computação em nuvem surgiram na década de 1960 a partir das ideias desenvolvedores da ARPANET¹, rede do Departamento de Defesa dos Estados Unidos precursora da Internet, que imaginavam a computação na forma de uma rede global onde a computação seria oferecida como um serviço de utilidade pública. Alguns dos primeiros usos foram vistos no processamento de transações financeiras e dados do censo norte-americano.

A computação em nuvem propriamente dita somente se tornou difundida a partir da década de 2000, quando as empresas começaram a pensar em trocar o seu parque instalado por serviços em nuvem, atraídas pelos benefícios como redução nos custos, aumento de eficiência e simplificação em relação à operação de ativos de tecnologia da informação e, mais recentemente, para reduzir o consumo de energia elétrica e a emissão de carbono.

A partir de 2010, a evolução da computação em nuvem possibilitou o surgimento de dois conceitos importantes. O primeiro se refere ao modelo de precificação dinâmico de recursos [61], que possibilita aproveitamento de recursos a um custo variável, dependendo da capacidade de provisionamento por parte do provedor e da demanda estabelecida pelo usuário. Outro conceito se refere a federação de nuvens [62], que estabelece a possibilidade de interoperabilidade e portabilidade entre nuvens, importante para proteção do investimento e diminuição de custos por parte do usuário. A Figura 2.1 apresenta as tecnologias que influenciaram o surgimento da computação em nuvem, assim como sua evolução desde a década de 1980 até os dias atuais.

2.2 Características da Computação em Nuvem

A computação em nuvem apresenta características essenciais em relação ao compartilhamento e ao provisionamento de recursos de infraestrutura computacional, que são [21]:

Serviço autônomo e sob demanda: um consumidor pode solicitar provisionamento de capacidades computacionais, tais como tempo de processamento, espaço de armazenamento e aumento de largura de banda, sem a necessidade de interação humana com o provedor de serviço;

Amplio acesso à rede: os recursos estão disponíveis por meio da Internet e podem ser acessados através de mecanismos padronizados por plataformas heterogêneas e de diferentes escalas (por exemplo, celulares, *tablets*, *laptops* e estações de trabalho);

Pool de recursos: recursos computacionais do provedor são reunidos para servir vários consumidores usando um modelo *multi-tenant* (ou multi-inquilino), com diferentes

¹<https://www.darpa.mil/about-us/timeline/arpamet>

recursos físicos e virtuais atribuídos dinamicamente e reatribuídos de acordo com a demanda dos consumidores. Há um senso de independência de localização em que o cliente não tem controle ou conhecimento sobre a localização exata dos recursos disponibilizados, mas pode ser capaz de especificar o local em um nível mais alto de abstração (por exemplo, país, estado ou *datacenter*). Exemplos de recursos incluem o armazenamento, o processamento, a memória e a largura de banda de rede;

Elasticidade rápida: os recursos podem ser provisionados e liberados rapidamente e dinamicamente, de forma que a escala seja alterada de acordo com a demanda. Para o consumidor, os recursos disponíveis para provisionamento muitas vezes parecem ser ilimitados e podem ser utilizados em qualquer quantidade e a qualquer momento;

Serviço medido: sistemas em nuvem automaticamente controlam e otimizam a utilização de recursos, aproveitando a capacidade de medição em algum nível de abstração apropriado para o tipo de serviço como, por exemplo, armazenamento, processamento e largura de banda. O uso de recursos pode ser monitorado, medido e controlado, tanto por parte do provedor quanto do consumidor do serviço.

Do ponto de vista de utilização de infraestrutura computacional, a computação em nuvem apresenta as seguintes características inovadoras [63]:

- Ilusão de recursos de infraestrutura computacional infinitos, o que elimina a necessidade de planejamento prévio em relação ao provisionamento;
- Ausência de necessidade quanto ao pagamento adiantado pela utilização da infraestrutura computacional alocada;
- Possibilidade de pagamento pelo uso de infraestrutura por intervalos de tempos curtos (algumas horas).

O compartilhamento de recursos computacionais para obtenção de maior eficiência quanto ao aproveitamento da infraestrutura provisionada é uma característica cada vez mais presente na computação em nuvem. O interesse surge tanto por parte dos provedores, que buscam melhor aproveitamento energético e aumento de lucro, quanto dos usuários, que procuram formas de reduzir custos [64].

2.3 Modelo de Serviços

Os serviços oferecidos na computação em nuvem são classificados de acordo com o nível de abstração em relação à infraestrutura disponibilizada [21], da seguinte forma:

Software como Serviço (*Software as a Service* ou SaaS): consiste em fornecer ao consumidor capacidade de utilizar aplicações do provedor que se executam em infraestrutura de nuvem. As aplicações são acessíveis a partir de vários dispositivos clientes, através de uma interface simplificada, como um navegador. O consumidor não gerencia ou controla a infraestrutura de nuvem subjacente, incluindo rede, servidores, sistemas operacionais, armazenamento ou até mesmo recursos de aplicação, com exceção a configurações de usuário específicas e limitadas;

Plataforma como Serviço (*Platform as a Service* ou PaaS): consiste em fornecer ao consumidor capacidade de implantar aplicações desenvolvidas ou adquiridas usando linguagens de programação, bibliotecas e ferramentas suportadas pelo provedor. O consumidor não gerencia ou controla a infraestrutura de nuvem subjacente, incluindo rede, servidores, sistemas operacionais, armazenamento, mas tem o controle sobre as aplicações implantadas e configurações do ambiente de hospedagem das aplicações;

Infraestrutura como Serviço (*Infrastructure as a Service* ou IaaS): consiste em fornecer ao consumidor capacidade de provisionamento de processamento, armazenamento, acesso à rede e outros recursos computacionais fundamentais, em que o consumidor é capaz de implantar e executar software arbitrário, que pode incluir bibliotecas, compiladores e aplicativos. O consumidor não gerencia ou controla a infraestrutura de nuvem subjacente, mas tem o controle sobre configurações de sistema operacional, armazenamento e aplicativos implantados, assim como controle, geralmente limitado, de componentes de rede (por exemplo, *firewalls*).

Independente do modelo de serviço adotado, é importante que sejam observadas e acordadas entre provedores e usuários as características de disponibilidade, segurança, desempenho e qualidade dos serviços provisionados [21].

Recentemente, vários modelos para definir “qualquer coisa como serviço” (*Everything as a Service* ou XaaS) [65] tem sido propostos no contexto de computação em nuvem, incluindo discussões sobre produtos, processos, dados, informações e segurança como serviço, dentre outros. Um desses modelos, denominado *Function-as-a-Service* ou FaaS, define a execução de código em resposta a eventos, sem existência de infraestrutura complexa normalmente associada a microsserviços [66].

2.4 Modelo de Implantação

Em relação às fronteiras da computação em nuvem e seu posicionamento em uma ou mais organizações participantes, o modelo de serviço pode ser classificado como [21]:

Nuvem Privada: a infraestrutura de nuvem é provisionada para uso exclusivo por uma organização, que compreende vários consumidores (por exemplo, unidades de negócio). Pode ser gerenciada e operada pela própria organização que a utiliza, por uma organização externa, ou por uma combinação de ambas.

Nuvem Comunitária: a infraestrutura de nuvem é provisionada para uso exclusivo de uma comunidade específica de consumidores de organizações que compartilham características (por exemplo estratégia, missão, políticas de segurança e de *compliance*). Assim como na nuvem privada, pode ser gerenciada e operada por uma ou mais organizações participantes da comunidade, por uma organização externa, ou por uma combinação de ambas.

Nuvem Pública: a infraestrutura de nuvem é provisionada para uso aberto pelo público em geral. Pode ser gerenciada e operada por uma empresa, instituição acadêmica, organização governamental, ou alguma combinação delas.

Nuvem Híbrida: a infraestrutura em nuvem é uma composição de duas ou mais infraestruturas distintas de nuvem (privada, comunitária ou pública), que permanecem como entidades únicas, mas que são unidas por tecnologia padronizada ou proprietária, que permite portabilidade de dados e aplicação.

O uso dinâmico de nuvens públicas e híbridas proporciona a aplicação do conceito de *cloud bursting*, que é a ampliação temporária da capacidade alocada para atendimento a uma demanda não prevista [21]. Este conceito abrange três grandes vantagens do ponto de vista do usuário [67]: (1) ajuste de escalabilidade em função da carga de trabalho, (2) utilização eficiente de recursos e (3) maiores garantias quanto à disponibilidade da aplicação.

2.5 Arquitetura de Computação em Nuvem

Existem diversas arquiteturas propostas para computação em nuvem, com enfoques acadêmico e comercial. No caso dos provedores de nuvem pública, como Amazon Web Services², Google Cloud Platform³ e Microsoft Azure⁴ é comum a definição de camadas de

²<http://aws.amazon.com/>

³<http://cloud.google.com/>

⁴<http://azure.microsoft.com/>

abstração, de forma a tratar os diferentes níveis de detalhamento e complexidade em relação à configuração e administração dos recursos físicos. Nesse sentido, a preocupação importante em relação à definição de arquiteturas para computação em nuvem se refere à complexidade para o usuário em relação aos procedimentos e passos necessários, desde a definição do tipo de recurso desejado até a execução da aplicação propriamente dita.

2.5.1 Modelo em Camadas

A arquitetura de computação em nuvem mostrada na Figura 2.2 se baseia na modelagem em camadas, levando em consideração o modelo de serviço, partindo do hardware físico na camada mais inferior e aumentando-se o nível de abstração, na medida em que as camadas superiores são introduzidas [2].

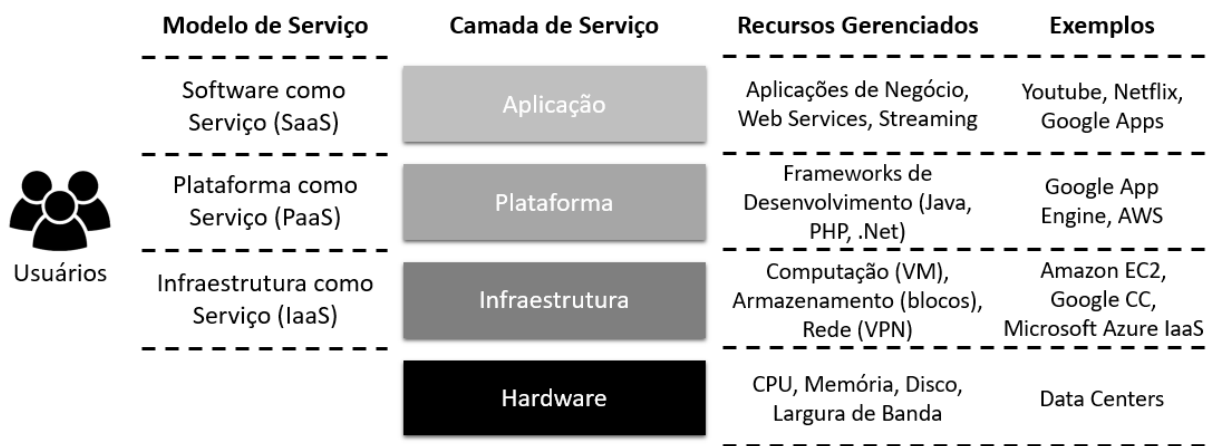


Figura 2.2: Arquitetura de Computação em Nuvem, adaptado de [2].

Na camada de Hardware são gerenciados recursos de infraestrutura computacional em nível físico, como CPU, memória RAM, recurso de armazenamento secundário (disco rígido ou outras mídias) e largura de banda. Esta camada se traduz em ativos físicos, como servidores, *storages*, *switches*, roteadores e outros equipamentos, normalmente localizados em *datacenters*.

Na camada seguinte, de Infraestrutura, são provisionados os recursos virtualizados, como máquinas virtuais (*Virtual Machines* ou VM), armazenamento secundário e redes virtuais. Os provedores de computação em nuvem no modelo *IaaS* se inserem nesta camada, como Amazon Elastic Compute Cloud (Amazon EC2), Google Compute Engine (Google CC) e Microsoft Azure IaaS.

Na camada de Plataforma, os componentes para desenvolvimento distribuído, como bibliotecas, *frameworks*, *middlewares*, servidores de aplicação, mensageria e banco de dados são provisionados, seguindo o modelo proposto de plataforma como serviço ou

PaaS. Os provedores de computação em nuvem que ofertam serviços no modelo *IaaS* normalmente também ofertam serviços no modelo *PaaS*, como Google e Amazon.

Na camada mais superior, serviços e aplicações específicos de negócio são oferecidos, de forma que a complexidade relacionada à gestão e configuração de recursos de infraestrutura é completamente transparente para o usuário final. Alguns exemplos de aplicações relacionadas ao compartilhamento e edição colaborativa de documentos, como Google Apps⁵, e de *streaming* de vídeo, como Netflix⁶, utilizam o modelo de software como serviço, em que o usuário usufrui dos recursos por meio de uma assinatura mensal.

2.5.2 Utilização de Nuvem IaaS

No modelo *IaaS*, o provedor de nuvem oferece aos usuários um conjunto de recursos virtualizados. Os usuários são responsáveis por solicitar recursos virtuais (por exemplo máquinas virtuais) e gerenciá-los enquanto o provedor é responsável por (a) selecionar um recurso físico e instanciar a máquina virtual nele; (b) monitorar o recurso físico de maneira a garantir a disponibilidade; e (c) cobrar pelo uso.

A Figura 2.3 apresenta uma arquitetura *IaaS* genérica. Como pode ser visto, acima no nível de hardware e virtualização, existe um gerente de infraestrutura virtual que cria recursos virtuais. Os serviços são organizados no *Service Layer* e incluem máquinas virtuais, *storage*, rede e imagens de máquinas virtuais. Esta camada também define como será realizada a contabilização dos custos pela utilização dos recursos por parte do usuário, de acordo com os modelos de precificação definidos pelo provedor de computação em nuvem.

Os serviços de computação são geralmente oferecidos como instâncias de máquinas virtuais. Uma máquina virtual pertence a um tipo de instância que determina características como número de *cores* (i.e. vCPUs), quantidade de memória RAM, capacidade de disco e capacidade de rede.

Para utilizar o serviço de nuvem *IaaS*, o usuário deve [68]: (1) escolher uma região; (2) selecionar um tipo de instância; (3) selecionar uma imagem de máquina virtual; (4) selecionar um tipo de disco; (5) solicitar que o provedor crie a máquina virtual (VM); (6) configurar a VM; (7) executar a aplicação; (8) transferir os dados de saída da nuvem para a máquina local; (9) terminar a VM.

Uma região (1) é um ambiente de nuvem independente composto por um ou mais *datacenters* em uma determinada região geográfica. Os grandes provedores de nuvem pública estão presentes em diversas regiões ao redor do mundo. Por exemplo, atualmente a Amazon possui 11 regiões, sendo uma delas em São Paulo (sa-east-1). Após escolher

⁵<http://gsuite.google.com/>

⁶<http://www.netflix.com/>

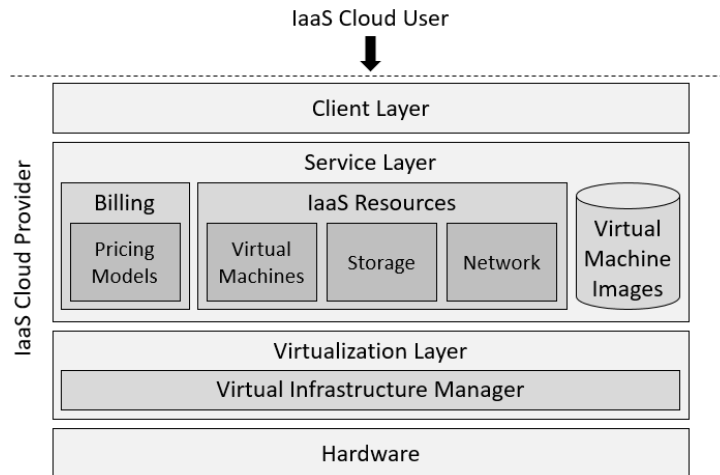


Figura 2.3: Arquitetura de Computação em Nuvem no modelo de serviço *IaaS*, adaptado de [1].

a região, o usuário deve selecionar o tipo de instância (2), considerando requisitos como número mínimo de *cores*, quantidade mínima de memória, preço máximo, entre outros. Geralmente, existem muitas instâncias que atendem os requisitos e os usuários muitas vezes selecionam uma delas, de maneira *ad-hoc*, o que pode levar a alto custo financeiro. Depois da escolha da instância, o usuário deve escolher uma imagem de máquina virtual (VMI) (3), que depende basicamente do sistema operacional suportado pela aplicação. Após isso, o usuário seleciona o tipo de disco (4), caso deseje usá-lo. Tendo a região, o tipo da instância, a VMI e o tipo de disco, o usuário solicita que o provedor de nuvem crie a VM (5). Para tanto, o provedor de nuvem seleciona, de maneira transparente, uma máquina física para abrigar a VM e copia a VMI para essa máquina, solicitando que o *layer* de virtualização inicialize a VM, atribuindo um IP à mesma e iniciando a cobrança por seu uso. Com a VM iniciada, os usuários podem configurá-la (6), instalando sua aplicação e movendo dados de entrada para a mesma. No passo (7), a aplicação é executada. Ao final da execução (8), o usuário transfere os dados de saída para a sua máquina local e libera a VM (9). Nesse ponto, a cobrança é interrompida.

Capítulo 3

Precificação em Computação em Nuvem

Políticas de precificação em computação em nuvem tem como objetivo o estabelecimento de regras para comercialização dos recursos computacionais entre provedores de serviço e usuários. Do ponto de vista do provedor, a política atende aos objetivos de maximização do lucro e melhoria na eficiência quanto à utilização dos recursos, evitando-se assim a ociosidade dos mesmos. Do lado do usuário, a política proporciona a definição da estratégia de alocação dos recursos, de modo a minimizar os custos e maximizar as características de infraestrutura e capacidade computacional. Portanto, a política de precificação estabelece o mecanismo que proporciona as transações de compra e venda temporária de recursos computacionais virtuais entre provedores e usuários, considerando capacidade, demanda, requisitos, utilidades e outros fatores econômicos [69].

3.1 Tipos de Precificação

Considerando o modelo de serviço de computação em nuvem do tipo Infraestrutura como Serviço ou *IaaS*, conforme definido na Seção 2.3, a precificação em computação em nuvem pode ser classificada quanto à variação de preços e possibilidade de revogação de instâncias.

Considerando a definição de preços de recursos a serem provisionados, os modelos de precificação podem ser classificados como estáticos ou dinâmicos. No primeiro caso, o preço é estático e não varia no decorrer do tempo de utilização do recurso, permitindo ao usuário a previsibilidade de custos ao término do período contratado. Já no caso da precificação dinâmica, o preço de provisionamento de um determinado tipo de recurso varia no decorrer do tempo em razão de um ou mais fatores, como a demanda existente

por aquele recurso, não permitindo ao usuário a previsibilidade exata de custos ao final do período contratado.

Os modelos de precificação também podem ser classificados como permanentes ou transientes, dada a possibilidade de revogação da instância por parte do provedor de nuvem. Esta característica tem forte influência nos custos das instâncias contratadas e impõe ao usuário a necessidade de tratamento de interrupções no fornecimento dos recursos. A Figura 3.1 mostra os critérios de classificação de precificação.

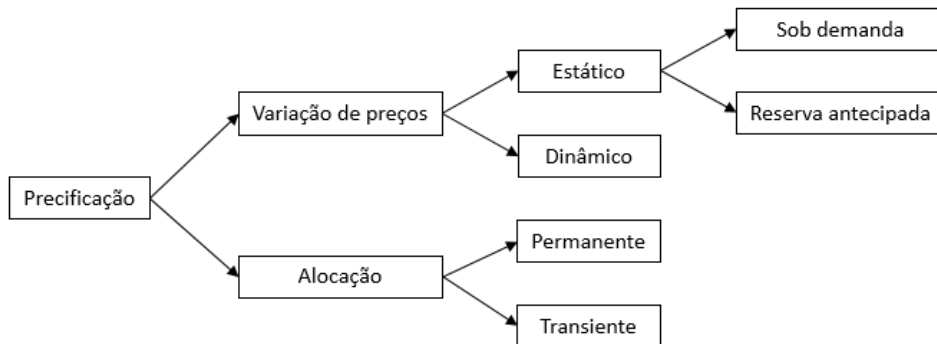


Figura 3.1: Critérios para classificação de precificação em nuvem.

Independente do modelo utilizado, o conceito de *pay-as-you-go* se aplica, ou seja, o usuário paga pela infraestrutura de forma proporcional ao tempo utilizado [70]. Essa característica faz com que a computação em nuvem possibilite o menor investimento em ativos de infraestrutura, reduzindo o custo de propriedade e proporcionando às organizações o redirecionamento de investimentos para o negócio propriamente dito [71].

3.1.1 Precificação Estática

No modelo de precificação estática, o provedor de acesso determina o preço a ser pago pelo usuário de forma fixa e em função do tempo de utilização da infraestrutura. Portanto, este preço invariável possibilita ao usuário a realização do planejamento prévio de custos em função da infraestrutura a ser utilizada. O pagamento é realizado após a utilização dos recursos computacionais, sendo também comumente conhecido como modelo de precificação sob demanda (*on demand*).

Uma variação do modelo de precificação estático conhecida como reserva antecipada (*reserved*) consiste na realização de reserva de recursos de infraestrutura por muito tempo (por exemplo, um ano) e com pagamento adiantado. Assim, o usuário arca com o risco de contratar uma infraestrutura computacional e mantê-la por um longo período pré-determinado, havendo ou não necessidade dos recursos contratados. O valor pago ao provedor é ligeiramente menor do que no modelo sob demanda, cerca de 5% inferior nas

horas de pico de utilização, entre segundas e sextas-feiras, e até 10% nas demais horas, aos sábados e domingos [72]. Por parte do provedor de nuvem, essa variação do modelo de precificação estático é bastante vantajosa, pois estabelece uma previsibilidade quanto ao provimento dos recursos e, conseqüentemente, melhor aproveitamento da infraestrutura disponível.

3.1.2 Precificação Dinâmica

O modelo de precificação dinâmica se baseia em flutuação do preço em função da demanda, sendo que o provedor oferece recursos de infraestrutura computacional a um preço variável, dependendo da quantidade ofertada e da demanda existente. Dessa forma, os usuários que estão dispostos a pagar mais conseguem contratar os recursos computacionais com probabilidade mais alta do que os usuários que possuem grandes restrições de custo. Assim sendo, se houver pouca demanda, os recursos podem ser contratados por valores significativamente menores do que os praticados no modelo estático. Normalmente, são utilizados modelos de mercado baseados em leilões, modelos teóricos e econômicos de jogos, técnicas de *machine learning* e simulações para a intermediação das contratações por meio de precificação dinâmica [61].

Uma característica importante que difere a precificação dinâmica em relação à estática é o fato de que se o preço dos recursos computacionais ultrapassar o valor que o usuário está disposto a pagar, o provimento do recurso é interrompido. Isso requer que a aplicação do usuário esteja preparada para situações desse tipo, de modo que o processamento não se perca e possa ser retomado posteriormente.

3.1.3 Precificação Transiente

A precificação dinâmica é utilizada pela Amazon para oferta de instâncias *spot* com objetivo de minimizar a capacidade ociosa de sua infraestrutura [24]. De forma semelhante, o Google fornece instâncias preemptíveis de modo a aproveitar a sua infraestrutura não utilizada [73]. Ambos os modelos de precificação dinâmica da Amazon e de instâncias preemptíveis do Google podem ser caracterizados como modelos de precificação de instâncias transientes. O conceito “transiente” se refere à possibilidade de o provedor poder revogar as instâncias unilateralmente dos usuários, de modo a possibilitar atendimento a requisições com maior prioridade, ou que proporcionam maior lucro [74].

As instâncias transientes, geralmente, possuem preços mais baixos em relação às instâncias de preço fixo, em razão de não haver garantia quanto à disponibilidade futura da instância. Para o usuário, pode proporcionar economia, desde que sua aplicação seja tolerante a interrupções, o que normalmente traz alguma sobrecarga de processamento em

função de tratamento de *checkpoints* [74]. Para o provedor de nuvem, proporciona maior utilização de recursos ociosos, assim como flexibilidade com a possibilidade de revogação de recursos provisionados.

3.2 Amazon Web Services (AWS)

No provedor AWS, instâncias de máquinas virtuais são oferecidas por meio do serviço Amazon Elastic Compute Cloud (Amazon EC2) em três principais modelos de precificação [75]: *on demand*, *reserved* e *spot*. Nos dois primeiros, o preço é estático por hora, não havendo variação. O terceiro modelo de precificação denominado *spot* é caracterizado como dinâmico e transiente, pois é utilizado um mecanismo de mercado que faz com que o preço das instâncias varie em função da oferta e demanda de recursos computacionais. O modelo de precificação *spot* divide-se em *spot* tradicional, descrito acima, *spot fleet* e *spot block*. O *spot fleet* permite que um conjunto de instâncias sejam tratadas de maneira única, enquanto o *spot block* permite que o usuário tenha a garantia de que a instância esteja disponível por até 6 horas.

O uso das tecnologias de contêineres [76] e *serverless* [77] se tornou bastante popular como alternativa ao uso de instâncias de máquinas virtuais. No entanto, serviços de nuvem pública no modelo IaaS bem conhecidos, como Amazon EC2, oferecem instância de máquina virtual no modelo de infraestrutura como serviço, seja para processamento de aplicações ou até mesmo provimento de contêineres e outros serviços. Portanto, as análises de custos e disponibilidade se concentrarão na utilização de instâncias de máquinas virtuais.

3.2.1 Amazon EC2 *on demand*

No modelo Amazon EC2 *on demand*, o usuário paga em dólar por hora (US\$/*hour*) sem necessidade de contratação por tempo mínimo [17]. É o modelo que traz mais comodidade ao usuário, uma vez que não é preciso fazer qualquer previsão com relação ao tempo ou à quantidade de recursos que serão alocados. Por outro lado, é considerado um modelo caro [78].

No serviço Amazon EC2, as instâncias, ou máquinas virtuais, são classificadas de acordo com o propósito de utilização, sendo para uso geral (*general purpose*) ou otimizada para algum tipo de uso, como memória (*memory optimized*), CPU (*CPU optimized*), unidade gráfica (*GPU optimized*) ou armazenamento (*storage optimized*). A Tabela 3.1 mostra cinco tipos de instâncias disponíveis para cada propósito, assim como suas características de infraestrutura e preços por hora de utilização, considerando instâncias com sistema operacional Linux e na localidade da Virgínia do Norte (*North Virginia*), leste dos

Estados Unidos (*US-East*). Em 23/09/2017 existiam 59 tipos de instâncias disponíveis para provimento neste modelo de precificação.

Tabela 3.1: Tipos de instâncias *on demand* disponíveis no serviço Amazon EC2 [17].

| Tipo de Instância | Quantidade de vCPU | ECU | Memória (GB) | Preço (USD / hora) | Propósito |
|-------------------|--------------------|----------|--------------|--------------------|-------------------|
| t1.micro | 1 | Variável | 0,6 | \$0,02 | General purpose |
| t2.medium | 2 | Variável | 4 | \$0,0464 | General purpose |
| m4.large | 2 | 6,5 | 8 | \$0,1 | General purpose |
| m4.10xlarge | 40 | 124,5 | 160 | \$2,0 | General purpose |
| m3.2xlarge | 8 | 26 | 30 | \$0,532 | General purpose |
| c4.large | 2 | 8 | 3,75 | \$0,1 | Compute Optimized |
| c4.4xlarge | 16 | 62 | 30 | \$0,796 | Compute Optimized |
| c4.8xlarge | 36 | 132 | 60 | \$1,591 | Compute Optimized |
| c3.4xlarge | 16 | 55 | 30 | \$0,84 | Compute Optimized |
| c3.8xlarge | 32 | 108 | 60 | \$1,68 | Compute Optimized |
| x1.16xlarge | 64 | 174,5 | 976 | \$6,669 | Memory Optimized |
| r3.large | 2 | 6,5 | 15 | \$0,166 | Memory Optimized |
| r3.2xlarge | 8 | 26 | 61 | \$0,665 | Memory Optimized |
| r4.large | 2 | 7 | 15,25 | \$0,133 | Memory Optimized |
| r4.2xlarge | 8 | 27 | 61 | \$0,532 | Memory Optimized |
| p2.xlarge | 4 | 12 | 61 | \$0,9 | GPU Instances |
| p2.8xlarge | 32 | 94 | 488 | \$7,2 | GPU Instances |
| g2.2xlarge | 8 | 26 | 15 | \$0,65 | GPU Instances |
| g3.4xlarge | 16 | 47 | 122 | \$1,14 | GPU Instances |
| g3.8xlarge | 32 | 94 | 244 | \$2,28 | GPU Instances |
| i3.large | 2 | 7 | 15,25 | \$0,156 | Storage Optimized |
| i3.4xlarge | 16 | 53 | 122 | \$1,248 | Storage Optimized |
| d2.xlarge | 4 | 14 | 30,5 | \$0,69 | Storage Optimized |
| d2.4xlarge | 16 | 56 | 122 | \$2,76 | Storage Optimized |
| d2.8xlarge | 36 | 116 | 244 | \$5,52 | Storage Optimized |

Como pode ser visto, há uma grande variação de preços e configurações, o que torna a escolha do tipo de instância uma tarefa complexa para o usuário. É importante observar que os valores em ECU (*EC2 Compute Unit*) se referem a uma medida comparativa de arquitetura computacional, na qual um ECU é equivalente à capacidade de um processador Opteron ou Xeon 2007, com velocidade de 1.0 a 1.2 GHz.

Em relação ao espaço de armazenamento, o provedor Amazon oferece o serviço *Elastic Block Storage* (EBS), que fornece volumes de armazenamento em blocos persistentes para uso com instâncias do serviço EC2 [3]. Cada volume EBS é replicado automaticamente e internamente na zona de disponibilidade (região geográfica) para proteção quanto a falhas de componente de hardware [79].

3.2.2 Amazon EC2 *reserved*

Outra forma de contratar instâncias da Amazon é por meio de reserva de recursos (*reserved instances*). Este modelo possibilita um desconto significativo (até 75%) em comparação com o preço *on demand* [80]. As reservas adiantadas podem ser feitas para períodos de 1 ou 3 anos, sendo este último período o que proporciona maiores descontos. Ambos os modelos *on demand* e *reserved* são classificados como modelos de preço estático.

O serviço Amazon EC2 disponibiliza 3 opções de pagamento adiantado [80]: a opção *All Upfront*, em que o usuário paga por todo o prazo da instância reservada em um pagamento inicial. Esta opção fornece o maior desconto em comparação com o preço da *on demand*. Com a opção *Partial Upfront*, o usuário faz um pagamento antecipado inicial e, em seguida, é cobrada uma taxa descontada por hora durante o período reservado. A opção *No Upfront* não requer nenhum pagamento antecipado e fornece uma taxa por hora com desconto para a duração do prazo.

3.2.3 Amazon EC2 *spot*

Por último, a Amazon oferece o modelo *spot* de precificação, que utiliza um mecanismo de leilão para oferta de instâncias virtuais [75]. Esta oferta é definida por instâncias ociosas que não foram comercializadas no preço fixo, sendo estabelecido um preço em função da demanda dos usuários, que oferecem lances. Caso o lance esteja acima do preço estabelecido, a instância é fornecida ao usuário, caso contrário, é interrompida.

Para fins de comparação de preços entre os três modelos, é considerado um tipo de instância denominado *c3.8xlarge*, cuja configuração está descrita na Tabela 3.1. Essa instância possui preço fixo definido em \$1,68/hora no modelo *on demand* e \$1,168/hora no modelo *reserved*, sendo que o preço médio no modelo *spot* é de \$0,55/hora, considerando o período de setembro a novembro de 2016 (3 meses), para instâncias disponíveis na costa leste dos Estados Unidos (US East), utilizando o sistema operacional Unix/Linux.

Segundo a Amazon, o modelo de instâncias *spot* é adequado para as seguintes cargas de trabalho [24]:

- Processamento analítico: análises complexas, como a varredura de *log* ou simulações, geralmente realizadas como trabalhos em lote (*batch processing*);
- Computação Científica: execução de simulações que vão desde a descoberta de novos medicamentos até a pesquisa genômica;
- Codificação de imagens e mídias: processamento e codificação de ativos de mídia, como imagens, vídeos ou conteúdo digital específico;
- Análise geoespacial: renderização e processamento de imagem de satélite.

Para usar instâncias *spot*, o usuário deve realizar uma requisição que inclui o preço máximo que o usuário está disposto a pagar por hora e por instância (preço de lance) e outras restrições, como o tipo de instância e a zona de disponibilidade (localização do *datacenter*). Se o preço do lance for maior do que o preço *spot* atual para a instância especificada e houver disponibilidade do tipo de instância, a solicitação é atendida imediatamente. Caso contrário, o pedido é atendido assim que o preço *spot* cair abaixo do preço

de lance ou a instância especificada ficar disponível. As instâncias são disponibilizadas até que o usuário as encerre ou até que a Amazon as encerre (também conhecida como interrupção de instância *spot*). Neste último caso, o provedor envia uma notificação de término ao usuário e interrompe a alocação da instância em 2 minutos [24].

Quando a requisição é atendida, ou seja, quando há disponibilidade do tipo de instância desejada e o preço de lance é maior do que preço *spot* atual, o usuário paga somente o preço o *spot* durante o tempo de atendimento. Portanto, durante este período o custo para o usuário é sempre menor ou igual ao preço de lance.

O preço *spot* é determinado pelo menor lance para um tipo de instância. Por exemplo, suponha que um usuário crie uma requisição de instância *spot* e que haja somente cinco instâncias daquele tipo. Seu preço de oferta é \$0,15/hora. A Tabela 3.2 mostra os lances classificados em ordem decrescente, sendo que as propostas 1 a 5 são atendidas no instante t . O lance 5, que é a última oferta atendida, define o preço *spot* em \$0,10, que é o preço pago por todos os usuários atendidos. O lance 6 não é atendido.

Tabela 3.2: Atendimento das requisições dependendo do preço do lance e da disponibilidade do tipo de instância.

| Item | Preço do Lance | Preço spot (t) | Observação (t) | Preço spot (t+Δ) | Observação (t+Δ) |
|------|----------------|----------------|---------------------------|------------------|---------------------------|
| 1 | \$0,35 | \$0,10 | | \$0,20 | |
| 2 | \$0,30 | \$0,10 | | \$0,20 | |
| 3 | \$0,20 | \$0,10 | | \$0,20 | Último lance atendido |
| | | | | | Limite de capacidade spot |
| 4 | \$0,20 | \$0,10 | | | Lance não atendido |
| 5 | \$0,10 | \$0,10 | Último lance atendido | | Lance não atendido |
| | | | Limite de capacidade spot | | |
| 6 | \$0,05 | | Lance não atendido | | Lance não atendido |

Caso a capacidade de atendimento no instante $(t + \Delta)$ para o tipo de instância seja reduzida para 3 instâncias, apenas os lances 1 a 3 são atendidos e o preço *spot* é redefinido em \$0,20/hora. Caso haja mais de um lance com preços iguais, estes são colocados de forma aleatória, como é o caso dos lances 3 e 4.

Para tornar o modelo *spot* mais adequado para cargas de trabalho com duração definida, a Amazon oferece a variação denominada *spot block*, de forma a permitir que instâncias *spot* sejam executadas continuamente por uma duração finita de 1 a 6 horas [3]. O preço é baseado na duração solicitada e na capacidade disponível, e normalmente é 30% a 45% menor do que o preço *on demand*, com 5% de desconto adicional durante as horas não pico da região. Para isso, o usuário deve enviar uma solicitação de instância *spot* e usar o parâmetro *BlockDuration* para especificar o número de horas que deseja que

sua instância esteja disponível, juntamente com o preço máximo que está disposto a pagar. Quando a capacidade da instância *spot* estiver disponível para a duração solicitada, a instância é iniciada e alocada continuamente por um preço horário fixo. Essa instância é encerrada automaticamente no final do tempo de bloqueio.

Outra variação do modelo *spot* é o *spot fleet*, em que um grupo de instâncias é tratado como uma coleção ou frota de instâncias *spot* [24]. O provedor tenta iniciar toda a coleção de recursos que são necessários para atender a capacidade especificada pelo usuário na solicitação. O provedor também tenta manter a capacidade alvo estabelecida para o *spot fleet* se suas instâncias forem candidatas à interrupção devido a uma mudança nos preços ou na capacidade disponível do provedor. A Figura 3.2 mostra a evolução dos modelos de precificação do serviço Amazon EC2 ao longo do tempo.

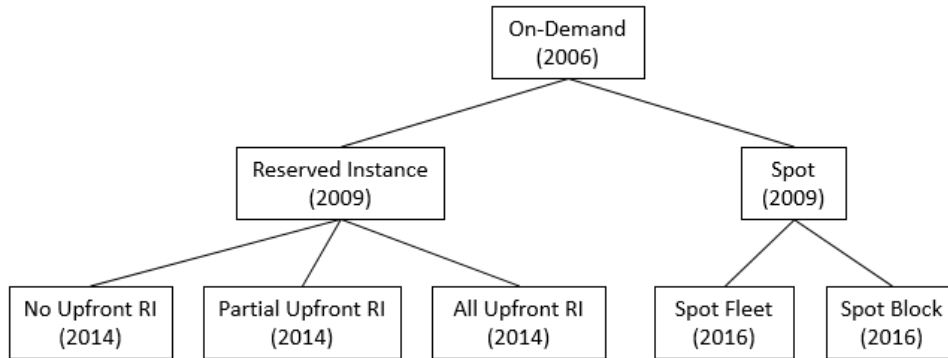


Figura 3.2: Evolução da precificação do serviço Amazon EC2, adaptado de [3].

A Amazon prioriza a comercialização de instâncias no modelo de preço fixo em função do maior lucro e da previsibilidade no uso da infraestrutura. Sendo assim, a capacidade de atendimento para determinado tipo de instância no modelo *spot* depende da procura pelo mesmo tipo de instância no modelo de preço fixo, ou seja, quanto mais houver ociosidade no preço fixo, mais instâncias são oferecidas no modelo *spot*. Entretanto, não são divulgados pela Amazon os valores de capacidade instalada ou volume demandado, apenas o histórico de 3 meses de variação dos preços por tipo de instância [75].

No final de 2017, o mercado *spot* da Amazon mudou, de forma que o valor de lance do usuário (*maximum price*) não precisa ser obrigatoriamente informado na requisição da instância. O preço *spot* continua sendo variável e definido por um mecanismo de mercado que se baseia em oferta e demanda. As instâncias podem ser revogadas a qualquer momento, a critério da Amazon. A existência de um preço máximo opcional é semelhante ao modelo de leilão anterior na perspectiva do usuário [25], portanto, todos os trabalhos anteriores que tratam de lances *spot* permanecem válidos.

3.3 Google Cloud Platform (GCP)

O Google Cloud Platform (GCP) [23] é a plataforma para computação em nuvem do Google, cujo serviço Google Compute Engine (Google CC) provê instâncias de máquinas virtuais no modelo Infraestrutura como Serviço ou *IaaS*. O provedor oferece duas categorias de configuração de hardware: máquinas pré-definidas (*predefined*) ou personalizadas (*custom*). Na primeira categoria, são oferecidas configurações sem compartilhamento de núcleos de processamento (*standard machine*) e configurações que permitem compartilhamento de forma a possibilitar *bursting*, isto é, onde instâncias virtuais usam núcleos de CPU adicionais por curtos períodos de tempo. Além dessas, são oferecidas configurações voltadas para uso intensivo de memória (*high-memory*) e CPU (*high-CPU*). Em todos os casos, a cobrança é feita pelo período mínimo de utilização de 10 minutos, sendo cobrados valores adicionais por minuto de utilização [18].

3.3.1 Google GCP *predefined*

As configurações e preços das instâncias pré-definidas para utilização sob demanda podem ser visualizadas na Tabela 3.3. Como pode ser visto, o Google GCP também classifica as instâncias em grupos de acordo com o propósito de utilização, como instâncias de propósito geral (*standard*), de núcleo compartilhado (*shared-core*), com grande capacidade de memória (*high-memory*) e para processamento (*high-CPU*).

Tabela 3.3: Configurações pré-definidas do GCP sob demanda [18].

| Tipo de Instância | vCPU | Memória | Preço (USD / hora) | Preemptível (USD/hora) | Característica |
|-------------------|------|---------|--------------------|------------------------|----------------|
| n1-standard-1 | 1 | 3,75GB | \$0,0475 | \$0,0100 | Standard |
| n1-standard-2 | 2 | 7,5GB | \$0,0950 | \$0,0200 | Standard |
| n1-standard-4 | 4 | 15GB | \$0,1900 | \$0,0400 | Standard |
| n1-standard-8 | 8 | 30GB | \$0,3800 | \$0,0800 | Standard |
| n1-standard-16 | 16 | 60GB | \$0,7600 | \$0,1600 | Standard |
| n1-standard-32 | 32 | 120GB | \$1,5200 | \$0,3200 | Standard |
| n1-standard-64 | 64 | 240GB | \$3,0400 | \$0,6400 | Standard |
| f1-micro | 1 | 0,60GB | \$0,0076 | \$0,0035 | Shared-core |
| g1-small | 1 | 1,70GB | \$0,0257 | \$0,0070 | Shared-core |
| n1-highmem-2 | 2 | 13GB | \$0,1184 | \$0,0250 | High-memory |
| n1-highmem-4 | 4 | 26GB | \$0,2368 | \$0,0500 | High-memory |
| n1-highmem-8 | 8 | 52GB | \$0,4736 | \$0,1000 | High-memory |
| n1-highmem-16 | 16 | 104GB | \$0,9472 | \$0,2000 | High-memory |
| n1-highmem-32 | 32 | 208GB | \$1,8944 | \$0,4000 | High-memory |
| n1-highmem-64 | 64 | 416GB | \$3,7888 | \$0,8000 | High-memory |
| n1-highcpu-2 | 2 | 1,80GB | \$0,0709 | \$0,0150 | High-CPU |
| n1-highcpu-4 | 4 | 3,60GB | \$0,1418 | \$0,0300 | High-CPU |
| n1-highcpu-8 | 8 | 7,20GB | \$0,2836 | \$0,0600 | High-CPU |
| n1-highcpu-16 | 16 | 14,40GB | \$0,5672 | \$0,1200 | High-CPU |
| n1-highcpu-32 | 32 | 28,80GB | \$1,1344 | \$0,2400 | High-CPU |
| n1-highcpu-64 | 64 | 57,6GB | \$2,2688 | \$0,4800 | High-CPU |

3.3.2 Google GCP *custom*

A segunda categoria de configuração oferecida pelo GCP é a de máquina personalizada, sendo que a configuração é definida de acordo com as características e necessidades do usuário. A partir de uma configuração pré-estabelecida, o usuário pode adicionar ou diminuir o número de CPUs virtuais (vCPU) ou Gigabytes de memória, no qual o custo unitário por hora é adicionado ou abatido ao custo da instância pré-definida escolhida como base para a configuração. A Tabela 3.4 mostra o custo de vCPU e memória por hora para personalização de tipos de instâncias GCP.

Tabela 3.4: Itens de personalização de instâncias do provedor GCP [18].

| Item | Preço <i>on demand</i> (USD/hora) | Preço preemptível (USD/hora) | Quantidade |
|---------|-----------------------------------|------------------------------|-----------------|
| vCPU | \$0,033174/vCPU | \$0,00698/vCPU | |
| Memória | \$0,004446/GB | \$0,00094/GB | Até 6,5 GB |
| Memória | \$0,009550/GB | \$0,002014/GB | Acima de 6,5 GB |

3.3.3 Google GCP *reserved*

Assim como ocorre na Amazon (Seção 3.2.2), as instâncias GCP podem ser contratadas por meio de reserva antecipada durante períodos de 1 ou 3 anos. Nesse modelo de contratação, as instâncias são disponibilizadas e cobradas pelo provedor havendo ou não utilização por parte do usuário. Os preços chegam a ser 57% mais baratos dos valores cobrados no modelo sob demanda. Na Tabela 3.5 são mostrados itens de personalização e exemplos de tipos de instâncias pré-definidas, com seus respectivos preços nos modelos sob demanda e reserva antecipada.

Tabela 3.5: Itens de personalização e configurações pré-definidas de instâncias do GCP no modelo de reserva antecipada [18].

| Tipo de Configuração | Item | Preço (USD) sob demanda | Preço de Reserva por 1 ano (USD) | Preço de Reserva por 3 anos (USD) |
|----------------------|----------------|-------------------------|----------------------------------|-----------------------------------|
| Personalizada | vCPU | \$0,033174/vCPU hora | \$0,019915/vCPU hora | \$0,014225/vCPU hora |
| Personalizada | Memória | \$0,004446/GB hora | \$0,002669/GB hora | \$0,001907/GB hora |
| Pré-definida | n1-standard-16 | \$0,7600/hora | \$0,47878/hora | \$0,34202/hora |
| Pré-definida | n1-highmem-16 | \$0,9472/hora | \$0,59622/hora | \$0,42593/hora |
| Pré-definida | n1-highcpu-16 | \$0,5672/hora | \$0,35707/hora | \$0,25506/hora |

Os preços dos recursos do GCP listados nas Tabelas 3.3, 3.4 e 3.5 se referem à região do Oregon nos Estados Unidos.

3.3.4 Google GCP *preemptible*

Seja qual for o tipo de configuração (pré-definida ou personalizada), as instâncias virtuais podem ser contratadas de forma dedicada ou passível de preempção. Uma instância virtual preemptível é uma máquina virtual que pode ser interrompida pelo GCP, caso o provedor necessite do recurso físico para alocação de instâncias dedicadas [73]. Portanto, instâncias preemptíveis representam excesso capacidade do GCP e sua disponibilidade varia de acordo com a demanda por instâncias dedicadas. Quanto maior for a demanda por instâncias dedicadas, maior é a probabilidade de instâncias preemptíveis serem interrompidas.

A grande vantagem das instâncias preemptíveis consiste na redução de custos para o usuário. Entretanto, de forma semelhante ao modelo *spot* da Amazon (Seção 3.2.3), este tipo de instância é recomendado para aplicações tolerantes a falhas e que podem suportar interrupção em sua execução, como por exemplo tarefas de processamento em lote. No processo de preempção, o usuário recebe uma notificação e, em 30 segundos, a instância é encerrada. Nesse curto período de tempo, a aplicação deve executar os procedimentos necessários para interrupção e futura retomada do processamento, como registro de *checkpoints* e informações de contexto de maneira que o processamento não seja perdido.

Para determinar quais instâncias preemptíveis serão interrompidas, o provedor GCP escolhe as mais recentemente inicializadas, não devendo haver concentração em um único usuário. Todas as instâncias preemptíveis são automaticamente encerradas após 24 horas. O GCP afirma, com base em observação de dados históricos, que estas instâncias são interrompidas entre 5% e 15% do período correspondente a uma semana de execução contínua [73].

3.4 Microsoft Azure

O provedor Microsoft Azure oferece serviços de provisionamento em computação em nuvem, incluindo máquinas virtuais, bancos de dados e ambientes de armazenamento. As máquinas virtuais suportam, além dos sistemas operacionais da própria Microsoft, várias distribuições Linux. Da mesma maneira que nos provedores AWS e GCP, elas são classificadas de acordo com a finalidade de utilização, ou seja, de propósito geral ou otimizada para algum tipo de recurso, como CPU, GPU, memória, armazenamento ou para processamento em alto desempenho (*high performance computing* ou HPC).

O provedor trabalha principalmente com o modelo de precificação *pay-as-you-go*, em que o usuário paga somente pelo tempo de utilização, após o provisionamento ter sido realizado. Existe também a previsão do fornecimento do serviço no modelo de reserva

antecipada, em que o usuário pode reservar antecipadamente recursos por um período de tempo pré-estabelecido, pagando de forma adiantada um valor menor em comparação ao preço sob demanda [19].

3.4.1 Azure *pay-as-you-go*

As instâncias disponíveis para utilização *pay-as-you-go* (sob demanda) são mostradas na Tabela 3.6. Ao final do mês de setembro de 2017, havia 71 tipos de instâncias disponíveis para provimento neste modelo de precificação com sistema operacional Linux, localizadas na região *East US* do provedor.

Tabela 3.6: Instâncias Microsoft Azure no modelo *pay-as-you-go* [19].

| Instância | Núcleos | Memória | Armazenamento | GPU | Preço | Propósito |
|-----------|---------|---------|---------------|---------|---------|-------------------|
| B1S | 1 | 1 | 2 | | \$0,006 | General purpose |
| A0 | 1 | 0,75 | 20 | | \$0,018 | General purpose |
| A4 v2 | 4 | 8 | 40 | | \$0,159 | General purpose |
| D8 v3 | 8 | 32 | 64 | | \$0,40 | General purpose |
| D16 v3 | 16 | 64 | 128 | | \$0,80 | General purpose |
| F1 | 1 | 2 | 16 | | \$0,05 | Compute optimized |
| F2 | 2 | 4 | 32 | | \$0,099 | Compute optimized |
| F4 | 4 | 8 | 64 | | \$0,199 | Compute optimized |
| F8 | 8 | 16 | 128 | | \$0,398 | Compute optimized |
| F16 | 16 | 32 | 256 | | \$0,796 | Compute optimized |
| E2 v3 | 2 | 16 | 32 | | \$0,133 | Memory optimized |
| E16 v3 | 16 | 128 | 256 | | \$1,064 | Memory optimized |
| D13 v2 | 8 | 56 | 400 | | \$0,598 | Memory optimized |
| G2 | 4 | 56 | 768 | | \$0,981 | Memory optimized |
| G5 | 32 | 448 | 6.144 | | \$7,843 | Memory optimized |
| L4 | 4 | 32 | 678 | | \$0,312 | Storage optimized |
| L8 | 8 | 64 | 1.388 | | \$0,624 | Storage optimized |
| L16 | 16 | 128 | 2.807 | | \$1,248 | Storage optimized |
| L32 | 32 | 256 | 5.630 | | \$2,496 | Storage optimized |
| NC6 | 6 | 56 | 340 | 1X K80 | \$0,90 | GPU |
| NC24 | 24 | 224 | 1.440 | 4X K80 | \$3,60 | GPU |
| NC6 v2 | 6 | 112 | 336 | 1X P100 | \$0,90 | GPU |
| NC24 v2 | 24 | 448 | 1.344 | 4X P100 | \$3,60 | GPU |
| NV24 | 24 | 224 | 1.440 | 4X M60 | \$4,37 | GPU |
| H8 | 8 | 56 | 1.000 | | \$0,796 | High performance |
| H16 | 16 | 112 | 2.000 | | \$1,591 | High performance |
| H8m | 8 | 112 | 1.000 | | \$1,066 | High performance |
| H16mr | 16 | 224 | 2.000 | | \$2,345 | High performance |
| H16r | 16 | 112 | 2.000 | | \$1,75 | High performance |

Assim como nos provedores AWS e GCP, há uma grande variação de preços e configurações, o que torna a escolha do tipo de instância uma tarefa complexa para o usuário.

3.4.2 Azure *reserved instances*

O modelo de reserva antecipada (*reserved instances*) possibilita contratação de instâncias virtuais por 1 ou 3 anos. Em novembro de 2017, foi observado que o provedor passou a oferecer o serviço em fase experimental, proporcionando redução significativa dos custos

entre 36% e 52% em comparação ao modelo sob demanda (ou *pay-as-you-go*) [81]. O modelo é semelhante aos modelos *EC2 reserved* (Seção 3.2.2) e *GCP reserved* (Seção 3.3.3).

3.5 Comparação entre Provedores

Com o objetivo de comparar os aspectos relacionados à precificação de instâncias, são analisados os provedores AWS, GCP e Microsoft Azure. Primeiramente, é feita a comparação em relação ao modelo de precificação estática sob demanda, presente nesses três provedores de nuvem pública. Em seguida, é feita uma análise comparativa dos modelos de precificação de instâncias de cada provedor, com o objetivo de caracterizar as opções oferecidas e estabelecer o critério de especialização em relação às opções disponíveis.

3.5.1 Comparação dos Preços de Instâncias *on demand*

A análise comparativa dos preços praticados pelos provedores AWS, GCP e Microsoft Azure no modelo de precificação sob demanda (*on demand*) considera os custos médios de CPU virtual, de memória (em Gigabyte) e de instância nos três provedores. Para isso, foram computados os preços de instâncias na região Leste dos Estados Unidos (US East), utilizando o sistema operacional Linux. Assim, são calculados os custos médios por hora de utilização em dólar americano (USD/hora), considerando os seguintes propósitos: propósito geral (*general purpose*), uso intensivo de CPU (*CPU optimized*), uso intensivo de memória (*memory optimized*) e todos os tipos de instâncias disponibilizadas pelo provedor (*overall*).

Tabela 3.7: Comparação dos preços praticados pelos provedores AWS, GCP e Microsoft Azure no modelo de precificação *on demand* em 2017.

| Propósito | Provedor | Custo médio de vCPU (por hora) | Custo médio de Memória (GB por hora) | Custo médio de Instância (por hora) |
|-------------------|----------------|--------------------------------|--------------------------------------|-------------------------------------|
| General Purpose | AWS | \$0,0509132 | \$0,0133189 | \$0,5396800 |
| | GCP | \$0,0107000 | \$0,0028533 | \$0,1941286 |
| | Microsot Azure | \$0,0528452 | \$0,0137695 | \$0,4888182 |
| Compute Optimized | AWS | \$0,0490882 | \$0,0272490 | \$1,0014000 |
| | GCP | \$0,0080250 | \$0,0089167 | \$0,1685250 |
| | Microsot Azure | \$0,0433013 | \$0,0216507 | \$0,8263333 |
| Memory Optimized | AWS | \$0,0972024 | \$0,0072369 | \$1,6330000 |
| | GCP | \$0,0133750 | \$0,0020577 | \$0,2808750 |
| | Microsot Azure | \$0,0757539 | \$0,0105511 | \$0,9696500 |
| Overall | AWS | \$0,0948504 | \$0,0128979 | \$1,5365760 |
| | GCP | \$0,0536711 | \$0,0144929 | \$0,1936448 |
| | Microsot Azure | \$0,0741269 | \$0,0140303 | \$0,8995169 |

Pode-se observar pela Tabela 3.7 que o provedor GCP pratica os menores preços sob a ótica de custos médios por hora, tanto para utilização de CPU virtual, quanto para uso de Gigabyte de memória. Esse provedor também apresenta o menor preço médio de instância, analisando todos os tipos de instâncias disponíveis. De forma geral, pode-se perceber que o custo médio de uso de recursos como CPU e memória é sempre menor nas instâncias especializadas para estes tipos de recursos (*compute optimized* e *memory optimized*) do que nas instâncias de propósito geral, apesar desse último propósito apresentar tipos de instâncias com custos médios inferiores em relação aos demais propósitos.

Assim como na precificação de instâncias sob demanda, análise comparativa semelhante entre os provedores de nuvem pública pode ser realizada, tendo como base o modelo de precificação de reserva antecipada. Entretanto, vale ressaltar que o provedor AWS apresenta a maior diversidade de opções de tipos de instâncias pré-configuradas sob o ponto de vista de características de infraestrutura, que podem ser mais adequadas à necessidade específica da aplicação. Assim, sob o ponto de vista do usuário, trata-se de uma vantagem em relação aos demais provedores que disponibilizam tipos de instâncias nos modelos sob demanda ou de reserva antecipada.

3.5.2 Comparação dos Modelos de Precificação

Em relação aos critérios de classificação propostos na Figura 3.1, foram identificados os níveis de especialização dos provedores de nuvem com relação aos mecanismos de precificação oferecidos, conforme dados apresentados na Tabela 3.8.

Tabela 3.8: Comparação dos modelos de precificação dos provedores AWS, GCP e Microsoft Azure.

| Provedor | Variação de Preços | Alocação | Reserva Antecipada | Especialização da Precificação |
|-----------------|--------------------|-----------------------|--------------------|--------------------------------|
| AWS | Estática/Dinâmica | Permanente/Transiente | Sim | Alta |
| GCP | Estática | Permanente/Transiente | Sim | Média |
| Microsoft Azure | Estática | Permanente | Sim* | Baixa |

Na primeira coluna são apresentados os provedores de nuvem pública analisados (AWS, GCP e Microsoft Azure). Na segunda coluna é feita uma classificação com relação à possibilidade de variação de preços ser estática ou dinâmica. Entende-se que somente o provedor AWS possui variação dinâmica através do modelo *spot* do serviço EC2. Na coluna seguinte, é analisada a característica de alocação de instâncias por parte do usuário, sendo permanente ou transiente. Neste último caso, há possibilidade de revogação da instâncias do usuário por parte do provedor nos modelos de precificação *spot* da AWS e no modelo transiente do GCP. Em seguida, é apontada a existência ou não do modelo

de alocação de instâncias através de reserva antecipada, que pode gerar economia para os usuários através da reserva e antecipação de pagamento. Cabe ressaltar que provedor Microsoft Azure passou a fornecer esse modelo muito depois dos outros dois provedores, a partir de 2018. Por último, é feita a classificação de especialização do modelo geral de precificação dos provedores, sendo que o serviço Amazon EC2 do provedor AWS oferece o modelo mais especializado e amadurecido de precificação de instâncias para nuvens públicas, compreendendo as opções sob demanda, reserva antecipada e *spot*.

Capítulo 4

Séries Temporais, Teoria de Jogos e Leilões aplicados à Utilidade

A utilização da infraestrutura de computação em nuvem pode ser entendida como um mecanismo de mercado. De um lado, usuários contratam recursos com base em requisitos computacionais de suas aplicações e em razão de interesses econômicos próprios, como restrições de custo e tempo de utilização. Do outro lado, provedores de nuvem oferecem diversos tipos de recursos computacionais, de acordo com características de infraestrutura bastante específicas, assim como localização geográfica e modelos de precificação, estes últimos também atrelados a diversos aspectos econômicos.

Este capítulo tem como objetivo abordar os fundamentos teóricos utilizados em diversas análises de custos e disponibilidade de recursos em computação em nuvem, mais especificamente no modelo de infraestrutura como serviço. São detalhados, então, métodos que utilizam técnicas estatísticas e econômicas para possibilitar um melhor aproveitamento dos recursos computacionais.

Nesse contexto, é dado destaque especial ao conceito de função de utilidade, que define uma função a ser otimizada de forma que os benefícios esperados sejam efetivamente alcançados. Desse modo, são apresentadas técnicas muito usadas em computação em nuvem e que são relacionadas à modelagem e ao cálculo da função de utilidade.

A estrutura do capítulo é detalhada da seguinte forma. Inicialmente, os fundamentos sobre função de utilidade são definidos na Seção 4.1 e, nas seções seguintes, as técnicas que utilizam o conceito de função de utilidade são apresentadas. A Seção 4.2 aborda a definição de série temporal. Na Seção 4.3, a técnica de média móvel com janela deslizante aplicada em dados históricos é detalhada. Em seguida, na Seção 4.4, são abordados os fundamentos sobre teoria de jogos. No desfecho do capítulo, nas Seções 4.5 e 4.6, são abordados conceitos de leilões e desenho de mecanismos, respectivamente.

4.1 Função de Utilidade

O conceito primário de utilidade foi usado inicialmente para modelar riqueza ou valor, da perspectiva do cliente. Posteriormente, na economia neoclássica, o conceito de função de utilidade foi reformulado para medir preferências em relação ao conjunto de bens e serviços disponíveis [82]. Nesse caso, os consumidores podem revelar preferências individuais independentemente do ambiente social, definindo uma função de utilidade racional que cobre todas as possíveis opções.

A definição de Bernoulli [83] para função de utilidade, publicada em 1954 e ainda utilizada atualmente, é apresentada na Expressão 4.1. A utilidade $u : X \rightarrow \mathbb{R}$ representa uma relação de preferência \prec no conjunto X , ou seja, se o valor da utilidade de x_i é menor do que de x_j , então x_j tem preferência sobre x_i .

$$\forall x_i, x_j \in X, u(x_i) < u(x_j) \implies x_i \prec x_j \quad (4.1)$$

Em [84] e [4], a função de utilidade é definida como uma função que mede a preferência relativa de um investidor por diferentes níveis de riqueza total. Uma função de utilidade $U(w)$ é uma função de riqueza w duas vezes diferenciável, definida para $w > 0$, que tem as propriedades de não saciedade (a primeira derivada $U'(w) > 0$) e aversão ao risco (a segunda derivada $U''(w) < 0$) [84].

A propriedade de não saciedade afirma que a utilidade aumenta com a riqueza, mas que o investidor nunca fica saciado, ou seja, ele sempre busca obter mais riqueza [4]. A propriedade de aversão ao risco afirma que a função de utilidade é côncava ou, em outras palavras, que a utilidade marginal da riqueza diminui à medida que a riqueza aumenta. Para ilustrar essa última propriedade, considere um exemplo em que a utilidade extra marginal é obtida pela aquisição de um dólar adicional. Para o investidor que tem apenas um dólar no início, obter mais um dólar é muito importante. Para aquele que já tem uma grande quantia, obter mais um dólar é quase irrelevante. Desse modo, o aumento da utilidade causada pela aquisição de um dólar adicional diminui à medida que a riqueza aumenta [4].

O princípio da maximização da utilidade esperada afirma que um investidor racional, quando confrontado com uma escolha entre um conjunto de alternativas de investimento viáveis concorrentes, age para selecionar um investimento que maximize sua utilidade esperada em função da riqueza. Expresso de maneira mais formal, para cada investimento I em um conjunto de alternativas de investimento viáveis concorrentes F , seja $X(I)$ a variável aleatória que dá o valor final do investimento para o período de tempo em questão. Então, um investidor racional com função de utilidade U enfrenta o problema de otimização de encontrar um investimento ótimo $I_{opt} \in F$ para o qual [4]:

$$E(U(X(I_{opt}))) = \max_{I \in F} E(U(X(I))) \quad (4.2)$$

Um exemplo emblemático de aplicação da função de utilidade e das suas propriedades de não saciedade e aversão ao risco é denominado “jogo justo” [4]. Considere um investidor com uma função de utilidade de ordem raiz quadrada, conforme Equação 4.3.

$$U(w) = \sqrt{w} = w^{0,5} \quad (4.3)$$

As derivadas de primeira e segunda ordem são definidas nas Equações 4.4 e 4.5, respectivamente.

$$U'(w) = 0,5w^{-0,5} > 0 \quad (4.4)$$

$$U''(w) = -0,25w^{-1,5} < 0 \quad (4.5)$$

Assim sendo, presume-se que o patrimônio atual do investidor seja de \$5 e que, para fins de simplificação, há apenas um investimento disponível. Nesse investimento, é lançada uma moeda para definição de um resultado aleatório. Se der cara (50% de chance), o investidor ganha (recebe) \$4, aumentando seu patrimônio para \$9. Se der coroa (50% de chance), o investidor perde (deve pagar) \$4, diminuindo seu patrimônio para \$1. Desse modo, o retorno esperado é de $0,5 \times (\$4) + 0,5 \times (-\$4) = \$0$, por isso é denominado de “jogo justo”.

Na prática, este jogo se assemelha mais a uma aposta do que a um investimento. De forma semelhante, considera-se um investimento que custe \$5 (todo o patrimônio atual do investidor) e que tenha dois valores futuros possíveis: \$1 no caso ruim e \$9 no caso bom. Dessa maneira, assume-se que o investidor tem apenas duas opções (o conjunto F de alternativas de investimento viáveis tem apenas dois elementos). O investidor pode jogar ou não fazer nada.

Se o investidor se recusar a jogar e ficar com seus \$5, ele ficará com os mesmos \$5, para uma utilidade esperada de $\sqrt{5} = 2,24$. Se ele jogar, o resultado esperado é o mesmo \$5, mas a utilidade esperada do resultado é apenas $0,5 \times 1 + 0,5 \times 3 = 2$. Uma vez que o investidor age para maximizar a utilidade esperada e sendo 2,24 maior do que 2, ele recusa-se a jogar o jogo. Em geral, um investidor avesso ao risco sempre se recusará a jogar um jogo justo, no qual o retorno esperado é de 0%. Se o retorno esperado for maior do que 0%, o investidor pode ou não optar pelo jogo, dependendo de sua função de utilidade e patrimônio inicial [4]. O exemplo é ilustrado na Figura 4.1.

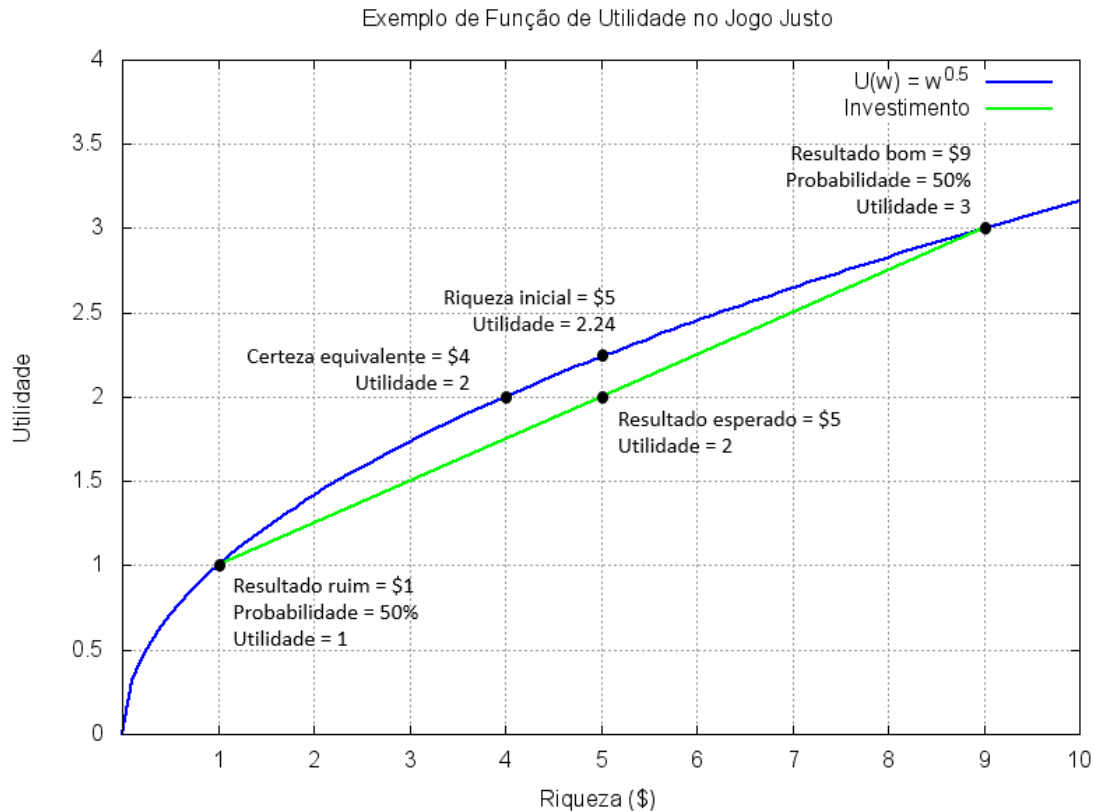


Figura 4.1: Exemplo da função de utilidade no jogo justo [4].

Se a probabilidade do resultado bom no exemplo fosse de 75% ao invés de 50%, o resultado esperado seria igual a \$7, o ganho esperado seria igual a \$2, o retorno esperado seria de 40% e a utilidade esperada seria 2,5. Como 2,5 é maior do que 2,24, o investidor estaria disposto a fazer o investimento, considerando o retorno esperado de 40% como uma compensação por assumir o risco do investimento [4].

4.2 Séries Temporais

Série temporal é um conjunto de observações x_t , cada uma registrada em um tempo t específico [5]. De modo geral, uma série temporal é composta por uma coleção de pontos listados em ordem natural de tempo, a partir da qual inferências podem ser extraídas. Para que a análise da série temporal seja feita é necessário o estabelecimento de um modelo hipotético de probabilidade que represente os dados. A partir desse modelo, é possível então estimar parâmetros, verificar a sua qualidade e ajustá-lo aos dados. O modelo ajustado possibilita o aprimoramento da compreensão em relação ao mecanismo que gera a série, podendo ser usado de várias maneiras, dependendo do campo de aplicação

específico, como para a descrição compacta dos dados, para a identificação de tendência momentânea ou sazonal e, principalmente, para a previsão de valores futuros.

Um modelo de série temporal para os dados observados x_t é uma especificação das distribuições conjuntas (ou possivelmente apenas das médias e covariâncias) de uma sequência de variáveis aleatórias X_t , das quais x_t é uma realização [5]. De maneira formal, um modelo de série temporal probabilístico completo para a sequência de variáveis aleatórias X_1, X_2, \dots especifica todas as distribuições conjuntas dos vetores aleatórios $(X_1, \dots, X_n)'$, $n = 1, 2, \dots$, ou equivalente às probabilidades descritas na Expressão 4.6.

$$P[X_1 \leq x_1, \dots, X_n \leq x_n], -\infty < x_1, \dots, x_n < \infty, n = 1, 2, \dots \quad (4.6)$$

A especificação 4.6 dificilmente é usada de forma isolada na análise de séries temporais, uma vez que, em geral, conterà muitos parâmetros a serem estimados a partir dos dados disponíveis. Em vez disso, são especificados apenas os momentos de primeira e segunda ordem das distribuições conjuntas, ou seja, os valores esperados EX_t e os produtos esperados $EX_t^2 = E(X_{t+h}X_t)$, $t = 1, 2, \dots, h = 0, 1, 2, \dots$, respectivamente [5].

Um exemplo de série temporal consiste na variação de preços de instâncias Amazon EC2 no modelo de precificação *spot*. Se for observada o período compreendendo os meses de janeiro a junho de 2020, ou seja, durante o primeiro semestre do ano 2020, é possível observar a ocorrência de valores de preços que oscilam no decorrer do tempo. Na Figura 4.2, estas variações podem ser observadas na região US-East para os diversos tipos de instâncias, sendo que cada uma delas pode ser considerada para análise como uma série temporal independente.

Uma abordagem geral para a análise de séries temporais consiste na observação visual dos dados plotados para a identificação de tendências, componentes sazonais, quaisquer mudanças bruscas de comportamento e observações remotas (ou *outliers*). Na interpretação econômica, é importante reconhecer a presença de componentes sazonais e removê-los. Este ajuste visa obter o processo estacionário, que possui propriedades estatísticas que não se alteram ao longo do tempo como média e variância.

Uma série temporal $X_t, t = 0, \pm 1, \dots$ é considerada estritamente estacionária se tiver propriedades estatísticas semelhantes às da série “deslocada no tempo” $X_{t+h}, t = 0, \pm 1, \dots$, para todo inteiro h [5]. Restringindo a atenção às propriedades que dependem apenas dos momentos de primeira e segunda ordem de X_t , a ideia se torna mais precisa com as definições de média μ (Equação 4.7) e covariância γ (Equação 4.8), assumindo que X_t é uma série temporal e $EX_t^2 < \infty$.

$$\mu_X(t) = E(X_t), \forall t \in \mathbb{Z} \quad (4.7)$$

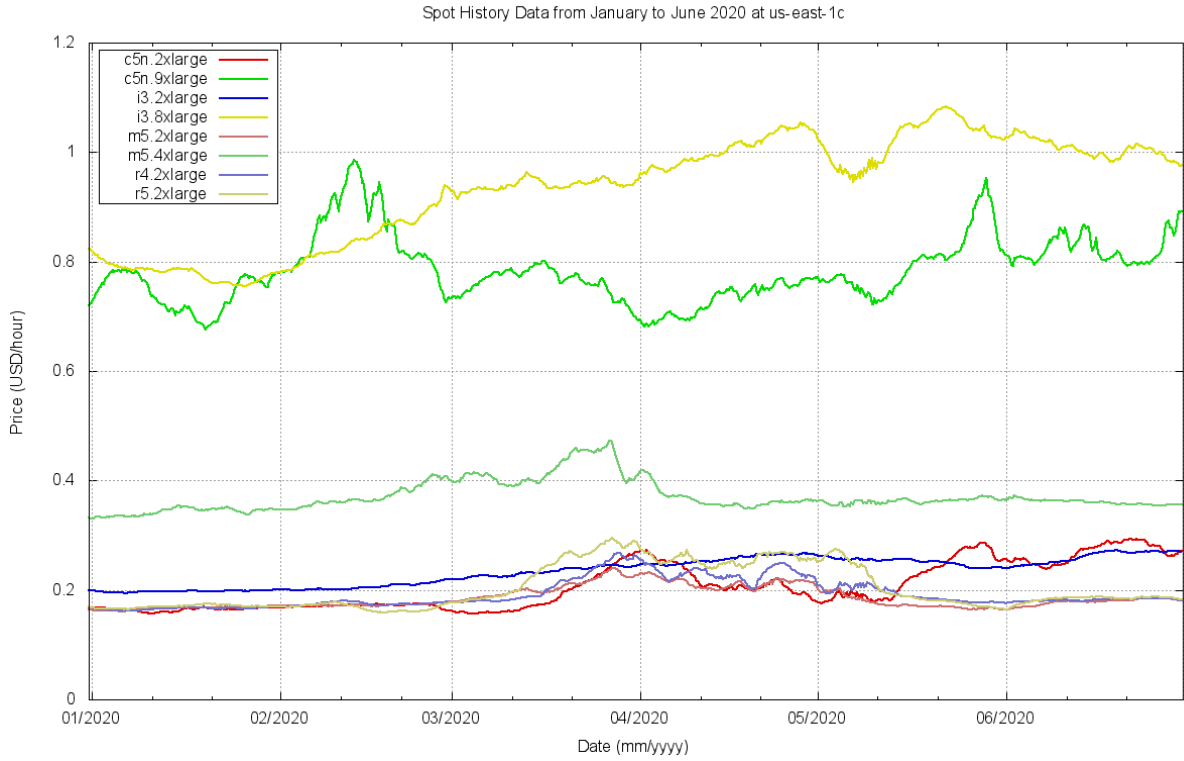


Figura 4.2: Série temporal com a variação de preços de instâncias *spot* da Amazon EC2 durante o primeiro semestre de 2020 na região US-East.

$$\gamma_X(r, s) = Cov(X_r, X_s) = E[(X_r - \mu_X(r))(X_s - \mu_X(s))], \forall r, s \in \mathbb{Z} \quad (4.8)$$

Desse modo, X_t é uma série temporal estacionária se $\mu_X(t)$ é independente de t e $\gamma_X(t+h, t)$ é independente de t para qualquer valor de h , sendo que $t, h \in \mathbb{Z}$. Existem tipos de séries estacionárias bastante conhecidas, como ruído branco ou *white noise* [5], definida por uma sequência de variáveis aleatórias X_t não correlacionadas, cada uma com média zero e variância σ^2 , indicada pela notação $X_t \sim WN(0, \sigma^2)$. A Figura 4.3 apresenta um exemplo desse tipo de série.

Infelizmente, a maioria das séries temporais econômicas não são estacionárias e, mesmo após o ajuste, normalmente exibem um comportamento não estacionário [85]. Isso requer que o uso da técnica de série temporal, por exemplo, para analisar dados históricos de preços *spot* da Amazon, seja combinado com outras técnicas, como médias móveis com janelas deslizantes. Uma variação dessa técnica, a *AutoRegressive Integrated Moving Average* (ARIMA), se baseia na transformação dos dados para obtenção de uma nova série estacionária e, posteriormente, a utilização de médias móveis [5].

Nesse contexto, também são frequentemente aplicadas técnicas de modelagem econômica, como cálculo de utilidade (Seção 4.1) e teoria de jogos (Seção 4.4).

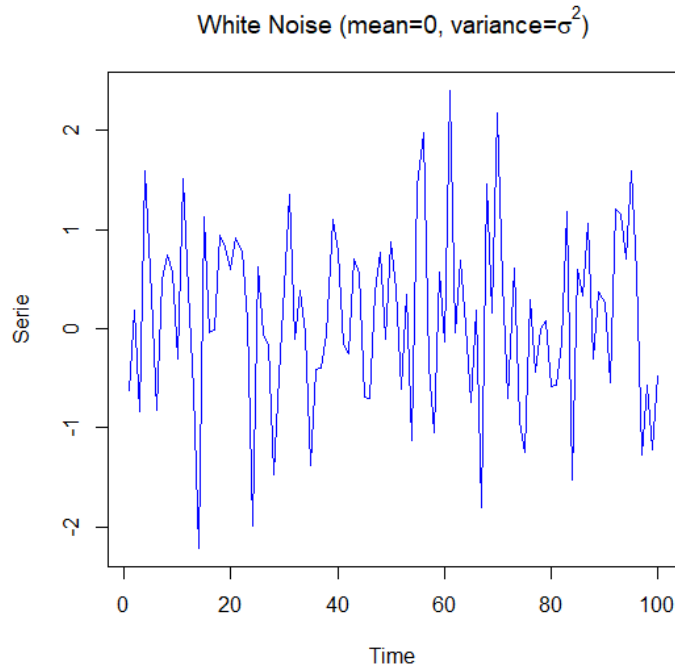


Figura 4.3: Série temporal do tipo ruído branco ou *white noise* [5].

4.3 Média Móvel com Janela Deslizante

Para as séries estacionárias discutidas na Seção 4.2, a técnica de média móvel de primeira ordem considera uma série temporal definida pela Equação 4.9.

$$X_t = Z_t + \theta Z_{t-1}, t = 0, \pm 1, \dots \quad (4.9)$$

Onde $Z_t \sim WN(0, \sigma^2)$, ou seja, é similar a uma série *white noise*, sendo que θ é uma constante com valor real [5]. Da Equação 4.9, assume-se que $EX_t^2 < \infty$ e $EX_t = \mu = 0$, sendo essa última premissa difícil de ser encontrada em séries econômicas, conforme mencionado na Seção 4.2.

Por outro lado, é possível considerar um cenário mais simples a partir de uma série temporal não estacionária, composta por uma única variável com valores discretos. O cálculo da média móvel simples com janela deslizante, considerando um conjunto finito de valores pré-existentes, é um método básico de decomposição de séries temporais usado para eliminar flutuações, processo também conhecido como “suavização” [6]. Esse método pode ser usado para obtenção de uma estimativa de previsão, com base em um número fixo de valores anteriores, determinado pelo tamanho da janela deslizante no tempo.

A Figura 4.4 mostra uma série temporal com valores $x_i \in X$, sendo i um número inteiro no intervalo $1 \leq i \leq n$, e n o tamanho do conjunto X . Nesse caso, cada janela w

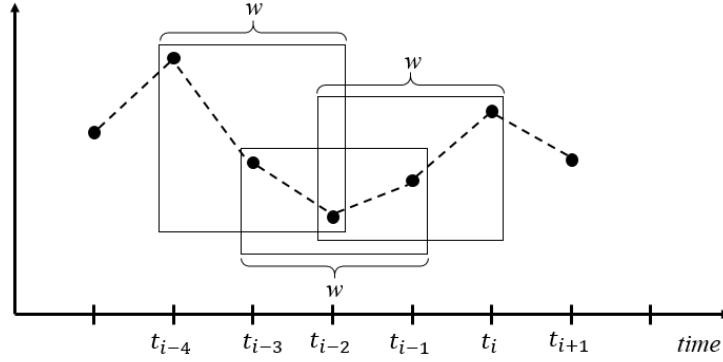


Figura 4.4: Média móvel com janela deslizante para uma série temporal discreta [6].

possui 3 valores no tempo. A Equação 4.10 define a média móvel simples \bar{x} da amostra, calculada com base nos elementos anteriores contidos na janela deslizante de largura fixa w , sendo $w > 0$ [6].

$$\bar{x}_i = \frac{1}{w} \sum_{j=i-w}^i x_j \quad (4.10)$$

Nesse cenário, o valor da média móvel simples \bar{x}_i pode ser usado no momento i como uma estimativa para o valor de x , de forma suavizada em relação a grandes variações ocorridas no período contido na janela deslizante w . Desse modo, pode-se estimar o valor de x em um determinado momento através de seus valores anteriores, considerando que $x_{i-w}, \dots, x_{i-2}, x_{i-1}, x_i$ são os valores reais da série temporal no intervalo $[i-w, i]$.

Além disso, é possível também calcular a variância e o desvio padrão da amostra na janela w , conforme observado nas Equações 4.11 e 4.12, respectivamente.

$$s_i^2 = \frac{1}{w-1} \sum_{j=i-w}^i (x_j - \bar{x}_j)^2 \quad (4.11)$$

$$s_i = \sqrt{s_i^2} \quad (4.12)$$

Assumindo uma distribuição normal de valores em X , o intervalo de confiança de 95% (ou somente 95%CI) para a estimativa do valor no momento i é dado pela Equação 4.13 [86]. Para o intervalo 95%CI, considera-se o quantil da distribuição Z, ou seja, $z_{(1-\alpha/2)}$ de uma distribuição normal, sendo que α é o nível de significância de 5% [86].

$$95\%CI_i = [\bar{x}_i - z_{(1-\alpha/2)} \times s_i; \bar{x}_i + z_{(1-\alpha/2)} \times s_i] \quad (4.13)$$

Da interpretação estatística, é possível concluir com 95% de confiança que a estimativa do valor de x , calculada a partir do momento i , será um valor entre os intervalos inferior

e superior calculados na Equação 4.13. Assim, tanto a estimativa da média móvel quanto o seu intervalo de confiança podem ser usados em momentos futuros, como no momento $i + 1$, com o propósito de previsão [86].

Da comparação de todos os valores previstos com os valores reais obtidos, é possível medir o erro residual e avaliar a precisão do modelo. Uma forma de se fazer isso é medir o erro quadrático médio ou *mean squared error* (MSE), conforme Equação 4.14.

$$MSE = \frac{1}{n} \sum_{k=1}^n (x_k - \hat{x}_k)^2 \quad (4.14)$$

Nessa Equação 4.14, \hat{x}_k é o valor previsto no momento k , sendo que o valor de MSE considera cada momento $1 \leq k \leq n$ em que tenha sido calculado o valor previsto \hat{x}_k e possa ser coletado o valor realizado x_k . No caso do uso da técnica de média móvel com janela deslizante para o cálculo de valor previsto, considera-se \hat{x}_k igual ao valor da média móvel \bar{x}_k , calculado para cada momento k , ou de um de seus intervalos de confiança (inferior ou superior), na Equação 4.14.

4.4 Teoria de Jogos

A Teoria dos Jogos é uma área da Economia que aborda situações em que o resultado da decisão de um jogador depende não apenas de como ele escolhe entre as várias opções disponíveis, mas também das escolhas feitas pelos jogadores com os quais está interagindo. Neste contexto, um jogo é definido da seguinte forma [87]: primeiro, os jogadores escolhem ações; em seguida os jogadores recebem recompensas ou penalidades que dependem da combinação das ações que acabaram de escolher. Dessa forma, um jogo é necessariamente composto por um ou mais jogadores (também referenciados como agentes), uma regra compreendida por todos os jogadores (conhecimento comum) e um resultado final que gera utilidades (em valores positivos, negativos ou nulos) para os jogadores.

No contexto de um jogo, assume-se que os jogadores são racionais e inteligentes [88]. As ações de um jogador racional são consistentes com o objetivo desejado, ou seja, maximizar a sua utilidade. Já o conceito de inteligência se refere ao comportamento de jogadores distintos que, ao terem o mesmo conhecimento sobre o jogo, conseguem realizar as mesmas inferências sobre a situação em que se encontram, não agindo de forma aleatória. A seguir, são descritos os principais tipos de jogos de acordo com [87] e [89].

Jogos estáticos/dinâmicos: o tipo de jogo mais simples é classificado como estático, em que todos os jogadores tomam as decisões e observam seus resultados simultaneamente. Os jogos dinâmicos, também conhecidos como sequenciais, são aqueles

em que um jogador toma sua decisão após observar a decisão de outros jogadores, sendo que estes podem vir a tomar novas decisões ao longo do jogo [87].

Jogos com informação completa/incompleta: quanto à informação disponível, os jogos podem ser classificados como de informação completa, ou seja, jogos em que a função de utilidade de cada jogador (ou função que determina o retorno para o jogador da combinação de ações escolhidas por todos os jogadores) é de conhecimento comum entre os demais jogadores. Finalmente, os jogos de informação incompleta são aqueles em que um jogador não tem conhecimento sobre a função de utilidade de outro jogador [87].

Existem diversas formas de representação de jogos, sendo cada uma mais apropriada ao tipo de jogo que está sendo modelado. A forma normal ou estratégica é adequada para representar um jogo estático com informação completa. Nesse caso, o jogo é descrito como uma função que associa cada perfil de decisões dos jogadores (sendo uma decisão para cada jogador) a um perfil de resultados (um resultado para cada jogador, quando as decisões descritas pelo perfil de decisões são tomadas simultaneamente por todos) [87]. Desse modo, um jogo na forma normal ou estratégica é definido por [89] e [87]:

$$J = (N; (S_i)_{i \in N}; (u_i)_{i \in N}) \quad (4.15)$$

Na Equação 4.15, para cada $i \in N$, $u_i : \prod_{j \in N} S_j \rightarrow R$ descreve a consequência para o jogador i das estratégias escolhidas por todos os jogadores. Se cada jogador $j \in N$ escolher a estratégia $s_j \in S_j$ e $s = (s_j)_{j \in N}$, então i tem por utilidade $u_i(s) \in R$. A escolha $s = (s_j)_{j \in N}$ é um perfil de estratégias. O valor de $u_i(s)$ é chamado de consequência, utilidade, recompensa ou *payoff* do jogo para o jogador i .

Na forma normal, os jogadores escolhem suas estratégias simultaneamente, sem que um jogador observe as estratégias dos outros. Quando N é um conjunto finito, é comum a utilização da notação $N = \{1, 2, \dots, n\}$, $J = (n; S_1, \dots, S_n; u_1, \dots, u_n)$ [89].

Um exemplo clássico de representação de jogos na forma normal é o dilema dos prisioneiros [87]. Nesse exemplo, dois suspeitos são presos e acusados de um crime. A polícia não possui provas suficientes para condená-los, a menos que um deles confesse. A polícia então prende os suspeitos em celas separadas e explica as consequências que resultarão das ações que irão tomar. Se não confessarem, ambos serão condenados por uma infração menor com detenção de um mês de prisão. Se os dois confessarem, ambos serão condenados à prisão por seis meses. Finalmente, se um suspeito confessa mas o outro não, então aquele que confessar será liberado imediatamente, mas o outro será condenado a nove meses de prisão (seis pelo crime e mais três por obstruir a justiça) [87].

Neste caso, trata-se de um jogo com $n = 2$ jogadores, cada um com duas estratégias possíveis, ou seja, confessar ou não. Assim, o conjunto de estratégias para o jogador 1 é $S_1 = \{C, N\}$ e para o jogador 2 é $S_2 = \{c, n\}$. As consequências para os jogadores são as seguintes [89]:

$$\begin{aligned} u_1(C, n) = 0, u_1(C, c) = -6, u_1(N, n) = -1, u_1(N, c) = -9 \\ u_2(C, n) = -9, u_2(C, c) = -6, u_2(N, n) = -1, u_2(N, c) = 0 \end{aligned} \quad (4.16)$$

O dilema dos prisioneiros também pode ser descrito na forma normal através de uma matriz que relaciona as possíveis decisões e seus respectivos *payoffs*, conforme apresentado na Figura 4.5.

| | | Jogador 2 | |
|-----------|----------|-----------|----------|
| | | <i>c</i> | <i>n</i> |
| Jogador 1 | <i>C</i> | -6, -6 | 0, -9 |
| | <i>N</i> | -9, 0 | -1, -1 |

Figura 4.5: Representação normal ou estratégica do dilema dos prisioneiros através de uma matriz de decisões.

No dilema dos prisioneiros, se um suspeito confessar, então, o outro prefere confessar e ficar na prisão por seis meses, a ficar calado e ter que permanecer na prisão por nove meses. Da mesma forma, se um suspeito ficar calado, então, o outro prefere confessar e ser liberado imediatamente a ficar calado e ter que permanecer na prisão por um mês. Assim, para o prisioneiro i , a estratégia de ficar calado é dominada pela estratégia de confessar. Para qualquer estratégia que o prisioneiro j escolher, a recompensa para o prisioneiro i ficar calado é menor do que a recompensa em confessar [87].

Na definição de equilíbrio de Nash argumenta-se que, se a teoria dos jogos fornece uma solução única para um problema teórico de jogos, então a solução deve ser um equilíbrio de Nash [87]. Supõe-se que a teoria dos jogos faça uma previsão única sobre a estratégia que cada jogador escolherá. Para que esta previsão seja correta, é necessário que cada jogador esteja disposto a escolher a estratégia prevista pela teoria. Assim, a estratégia prevista de cada jogador deve ser a melhor solução do jogador para as estratégias previstas dos outros jogadores. Tal predisposição pode ser chamada estrategicamente estável, pois nenhum jogador quer desviar-se da estratégia prevista. Denomina-se essa previsão de um equilíbrio de Nash [87].

No jogo na forma normal $J = (n; S_1, \dots, S_n; u_1, \dots, u_n)$, as estratégias (s_1^*, \dots, s_n^*) são um equilíbrio de Nash se, para cada jogador i , s_i^* é a melhor resposta para as estratégias escolhidas pelos outros $n - 1$ jogadores $(s_1^*, \dots, s_{i-1}^*, s_{i+1}^*, \dots, s_n^*)$. Assim, tem-se que:

$$u_i(s_1^*, \dots, s_{i-1}^*, s_i^*, s_{i+1}^*, \dots, s_n^*) > u_i(s_1^*, \dots, s_{i-1}^*, s_i, s_{i+1}^*, \dots, s_n^*) \quad (4.17)$$

Para toda estratégia possível $s_i \in S_i$, tal que s_i é a solução para:

$$\max_{s_i \in S_i} u_i(s_1^*, \dots, s_{i-1}^*, s_i, s_{i+1}^*, \dots, s_n^*) \quad (4.18)$$

Um par de estratégias satisfaz a condição do equilíbrio de Nash se a estratégia de cada jogador for a melhor resposta para a estratégia escolhida pelo outro jogador [87]. Uma abordagem de força bruta para encontrar esse equilíbrio em um jogo é simplesmente verificar se cada combinação possível de estratégias satisfaz a Equação 4.17. No caso de dois jogadores, essa abordagem começa da seguinte forma: para cada jogador e para cada estratégia viável para esse jogador, determine a melhor resposta do outro jogador para essa estratégia. Se a estratégia e a melhor resposta do outro jogador para a estratégia se localizarem na mesma célula da matriz utilizada na representação normal ou estratégica, trata-se de um equilíbrio de Nash. Assim, (C, c) é o único par estratégico que satisfaz a Equação 4.17 no Dilema dos Prisioneiros.

4.5 Leilões

O mecanismo descrito por um leilão é a situação em que um determinado objeto é colocado à venda para um grupo de participantes candidatos à compra. Normalmente, o vendedor desconhece os valores atribuídos ao bem pelos possíveis compradores. Entretanto, pode haver alguma informação sobre a distribuição de valores possíveis entre os compradores como, por exemplo, valores mínimo e máximo que os compradores estão dispostos a pagar. Nessa configuração, há alguns tipos de leilão bastante conhecidos e que podem ser usados de forma que o vendedor consiga negociar melhor o objeto em questão. Cada tipo de leilão influencia no valor de receita obtida por parte do vendedor e de dispêndio financeiro por parte do comprador que adquire o objeto [90].

Um exemplo tradicional é o leilão selado, em que cada jogador conhece o seu lance, mas não conhece o lance dos demais. Nesse leilão, os lances são revelados simultaneamente e o vencedor do leilão é conhecido. Trata-se de um jogo de informação incompleta (Seção 4.4), em que cada jogador não conhece a estratégia dos demais jogadores [90].

A venda de um objeto único por parte de um vendedor, com o objetivo de atingir o maior valor possível, pode ser classificada de quatro formas diferentes utilizando leilões, conforme descrito a seguir [91]:

Leilão selado de primeiro preço: cada possível comprador envia um lance selado ao vendedor. O caráter selado se dá pelo fato de somente o candidato à compra e o

vendedor saberem o valor do lance. Nesse tipo de leilão, o maior lance ganha e o candidato vencedor paga sua oferta pelo bem.

Leilão selado de segundo preço (ou leilão de Vickrey): semelhante ao anterior, em que cada candidato à compra submete um lance selado ao vendedor. O maior lance ganha e o candidato vencedor paga o segundo maior lance pelo bem.

Leilão holandês: o vendedor começa com um preço muito alto e, em seguida, passa a reduzi-lo. O primeiro candidato a manifestar interesse ganha e adquire o objeto ao preço anunciado.

Leilão inglês: o vendedor começa com um preço muito baixo (talvez zero) e passa a aumentá-lo. Cada candidato sinaliza quando ele deseja abandonar o leilão. Se um candidato abandonou o leilão, ele não pode continuar a oferecer lances mais tarde. Quando apenas um candidato permanece, ele é o vencedor e paga o preço atual.

Assim sendo, modelagem considera que existe um objeto a ser vendido e que o vencedor não sabe o valor que seus potenciais compradores estão dispostos a pagar por esse objeto. O vendedor gostaria de encontrar algum procedimento de leilão que lhe desse a maior receita entre todos os diferentes tipos de leilões. Para análise do problema, a teoria dos jogos (Seção 4.4) estuda os leilões como jogos não cooperativos e com informações incompletas [92].

Nesse contexto, assume-se que há um vendedor que tem um único objeto para ser vendido. Ele enfrenta n potenciais compradores, numerados $1, 2, \dots, n$. Os índices i e j são utilizados para representar dois candidatos entre os possíveis compradores. Supondo um vendedor neutro ao risco, ou seja, que está disposto a vender o objeto mesmo que por valor inferior a um limite mínimo pré-estabelecido e, assumindo que existam n participantes candidatos a compra, os valores de v_i e v_j indicam os valores atribuídos ao objeto a ser leiloado pelos compradores candidatos i e j . Estes valores são de conhecimento privado por parte de cada candidato à compra e são independentes entre si, condição denominada de Modelo de Valoração Privada e Independente [91]. Assim, uma vez que um comprador emprega sua própria informação privada para avaliar o valor do objeto, esta avaliação não é afetada se ele posteriormente obtiver informações privadas de outros compradores, ou seja, a informação privada de cada comprador é suficiente para determinar o valor atribuído ao objeto que está sendo leiloado.

4.6 Desenho de Mecanismos

O Desenho de Mecanismos é um campo teórico da Economia que adota uma abordagem de engenharia para a concepção de mecanismos econômicos ou de incentivos, em função de objetivos desejados pelo projetista [93]. Os objetivos podem ser, por exemplo, a maximização de lucros ou a alocação eficiente de recursos, no contexto de ambientes estratégicos, nos quais os jogadores atuam de forma racional. O Desenho de Mecanismos possui amplas aplicações nas áreas de Economia e Política, como modelagem de mercados, leilões, procedimentos de votação, assim como na Ciência da Computação, como roteamento de tráfego em rede e indexação de conteúdo para busca patrocinada (*sponsored search auction*) [94]. Essa teoria é especialmente relevante quando o projetista do mecanismo requer informações privadas, conhecidas apenas pelos demais participantes, de forma a alcançar seus objetivos. A eficiência na concepção de um mecanismo reside na garantia de que seja dado, a quem possui a informação desejada, o incentivo adequado para revelá-la [91].

A fundamentação teórica de Desenho de Mecanismos se baseia em Teoria dos Jogos (Seção 4.4) e Leilões (Seção 4.5). Os participantes do jogo são definidos como agentes, em razão de sua racionalidade e existência de interesses próprios [95]. Dado um contexto composto por agentes de diferentes tipos, um problema de Desenho de Mecanismos é definido por dois componentes: a especificação estruturada de saída do jogo, produzida a partir de um algoritmo pré-estabelecido, e as descrições do que os agentes participantes desejam, definidas como funções de utilidade sobre o conjunto de saídas possíveis [95].

O mecanismo Vickrey-Clarke-Groves (VCG) [96] se aplica à situação em que se busca maximizar os valores aferidos ao bem pelos agentes participantes. O leilão selado de segundo preço (Seção 4.5) é um exemplo de mecanismo VCG genérico para a negociação de um único item, conforme apresentado na Figura 4.6. É considerado o conceito de bem-estar como a soma dos lances dos agentes vencedores. Nesse mecanismo, o agente deve considerar o bem-estar social, também conhecido como benefício ou prejuízo causado aos outros participantes (ou *social cost*) em razão da sua participação no jogo [7].

$$p_i = \boxed{\text{Bem-estar ótimo (para os outros jogadores) se o jogador } i \text{ não estivesse participando}} - \boxed{\text{Bem-estar dos outros jogadores a partir do resultado do mecanismo}}$$

No leilão selado de 2º preço de um único item:

$$p_i = \boxed{2^\circ \text{ maior valor}} - \boxed{0}$$

(quando o jogador i não está jogando, o bem-estar é o 2º maior valor) (se o jogador i ganha, o bem-estar do outro jogador é igual a zero)

Figura 4.6: Mecanismo VCG genérico, adaptado de [7].

Capítulo 5

Redes Neurais Artificiais

Neste capítulo são caracterizados os principais aspectos sobre redes neurais e sua aplicação como técnica de *machine learning* para aprendizado supervisionado. Inicialmente, são apresentados os conceitos gerais sobre arquitetura, retropropagação de erros e utilização do algoritmo de gradiente descendente para otimização dos parâmetros da rede neural.

No decorrer do capítulo também são apresentadas e discutidas as redes neurais recorrentes ou *Recurrent Neural Networks* (RNN), utilizadas para representação de conhecimento, assim como as redes *Long Short Term Memory* (LSTM), que são uma evolução das RNNs para geração e armazenamento eficiente de conhecimento de curto e longo prazo.

A estrutura do capítulo é composta da seguinte forma. A utilização de Redes Neurais como técnica de *Machine Learning* é discutida na Seção 5.1. Os algoritmos de otimização baseados em gradiente descendente e aplicados em redes neurais são discutidos na Seção 5.2. Finalmente, na Seção 5.3, a arquitetura das RNNs e as redes LSTM são detalhadas.

5.1 Redes Neurais e *Machine Learning*

5.1.1 *Perceptron*

A rede neural artificial é um conceito análogo às redes neurais biológicas, em que sistemas são capazes de reconhecer padrões e relacionamentos subjacentes em um conjunto de dados, por meio de um processo que simula a maneira como o cérebro humano opera. O conceito original de rede neural vem da definição de *perceptron* [8] ou estrutura análoga ao neurônio. Tal estrutura possibilitou a primeira e mais simples implementação de rede neural, utilizada para classificar dados em duas categorias [97].

Na Figura 5.1 é apresentada a estrutura do *perceptron* como unidade de processamento individual da rede neural. A unidade recebe n valores de entrada (x_1, x_2, \dots, x_n) e são definidos n coeficientes ou pesos (w_1, w_2, \dots, w_n) , que definem a medida de importância

de cada valor de entrada. Como na sinapse do neurônio biológico, que possibilita a transmissão de impulsos nervosos, existe uma função de ativação que conecta os valores de entrada e produz um único valor de saída y , respeitando um limite arbitrário (ou *threshold*) pré-estabelecido [8].

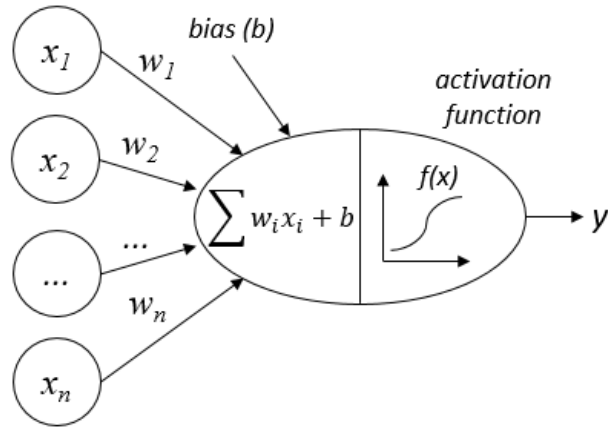


Figura 5.1: *Perceptron* ou unidade de processamento individual da rede neural [8].

O processamento se inicia com cada valor de entrada sendo multiplicado por seu respectivo peso ($w_i x_i$). O termo b refere-se ao viés (*bias*), que determina a parte invariável do cálculo. Uma vez obtida a soma ponderada, a função de ativação é aplicada. Um exemplo simples dessa função é mostrado na Equação 5.1, em que o neurônio gera um de dois valores possíveis, determinados pelo fato de que o resultado da soma ponderada é menor ou maior do que o limite θ .

$$y = \begin{cases} 0, & \text{se } \sum_{i=1}^n w_i x_i + b \leq \theta \\ 1, & \text{se } \sum_{i=1}^n w_i x_i + b > \theta \end{cases} \quad (5.1)$$

5.1.2 Arquitetura da Rede Neural

A arquitetura da rede neural artificial é organizada em camadas de unidades de processamento ou neurônios [98]. Na camada de entrada, os neurônios recebem os dados necessários para explicar o problema ou fenômeno a ser analisado. Os dados de entrada da rede são também chamados de *features* e, normalmente, compreendem um subconjunto relevante ao domínio do problema. A camada oculta é uma camada intermediária que usa funções de ativação em cada unidade de processamento para modelar relações não lineares entre os domínios de entrada e saída [99]. O emprego do termo “camada oculta” vem do fato de não haver contato direto com dados externos. As saídas de cada camada são as

entradas da camada seguinte. A camada de saída é a última camada da rede. Ela produz o resultado, que é uma informação “rotulada” (ou *label*), para problemas de classificação, ou um valor numérico, no caso de problemas de previsão. Essa arquitetura baseada em fluxo direto de dados através de camadas é também conhecida como rede de alimentação direta ou *feed-forward network* [9].

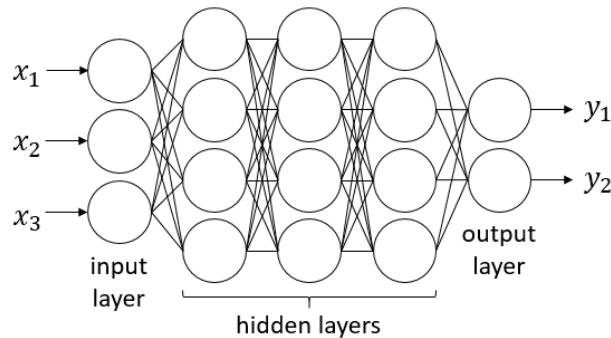


Figura 5.2: Arquitetura de rede neural em camadas ou *multilayer perceptron* [9].

A arquitetura de rede neural em camadas é mostrada na Figura 5.2. A estrutura consiste em uma camada de entrada, uma camada de saída e três camadas ocultas. Teoricamente, adicionar um número suficiente de neurônios às camadas ocultas pode ser suficiente para aproximar os valores de saída, matematicamente, a qualquer função não linear [9]. A camada de saída pode ser precedida por uma camada densa, que simula uma função de ativação *softmax* para expressar probabilidades dos rótulos de classificação. Essa arquitetura multicamadas foi amplamente utilizada em redes neurais convolucionais, que são redes de inspiração biológica usadas em problemas de visão computacional, como por exemplo para classificação de imagens e detecção de objetos [100].

O conceito de rede neural foi posteriormente aplicado à aprendizagem supervisionada, cujos sistemas “aprendem” a realizar tarefas analisando exemplos, sem serem explicitamente programados com regras específicas para a realização das tarefas [101]. Desse cenário emerge o termo aprendizado de máquina ou *machine learning* [9], que se refere à capacidade de generalizar modelos de aprendizagem a partir de dados vistos para exemplos não vistos.

O processo de retropropagação dos erros atualiza os pesos das conexões da rede neural, de acordo com uma taxa de aprendizado pré-definida [102]. Esse processo ocorre durante a etapa de treinamento da rede e será discutido na Seção 5.1.3.

5.1.3 Retropropagação dos Erros

Em redes neurais artificiais, o treinamento possibilita a atualização dos pesos em cada uma das unidades de processamento através do cálculo de uma função do erro ou perda (*loss function*). Para isso, o valor de treinamento esperado é comparado com o valor de saída produzido pela rede. É comum a utilização de uma função de erro como MSE (Seção 4.3), que pode possuir diversos mínimos locais e um único mínimo global, conforme apresentado na Figura 5.3.

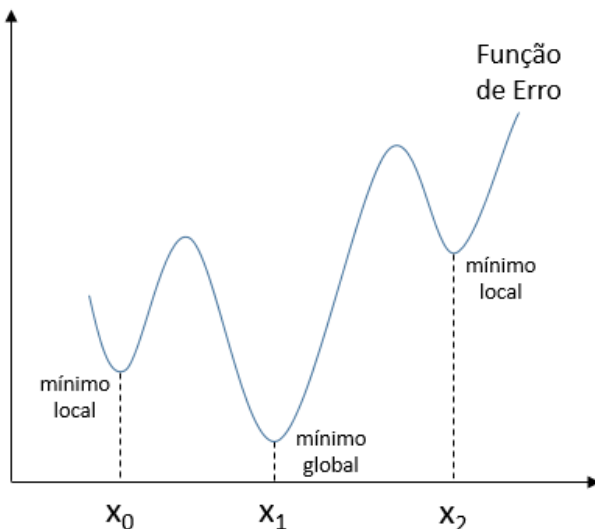


Figura 5.3: Função de erro e seus mínimos locais e global.

O problema é que o processo de minimização da função de erro durante o treinamento pode ficar restrito a valores próximos a mínimos locais, em que são obtidos resultados sub-ótimos. Nesse cenário, o treinamento de redes neurais utilizando o cálculo de gradiente descendente (ou *gradient descent*) de uma função de erro é uma técnica bastante conhecida [103]. A partir do cálculo do gradiente, é possível ajustar os pesos de forma que a rede produza saídas apropriadas em relação aos valores esperados e, conseqüentemente, calcular o erro através de uma função previamente definida.

O treinamento de redes neurais é realizado através da utilização do algoritmo de retropropagação dos erros, que contém duas fases principais conhecidas como fases de avanço e retrocesso [9].

Fase de avanço: Nesta fase, as entradas das unidades de processamento são alimentadas com valores de treinamento na rede neural. Isso resulta em uma cascata direta de cálculos através das camadas, que utilizam os valores atuais dos pesos e suas respectivas funções de ativação. A saída prevista pode ser então comparada com a da rede e a derivada da função de perda em relação à saída é calculada.

Fase de retrocesso: O objetivo principal desta fase consiste em aprender o gradiente da função de perda em relação aos diferentes pesos, usando a regra da cadeia do cálculo diferencial. Primeiramente, a derivada da função de perda é calculada em relação aos pesos em todas as unidades das camadas anteriores. Em seguida, os gradientes são usados para atualizar os pesos na direção reversa, começando pelo nó de saída, passando pelos nós intermediários, até os nós da camada de entrada.

Em um caminho simples, no qual a função $g(\cdot)$ precede $f(\cdot)$, a função de composição é definida por $f(g(\cdot))$. Desse modo, considerando um grafo computacional trivial com dois nós, conforme mostrado na Figura 5.4, a regra da cadeia do cálculo diferencial para obtenção do gradiente é detalhada na Equação 5.2 [9].

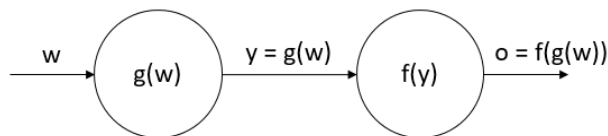


Figura 5.4: Ilustração da cadeia do cálculo diferencial em um grafo simples [9].

$$\begin{aligned} \frac{\partial o}{\partial w} &= \frac{\partial f(g(w))}{\partial w} \\ &= \frac{\partial f(g(w))}{\partial g(w)} \cdot \frac{\partial g(w)}{\partial w} \end{aligned} \quad (5.2)$$

De forma genérica, considera-se uma sequência de unidades ocultas h_1, h_2, \dots, h_k seguida pela saída o , em relação à qual a função de perda L é calculada. Além disso, assume-se que o peso da conexão da unidade oculta h_r para h_{r+1} é $w_{(h_r, h_{r+1})}$. Então, no caso de existir um único caminho de h_1 para o , pode-se derivar o gradiente da função de perda em relação a qualquer um desses pesos usando a regra da cadeia expressa na Equação 5.3 [9].

$$\frac{\partial L}{\partial w_{(h_{r-1}, h_r)}} = \frac{\partial L}{\partial o} \cdot \left[\frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right] \frac{\partial h_r}{\partial w_{(h_{r-1}, h_r)}} \quad \forall r \in 1 \dots k \quad (5.3)$$

A Equação 5.3 assume que apenas um único caminho de h_1 a o existe na rede, enquanto um número exponencial de caminhos pode existir na realidade. Uma variante generalizada da regra da cadeia multivariável calcula o gradiente em um grafo computacional, onde mais de um caminho pode existir. Isso é obtido adicionando a composição ao longo de cada um dos caminhos de h_1 a o . Portanto, generaliza-se a expressão para o caso em que existe um conjunto P de caminhos de h_r a o .

$$\frac{\partial L}{\partial w_{(h_{r-1}, h_r)}} = \frac{\partial L}{\partial o} \cdot \left[\sum_{[h_r, h_{r+1}, \dots, h_k, o] \in P} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right] \frac{\partial h_r}{\partial w_{(h_{r-1}, h_r)}} \forall r \in 1 \dots k \quad (5.4)$$

Após o cálculo do gradiente, segundo a Equação 5.4, o novo peso $w'_{(h_{r-1}, h_r)}$ da unidade oculta h_r para h_{r+1} pode ser atualizado. Para isso, aplica-se a Equação 5.5, considerando uma taxa de aprendizado α previamente estabelecida [9]. Os pesos das conexões entre as demais unidades ocultas da rede são calculados e atualizados, de forma que um novo ciclo de treinamento possa ser realizado.

$$w'_{(h_{r-1}, h_r)} = w_{(h_{r-1}, h_r)} - \alpha \frac{\partial L}{\partial w_{(h_{r-1}, h_r)}} \quad (5.5)$$

5.1.4 Algoritmo Gradiente Descendente

O algoritmo do gradiente descendente é muito utilizado para o aprendizado de parâmetros, que no caso das redes neurais, definem os pesos das conexões entre as unidades de processamento [9]. Desse modo, o processo recursivo de atualização de pesos dos nós é executado até a convergência, aplicando-se repetidamente os dados de treinamento em épocas. De forma generalizada, a convergência ocorre quando é encontrado um valor mínimo global da função de perda L , em relação a todos valores dos pesos ajustados $w' \in P$, conforme mostrado na Relação 5.6.

$$\left(w'_{(h_{r-1}, h_r)} = w_{(h_{r-1}, h_r)} - \alpha \frac{\partial L}{\partial w_{(h_{r-1}, h_r)}} \right) \rightarrow \min_{\forall w' \in P} L \quad (5.6)$$

Reescrevendo a derivada parcial utilizando a notação vetorial, no caso de uma rede neural com d conexões, os pesos correspondem ao vetor $\bar{W} = (w_1, w_2, \dots, w_d)$. Portanto, na descida gradiente, deve-se realizar o cálculo descrito na Expressão 5.7 para minimização da função de perda L .

$$\bar{W} \Leftarrow \bar{W} + \alpha \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2} \dots \frac{\partial L}{\partial w_d} \right) \quad (5.7)$$

$$\bar{W} \Leftarrow \bar{W} + \alpha \left(\frac{\partial L}{\partial \bar{W}} \right) \quad (5.8)$$

Ao considerar todo o vetor \bar{W} na notação vetorial, tem-se como resultado a Expressão 5.8. Para redes de uma única camada, o gradiente descendente é calculado apenas em relação a \bar{W} , enquanto que para redes multicamadas, todos os parâmetros dos pesos da

rede precisam ser atualizados aplicando-se a regra cadeia do cálculo diferencial (vide Seção 5.1.3), durante o processo de retropropagação [9].

O Algoritmo 1 define o cálculo do gradiente descendente genérico aplicado em redes neurais (adaptado de [9] e [104]). Os parâmetros iniciais do algoritmo são: a taxa de aprendizado α , o vetor de valores de entrada \bar{X} e o vetor de valores de saída esperados \bar{Y} . O algoritmo assume que a função de perda L seja previamente definida. O algoritmo também assume um limite máximo de iterações ou épocas, representado pela constante $epochs$, de forma a limitar o processo de otimização. A saída do algoritmo é justamente o vetor \bar{W} que minimiza a função de perda L .

Algorithm 1 *GenericGradientDescent*(α, \bar{X}, \bar{Y})

```

1: for each  $w_i \in \bar{W}$  do
2:    $w_i \leftarrow rand[0, 1]$ 
3: end for
4:  $e \leftarrow 1$ 
5: repeat
6:   Calculates  $L(f(\bar{X}, \bar{W}), \bar{Y})$  in terms of:
        $\bar{X}$  as input
        $f(\bar{X}, \bar{W})$  as output
        $\bar{Y}$  as expected output
        $\bar{W}$  as weight parameters
7:    $\bar{W} \leftarrow \bar{W} + \alpha \left( \frac{\partial L}{\partial \bar{W}} \right)$ 
8:    $e \leftarrow e + 1$ 
9: until  $L$  converges or  $e > epochs$ 
10: return  $\bar{W}$ 

```

Nos passos de 1 a 3 do Algoritmo 1, o vetor de parâmetros \bar{W} correspondente aos pesos das conexões é inicializado com valores randômicos no intervalo entre 0 e 1. No passo 4, o contador de épocas e é inicializado com 1. Esse contador é importante para restringir a execução do algoritmo no caso da função de perda L não convergir. Na linha 6 é calculada a função de perda L em relação aos valores de entrada da rede (vetor \bar{X}), valores de saída produzidos com base nas funções de ativação da rede ($f(\bar{X}, \bar{W})$) e valores de saída esperados (vetor \bar{Y}). O cálculo e a atualização simultânea dos novos parâmetros com vistas à minimização da função de perda L são realizados na linha 7. Os novos parâmetros são calculados em função da taxa de aprendizado e do gradiente descendente, sendo que este último é calculado utilizando a regra da cadeia multivariável, conforme detalhado na Equação 5.5. O contador de iterações de treinamento ou número de épocas decorridas é atualizado na linha 8. Na linha 9 são avaliadas as condições de convergência da função de perda L ou de alcance do limite máximo de iterações ($e > epochs$). Finalmente, os valores otimizados dos parâmetros \bar{W} são retornados na linha 10.

5.1.5 Variações do Gradiente Descendente

O problema de aprendizado de máquina de forma geral pode ser reformulado como um problema de otimização sobre uma função objetivo específica [104]. No caso das redes neurais, a otimização é realizada através do cálculo do gradiente descendente e a função objetivo é definida como uma função de perda L , que muitas vezes é a soma linear das funções de perda nos pontos de dados de treinamento individuais. Desse modo, a função de perda L de uma rede neural pode ser escrita, conforme apresentado na Equação 5.9, considerando L_i a contribuição de perda para cada i -ésimo ponto de treinamento [9].

$$L = \sum_{i=1}^n L_i \quad (5.9)$$

Batch gradient descent

Uma vez que a função de perda da maioria dos problemas de otimização pode ser expressa como uma soma linear das perdas em relação aos pontos individuais (Equação 5.9), a derivada parcial da função de perda pode ser decomposta em na soma das derivadas parciais das funções de perda em cada i -ésimo ponto de treinamento, como mostrado na Equação 5.10.

$$\frac{\partial L}{\partial \bar{W}} = \sum_{i=1}^n \frac{\partial L_i}{\partial \bar{W}} \quad (5.10)$$

Nesse caso, todos os pontos de treinamento são considerados em uma única etapa do treinamento. A atualização do gradiente total em relação a todos os pontos soma os efeitos individuais específicos de cada ponto. A consequência pode ser um alto nível de redundância no conhecimento capturado por diferentes pontos de treinamento. Essa abordagem é conhecida como *batch gradient descent* [9], sendo que apenas uma etapa de cálculo do gradiente e atualização dos parâmetros é realizada por época. O Algoritmo 2 apresenta esse comportamento, mostrando a mudança na linha 7, em relação ao Algoritmo 1, e a atualização da época na linha 8.

Algorithm 2 *BatchGradientDescent*(α, \bar{X}, \bar{Y})

```
6:  ...
7:   $\bar{W} \leftarrow \bar{W} + \alpha \left( \sum_{i=1}^n \frac{\partial L_i}{\partial \bar{W}} \right)$ 
8:   $e \leftarrow e + 1$ 
9:  ...
```

Stochastic gradient descent

Um processo de aprendizado mais eficiente, cujas atualizações se baseiam em valores de treinamento específicos, é mostrado na Expressão 5.11. Essa segunda abordagem de cálculo do gradiente descendente é conhecida como estocástica, pois percorre os pontos de treinamento em ordem aleatória. Cada gradiente local pode ser calculado com eficiência e, em seguida, ser utilizado para atualização dos parâmetros. Isso torna a descida rápida, embora às custas de precisão no cálculo do gradiente [9].

$$\bar{W} \leftarrow \bar{W} + \alpha \frac{\partial L_i}{\partial \bar{W}} \quad (5.11)$$

Uma propriedade interessante do gradiente descendente estocástico é que, embora não tenha um desempenho tão bom nos dados de treinamento em comparação com a abordagem *batch*, pode ter um desempenho comparável e às vezes até melhor nos dados de teste [105]. O Algoritmo 3 detalha a mudança proposta pela versão estocástica do cálculo do gradiente descendente.

Algorithm 3 *StochasticGradientDescent*(α, \bar{X}, \bar{Y})

```
6:   ...
7:    $i \leftarrow \text{rand}[1, n]$ 
8:    $\bar{W} \leftarrow \bar{W} + \alpha \frac{\partial L_i}{\partial \bar{W}}$ 
9:    $e \leftarrow e + 1$ 
10:  ...
```

Mini-batch gradient descent

A abordagem *mini-batch* do cálculo do gradiente descendente utiliza um lote $B = \{j_1, \dots, j_m\}$ de pontos de treinamento para a atualização, sendo que cada ponto de treinamento é composto por valores de entrada $x \in X$, valores esperados de saída $y \in Y$ e pesos das conexões $w \in W$ das unidades que ligam as entradas às saídas no cálculo de L_i , de forma que $j_i = ((x_a \dots x_b), (w_k \dots w_l), (y_p \dots y_q))$, sendo $a < b$, $k < l$ e $p < q$, conforme apresentado na Expressão 5.12.

$$\bar{W} \leftarrow \bar{W} + \alpha \sum_{i \in B} \frac{\partial L_i}{\partial \bar{W}} \quad (5.12)$$

A descida do gradiente geralmente nessa abordagem oferece a melhor compensação entre os requisitos de precisão, velocidade e consumo de memória [9]. As saídas de uma camada são matrizes ao invés de vetores, e a propagação direta requer a multiplicação da matriz de peso pela matriz de ativação. O mesmo é verdadeiro para a retro-propagação, em que matrizes de gradientes são mantidas. Portanto, a implementação aumenta os requisitos de memória, o que é um fator limitante no tamanho do lote. Várias etapas de

cálculo do gradiente são realizadas por época nessa abordagem, dependendo da quantidade de lotes que agrupam pontos de treinamento. O Algoritmo 4 mostra a abordagem *mini-batch* do cálculo do gradiente descendente, considerando que o valor de m , referente ao tamanho dos lotes, é informado como parâmetro adicional do algoritmo.

Algorithm 4 *MinibatchGradientDescent*($\alpha, \bar{X}, \bar{Y}, m$)

```

6:   ...
7:   for each batch of size  $m$  do
8:      $B \leftarrow j_1, \dots, j_m$ 
9:      $\bar{W} \leftarrow \bar{W} + \alpha \left( \sum_{i \in B} \frac{\partial L_i}{\partial \bar{W}} \right)$ 
7:   end for
11:   $e \leftarrow e + 1$ 
12:  ...

```

5.1.6 Entendimento Geométrico do Gradiente Descendente

O valor da taxa de aprendizado influencia na eficiência e no desempenho da execução do algoritmo do gradiente descendente. Conforme definido na Expressão 5.6, o cálculo do gradiente é realizado até a convergência, ou seja, até o ponto ótimo em relação à função de perda, nesse caso, o mínimo global. Assumindo um cenário simples de cálculo do gradiente de uma variável w em relação a uma função de perda convexa L (com um mínimo global), podemos analisar a taxa de aprendizado α sob dois aspectos: quando o valor da taxa é muito pequeno ou quando é muito grande [9].

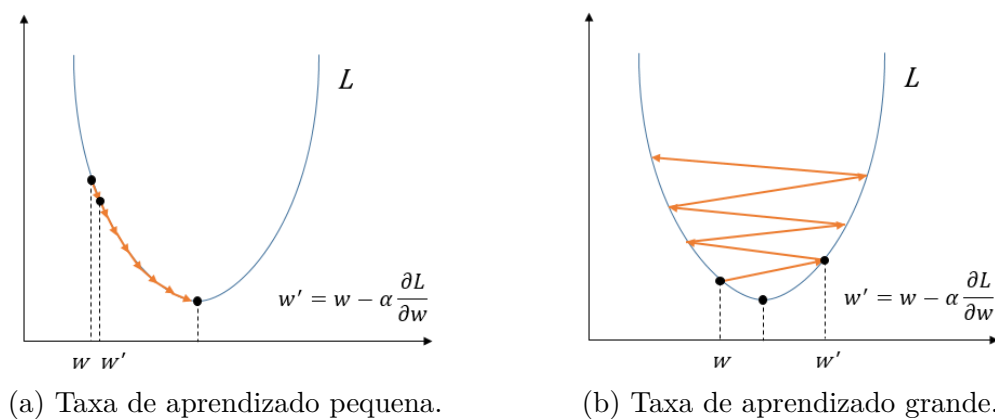


Figura 5.5: Convergência do cálculo do gradiente descendente [11].

No primeiro caso, em que a taxa de aprendizado é muito pequena, o algoritmo pode se tornar lento, uma vez que a atualização do valor de w se dá em pequenos passos para correção de direção em busca do valor mínimo da função de perda L [11]. Isso ocorre porque o termo que expressa a multiplicação da taxa de aprendizado pelo gradiente, ou seja, o valor de $\alpha \frac{\partial L}{\partial w}$ é muito pequeno numericamente em relação a w , conforme ilustrado

na Figura 5.5a. Entretanto, se a taxa de aprendizado for muito grande, como descrito no segundo caso, corre-se o risco do cálculo do gradiente descendente ultrapassar o valor mínimo da função de perda L . Isso pode fazer com que o algoritmo falhe em convergir [11], uma vez que o termo $\alpha \frac{\partial L}{\partial w}$ é numericamente muito grande em relação a w , conforme exemplo ilustrado na Figura 5.5b.

5.1.7 Inicialização e Parâmetros de Treinamento da Rede

As redes neurais tendem a ter maiores variações de parâmetros em comparação com outros algoritmos de *machine learning*, o que aumenta a importância da inicialização adequada da rede e do pré-processamento dos dados [9]. Os parâmetros relacionados à inicialização e ao comportamento da rede são também conhecidos como hiperparâmetros e são diferentes dos parâmetros que definem os pesos iniciais das conexões da rede.

O pré-processamento dos dados de entrada normalmente é realizado através de um procedimento de normalização ou *feature normalization*, no qual os dados são ajustados com valores no intervalo $[0, 1]$ [9]. Sejam min_j e max_j os valores mínimo e máximo do j -ésimo atributo. Então, cada valor x_{ij} referente ao i -ésimo elemento da *feature* de entrada j é normalizado (normalização Min-Max), conforme a Expressão 5.13.

$$x_{ij} \leftarrow \frac{x_{ij} - min_j}{max_j - min_j} \quad (5.13)$$

A normalização dos valores de entrada geralmente garante melhor desempenho no treinamento da rede, principalmente quando os valores variam em mais de uma ordem de magnitude [9]. O mesmo ajuste de escala deve ser realizado nos dados de entrada e saída de teste, que são usados em problemas de previsão ou classificação, no momento após treinamento.

A Tabela 5.1 resume os hiperparâmetros mais comuns utilizados para a configuração de redes neurais. O primeiro parâmetro se refere ao número de *features*, que define a quantidade de nós na camada de entrada da rede. De forma análoga, o segundo parâmetro define número de *labels* e, conseqüentemente, a quantidade de nós na camada de saída da rede. Para uma sequência de treinamento de tamanho $(n + k)$, a rede neural compreende que n valores de entrada causam influência em k valores de saída. Portanto, as sequências utilizadas para treinamento e teste da rede precisam estar nesse formato, o que requer pré-processamento adicional dos dados.

O terceiro parâmetro define o valor da taxa de aprendizado usada para o cálculo do gradiente, durante o processo de retropropagação dos erros, conforme foi visto na Seção 5.1.3. Ela controla como o modelo se adapta, cada vez que os pesos das conexões são atualizados. O quarto parâmetro especifica o algoritmo de otimização utilizado no

Tabela 5.1: Hiperparâmetros mais comuns em redes neurais.

| # | Hiperparâmetro | Descrição |
|---|---------------------------------------|--|
| 1 | Quantidade de <i>features</i> | Determina a quantidade de nós na camada de entrada da rede e o formato das sequências utilizadas para treinamento. |
| 2 | Quantidade de <i>labels</i> | Determina a quantidade de nós na camada de saída da rede e também influencia no formato das sequências de treinamento. |
| 3 | Taxa de aprendizado | Determina a taxa de aprendizado utilizada para o cálculo do gradiente, durante a retropropagação dos erros. |
| 4 | Algoritmo de otimização | Determina o algoritmo de otimização utilizado no cálculo do gradiente, durante a retropropagação dos erros. |
| 5 | Quantidade de camadas ocultas | Determina a quantidade de camadas ocultas ou intermediárias na rede. |
| 6 | Quantidade de nós nas camadas ocultas | Determina a quantidade de nós em cada camada de oculta da rede. |
| 7 | Quantidade de épocas de treinamento | Determina a quantidade de épocas na etapa de treinamento da rede. |
| 8 | Tamanho do lote | Determina o tamanho do lote no caso do uso da abordagem <i>mini-batch</i> do gradiente descendente. |

cálculo do gradiente, conforme será visto na Seção 5.2. Os parâmetros 3 e 4 impactam diretamente no comportamento do modelo e na precisão dos resultados [15].

A quantidade de camadas ocultas da rede e a quantidade de nós em cada camada oculta são determinadas pelo quinto e sexto parâmetros. Esses dois parâmetros determinam o volume de conhecimento armazenado pela rede durante o processo de treinamento. Além desses dois parâmetros, a definição da arquitetura da rede deve especificar quais os tipos de nós intermediários serão utilizados, como nós recorrentes simples para construção de RNNs ou nós mais elaborados, como os utilizados em redes LSTM.

O sétimo parâmetro determina a quantidade de épocas ou ciclos de aprendizagem, nos quais a rede neural processa todos os dados de treinamento. Poucas épocas podem ser insuficientes para reter as informações relevantes, ao passo que muitas épocas podem causar ajuste excessivo dos pesos e perda de precisão no processo de previsão ou classificação de dados inéditos na etapa de teste.

O último parâmetro especifica o tamanho do lote, caso seja utilizada a abordagem *mini-batch* do algoritmo gradiente descendente. O tamanho do lote influenciará na eficiência e consumo de memória, conforme visto na Seção 5.1.5. Geralmente, são usadas potências de 2 (32, 64, 128 ou 256) [9].

É importante notar que não há relação direta entre o aumento do número de camadas ocultas e a melhoria da qualidade do aprendizado [100]. Outros parâmetros podem influenciar consideravelmente nos resultados, como o tamanho da sequência de entrada, o algoritmo de otimização e a presença de uma camada densa [106]. A escolha da melhor combinação de parâmetros é importante, pois impacta na eficiência da rede [107].

5.2 Algoritmos de Otimização em Redes Neurais

É difícil capturar dependências de longo prazo, pois os gradientes tendem a valores muito pequenos (desaparecer) ou muito grandes (explodir), problema conhecido como *vanishing/exploding gradients* [15]. Esse problema é inerente à otimização multivariável, mesmo em casos onde não há ótimos (no caso, mínimos da função de perda) locais. Na prática, isso acontece porque gradientes das camadas mais profundas da rede são calculados como produtos de muitos gradientes de camadas anteriores, que por sua vez são calculados a partir de derivadas das funções de ativação, conforme visto na Seção 5.1.3.

O método mais comum para aprendizado de parâmetros em redes neurais é o método de cálculo do gradiente descendente [9], visto nas Seções 5.1.3 e 5.1.4. Entretanto, uma direção de descida íngreme pode se reverter em subida, após uma pequena atualização nos parâmetros, conforme visto na Seção 5.1.6. Como resultado, muitas correções de curso são necessárias. Os problemas de oscilação e zigue-zague, descritos na Seção 5.1.6, também são bastante presentes sempre que a direção de descida mais íngreme se move ao longo de uma direção de alta curvatura na função de perda [9].

A seguir, serão discutidas estratégias de aprendizagem que funcionam bem em ambientes mal condicionados. As equações apresentadas são originárias da referência [9] e dos trabalhos originais de cada abordagem. Os métodos apresentados nas Seções 5.2.1, 5.2.2 e 5.2.3 objetivam aumentar a consistência na direção dos gradientes, de forma a acelerar as atualizações dos parâmetros em busca da convergência da função de perda. Tal objetivo pode ser alcançado com a utilização de otimizações específicas para os diferentes parâmetros da rede [9]. Esse é o caso dos algoritmos com taxas de aprendizado adaptativas, como AdaGrad, RMSProp, ADAM e NADAM, que serão vistos nas Seções 5.2.4, 5.2.5, 5.2.6 e 5.2.7, respectivamente.

5.2.1 Taxa de Aprendizado com Decaimento Dinâmico

A utilização de uma taxa de aprendizado constante representa uma limitação no algoritmo do gradiente descendente [9]. Uma taxa de aprendizado mais baixa usada no início fará com que o algoritmo demore muito para chegar perto de uma solução ótima. Por outro lado, uma taxa grande no final pode provocar oscilação em torno do ponto mínimo da função de perda por um longo tempo, ou divergir de forma instável. Permitir que a taxa de aprendizado diminua com o tempo pode evitar esses comportamentos indesejados.

As duas funções de diminuição da taxa de aprendizado mais comuns são decaimento exponencial e decaimento inverso [9], conforme Equações 5.14 e 5.15, respectivamente. A taxa de aprendizado α_e , utilizada na Equação 5.5 e no Algoritmo 1, é expressa em termos

de uma taxa inicial α_0 e da época e . O parâmetro k controla a taxa de decaimento e possui valor no intervalo $[0, 1]$.

$$\alpha_e = \alpha_0^{-ke} \quad (\text{decaimento exponencial}) \quad (5.14)$$

$$\alpha_e = \frac{\alpha_0}{1 + ke} \quad (\text{decaimento inverso}) \quad (5.15)$$

5.2.2 Momento Clássico

As técnicas de aprendizagem baseadas em *momentum* ou momento clássico [108] reconhecem o comportamento em zigue-zague. Exemplo desse cenário é ilustrado na Figura 5.5b. A simples tentativa de aumentar o tamanho do degrau de descida para obter um maior movimento na direção correta pode, na verdade, afastar a solução atual ainda mais da solução ótima. Nesse aspecto, faz mais sentido mover-se em uma direção média das últimas etapas, de forma que o zigue-zague seja suavizado [9].

Para o entendimento da técnica baseada em momento clássico, consideramos uma configuração de otimização do vetor de parâmetros \bar{W} através do uso do algoritmo tradicional do gradiente descendente. As atualizações dos parâmetros em relação à função de perda L , de forma bastante semelhante à definição apresentada na Expressão 5.8, são mostradas nas Expressões 5.16 e 5.17, para o cálculo tradicional e utilizando a técnica baseada em momento clássico, respectivamente.

$$\bar{V} \Leftarrow \alpha \frac{\partial L(\bar{W})}{\partial \bar{W}} ; \quad \bar{W} \Leftarrow \bar{W} + \bar{V} \quad (\text{técnica tradicional}) \quad (5.16)$$

$$\bar{V} \Leftarrow \beta \bar{V} - \alpha \frac{\partial L(\bar{W})}{\partial \bar{W}} ; \quad \bar{W} \Leftarrow \bar{W} + \bar{V} \quad (\text{momento clássico}) \quad (5.17)$$

Nessa técnica, o vetor \bar{V} é modificado considerando valores de “velocidade” previamente calculados, sendo que $\beta \in [0, 1]$ é um parâmetro de suavização [9]. Esse parâmetro também é conhecido como parâmetro de momento ou parâmetro de “atrito”, pois valores significativos de β atuam como “freios” na descida (convergência da função L) [9]. Desse modo, o aprendizado pode ser acelerado, uma vez que as oscilações laterais desnecessárias são minimizadas. A ideia básica é dar maior preferência a direções que têm maior importância na descida do gradiente.

Valores maiores de β ajudam a obter uma velocidade consistente \bar{V} na descida do gradiente em direção ao mínimo global da função de perda L . O valor de β próximo a zero possibilita uma descida direta do gradiente, entretanto, não garante a direção adequada [9].

5.2.3 Momento Nesterov

O momento Nesterov ou *Nesterov momentum* [109] é uma modificação do método baseado em momento clássico, em que os gradientes são calculados em um ponto que seria alcançado após a execução da porção de momento da etapa atual. Este ponto é obtido multiplicando-se o vetor de atualização anterior \bar{V} pelo parâmetro de atrito β e, em seguir, calculando-se o gradiente em $\bar{W} + \beta\bar{V}$. A ideia é que o valor do gradiente corrigido use um melhor entendimento sobre como os gradientes mudarão, por causa da porção de momento da atualização, e incorpore essa informação na porção de gradiente da atualização. Desse modo, o método usa uma certa quantidade de antecipação ao calcular as atualizações dos parâmetros [9].

Seja $L(\bar{W})$ a função de perda na solução atual \bar{W} . A atualização em razão do cálculo do gradiente é calculada conforme mostrado na Expressão 5.18.

$$\bar{V} \leftarrow \beta\bar{V} - \alpha \frac{\partial L(\bar{W} + \beta\bar{V})}{\partial \bar{W}} ; \quad \bar{W} \leftarrow \bar{W} + \bar{V} \quad (5.18)$$

A diferença em relação ao método clássico (Seção 5.2.2) é em termos do momento considerado para o cálculo do gradiente. Usar o valor do gradiente que seria calculado na próxima iteração, ou seja, no momento mais adiante, pode levar a uma convergência mais rápida [9].

O método de Nesterov funciona apenas na versão *mini-batch* do algoritmo gradiente descendente, visto na Seção 5.1.5, com tamanhos de lote pequenos. Nesses casos, pode-se mostrar que o método de Nesterov reduz o erro para $O(1/e^2)$ após e épocas, em comparação com um erro de $O(1/e)$ no método do momento clássico [109].

5.2.4 AdaGrad

O algoritmo AdaGrad ou *Adaptive Gradient* [110] tem por objetivo otimizar de forma independente o gradiente de cada parâmetro da rede. Nesse algoritmo, o valor agregado dos quadrados das derivadas parciais em relação a cada parâmetro é mantido ao longo da sua execução. A raiz quadrada do valor agregado é proporcional à inclinação média para cada parâmetro (embora o valor absoluto aumente com o número de épocas devido a agregações sucessivas). Esse método utiliza taxas de aprendizado específicas, o que implica na necessidade de memória adicional para manutenção dos valores agregados de cada parâmetro. A Expressão 5.19 mostra o cálculo do valor agregado para cada parâmetro w_i da rede neural.

$$A_i \leftarrow A_i + \left(\frac{\partial L}{\partial w_i} \right)^2 ; \quad \forall w_i \in \bar{W} \quad (5.19)$$

A atualização do *i*-ésimo parâmetro w_i é mostrada na Expressão 5.20.

$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} \left(\frac{\partial L}{\partial w_i} \right) ; \quad \forall w_i \in \overline{W} \quad (5.20)$$

A ideia de multiplicar o gradiente pelo inverso de $\sqrt{A_i}$ é originária da normalização sinal-ruído ou *signal-to-noise ratio* [111], uma vez que A_i mede apenas o ruído, ou a magnitude histórica do gradiente do parâmetro w_i , ao invés do seu sinal [110]. De acordo com o método AdaGrad, isso incentiva movimentos relativos mais rápidos ao longo de direções suavemente inclinadas da função de perda L .

Assim, o algoritmo AdaGrad otimiza o cálculo dos gradientes. Entretanto, foi descoberto empiricamente que, no treinamento de redes com muitas camadas, o acúmulo de gradientes quadráticos desde o início do treinamento pode resultar em uma diminuição prematura e excessiva na efetividade do aprendizado [104].

5.2.5 RMSProp

O algoritmo RMSProp [112] também objetiva otimizar de forma independente o gradiente de cada parâmetro da rede e usa uma motivação semelhante ao AdaGrad ao utilizar a normalização sinal-ruído. No entanto, em vez de simplesmente adicionar os gradientes quadrados para estimar A_i , essa abordagem usa a média exponencial. Ao usar a média para normalizar em vez de agregar valores, o progresso não é retardado prematuramente por um fator de escala A_i que aumenta constantemente. A ideia básica é usar um fator de decaimento $\rho \in [0, 1]$ e ponderar as derivadas parciais quadradas que ocorreram há t atualizações anteriores por ρt . Isso é alcançado multiplicando o agregado atual ao quadrado (isto é, estimativa em execução) por ρ e, em seguida, adicionando $(1 - \rho)$ vezes a derivada parcial atual (ao quadrado). Portanto, se A_i é o valor exponencialmente médio do *i*-ésimo parâmetro w_i , a atualização de A_i é realizada conforme a Expressão 5.21.

$$A_i \leftarrow \rho A_i + (1 - \rho) \left(\frac{\partial L}{\partial w_i} \right)^2 ; \quad \forall w_i \in \overline{W} \quad (5.21)$$

De forma semelhante ao algoritmo AdaGrad, a raiz quadrada de A_i para cada parâmetro é usada para normalizar seu gradiente. Então, a atualização do *i*-ésimo parâmetro w_i é mostrada na Expressão 5.22, considerando para a taxa de aprendizado α global.

$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} \left(\frac{\partial L}{\partial w_i} \right) ; \quad \forall w_i \in \overline{W} \quad (5.22)$$

Apesar da introdução do hiperparâmetro ρ , empiricamente, o RMSProp mostrou ser um algoritmo de otimização eficaz para redes neurais com muitas camadas [104].

5.2.6 ADAM

O algoritmo de ADAM (*ADaptive Moment estimation*) [113] também usa uma normalização sinal-ruído semelhante aos algoritmos AdaGrad e RMSProp vistos nas Seções 5.2.4 e 5.2.5. No entanto, ele pode ser visto como uma variante do algoritmo RMSProp combinado com a técnica de momento clássico, com uma distinção importante: são incorporadas, em cada iteração de atualização dos parâmetros $w_i \in \overline{W}$, as estimativas de momento de primeira e segunda ordem [104]. Isso proporciona ganho de eficiência no processo de convergência da função de perda L , apesar do maior consumo de memória e da necessidade de um fator de decaimento adicional.

Como no caso do algoritmo RMSProp, considera-se A_i como o valor exponencialmente médio do i -ésimo parâmetro w_i . Este valor é atualizado da mesma forma que no algoritmo RMSProp, com o fator de decaimento $\rho \in [0, 1]$, conforme mostrado na Expressão 5.21. Ao mesmo tempo, um valor exponencialmente suavizado do gradiente é mantido para o i -ésimo componente, sendo esse valor denotado por F_i . Esta suavização é realizada com um fator de decaimento adicional ρ_f , conforme Expressão 5.23.

$$F_i \leftarrow \rho_f F_i + (1 - \rho_f) \left(\frac{\partial L}{\partial w_i} \right); \quad \forall w_i \in \overline{W} \quad (5.23)$$

Este tipo de suavização exponencial do gradiente com ρ_f é uma variação do método do momento clássico (Seção 5.2.2), que é parametrizado por um parâmetro de atrito β , ao invés de ρ_f [9]. Assim, a atualização do parâmetro w_i , considerando uma taxa de aprendizado dinâmica α_e na e -ésima iteração (época), é mostrada na Expressão 5.24.

$$w_i \leftarrow w_i - \frac{\alpha_e}{\sqrt{A_i}} F_i; \quad \forall w_i \in \overline{W} \quad (5.24)$$

Portanto, existem duas diferenças principais em relação ao algoritmo RMSProp [9]. Primeiro, o gradiente é substituído por seu valor suavizado exponencialmente para incorporar o momento. Em segundo lugar, a taxa de aprendizado α_e depende de uma taxa inicial α_0 e do índice de iteração da época e , definida conforme Equação 5.25.

$$\alpha_e = \alpha_0 \left(\frac{\sqrt{1 - \rho^2}}{1 + \rho_f^e} \right) \quad (5.25)$$

O algoritmo ADAM é, frequentemente, utilizado para minimizar o problema de cálculo do gradiente e otimização eficiente dos parâmetros de redes neurais [113]. Assim, ele é um método que requer pouca memória, tem bom desempenho e converge rapidamente [114].

5.2.7 NADAM

O algoritmo NADAM [114] incorpora o momento Nesterov, visto na Seção 5.2.3, ao algoritmo ADAM apresentado na Seção 5.2.6. O algoritmo ADAM original tem dois componentes principais: um componente de momento clássico e uma taxa de aprendizado adaptativa. No entanto, o momento clássico pode ser considerado conceitualmente e empiricamente como inferior ao algoritmo de Nesterov em relação a velocidade de convergência e qualidade do aprendizado [114].

De forma semelhante aos algoritmos RMSProp e ADAM, considera-se A_i como o valor exponencialmente médio do i -ésimo parâmetro w_i , atualizado pelo fator de decaimento $\rho \in [0, 1]$. Entretanto, essa atualização difere em relação a parametrização da derivada parcial da função de perda L no cálculo do gradiente, assumindo a porção de momento da atualização $(w_i + \rho A_i)$, conforme mostrado na Expressão 5.26.

$$A_i \Leftarrow \rho A_i + (1 - \rho) \left(\frac{\partial L(w_i + \rho A_i)}{\partial w_i} \right)^2 ; \quad \forall w_i \in \overline{W} \quad (5.26)$$

Ao mesmo tempo, considera-se F_i um valor exponencialmente suavizado do gradiente para o i -ésimo parâmetro, atualizado pelo fator de decaimento ρ_f . Assim como ocorre no cálculo de A_i , a atualização de F_i difere em relação a parametrização da derivada parcial da função de perda L no cálculo do gradiente, assumindo a porção de momento da atualização $(w_i + \rho_f F_i)$, conforme mostrado na Expressão 5.27.

$$F_i \Leftarrow \rho_f F_i + (1 - \rho_f) \left(\frac{\partial L(w_i + \rho_f F_i)}{\partial w_i} \right) ; \quad \forall w_i \in \overline{W} \quad (5.27)$$

De modo semelhante ao algoritmo ADAM, a atualização de cada parâmetro $w_i \in \overline{W}$ é realizada conforme Expressão 5.24. O cálculo da taxa de aprendizado dinâmica α_e na iteração/época e ocorre conforme Expressão 5.25.

5.3 Redes Neurais Recorrentes

Uma Rede Neural Recorrente (RNN) é uma especialização de rede neural artificial adequada para análise de comportamento temporal dinâmico, no qual conexões entre nós efetivamente formam um grafo direcionado [103]. Este tipo de rede neural deriva do conceito de retropropagação de erros e possibilita a realização de tarefas antes inviáveis, como o reconhecimento de escrita manual [102]. As RNNs possuem um estado interno (memória) que pode ser usado para armazenar informações relevantes e processar sequências de dados temporais de comprimento variável [115], sendo que essa memória fica contida na camada oculta.

Alguns exemplos de aplicações de RNNs incluem reconhecimento de palavras e tradução entre idiomas, assim como problemas de previsão utilizando séries temporais [9].

5.3.1 Arquitetura RNN

A arquitetura RNN se baseia em uma rede multicamadas, assim como na rede tradicional *feed-forward* (Seção 5.1.3). A diferença estrutural entre as duas redes reside no fato do conjunto de ativações das unidades ocultas retornarem à rede juntamente com as entradas, conforme a representação de arquitetura mostrada na Figura 5.6.

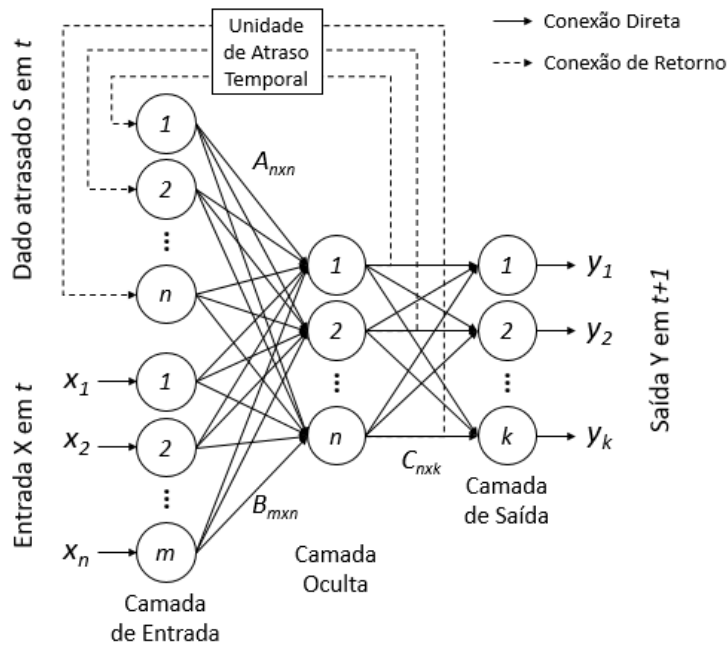


Figura 5.6: Representação da arquitetura da rede RNN com conexões diretas e de retorno, adaptada de [12] e [13].

Na arquitetura, $X(t)$ é o vetor de m entradas no tempo discreto t e $Y(t+1)$ denota o vetor de k saídas no tempo $t+1$. O vetor de entrada $X(t)$ e o vetor de saídas com retardo de um passo na camada oculta $S(t)$ são concatenados para formar o vetor μ de tamanho $(m+n)$, no qual o i -ésimo elemento no tempo t é denotado por $\mu_i(t)$. Seja M o conjunto de índices i para o qual x_i em t é uma entrada externa e N denota o conjunto de índices i para o qual s_i é a saída de uma unidade na rede também em t , então [13]:

$$\mu_i(t) = \begin{cases} x_i(t) & \text{se } i \in M, x \in X \\ s_i(t) & \text{se } i \in N, s \in S \end{cases} \quad (5.28)$$

De forma semelhante, o vetor $B(t)$ dos pesos das conexões de entrada e o vetor $A(t)$ dos pesos das conexões de retorno da camada oculta, ambos no tempo t , são concatenados

para formar o vetor ν de tamanho $(m + n)$, no qual o i -ésimo elemento no tempo t é denotado por $\nu_i(t)$, conforme Equação 5.29.

$$\nu_i(t) = \begin{cases} b_i(t) & \text{se } i \in M, b \in B \\ a_i(t) & \text{se } i \in N, a \in A \end{cases} \quad (5.29)$$

Assume-se que a rede seja totalmente interconectada, portanto existem $m \times n$ conexões diretas e $n \times m$ conexões de retorno. Sejam A , B e C as matrizes de pesos das conexões recorrentes, das entradas e das saídas da rede, respectivamente, a atividade do neurônio j no tempo t é calculada pela Equação 5.30 [13].

$$rnn_j(t) = \sum_{i \in MUN} \nu_{i,j}(t-1) \mu_i(t-1) \quad (5.30)$$

A saída do neurônio j na camada oculta é dada aplicando-se a função de ativação não linear $f(\cdot)$, conforme mostrado na Equação 5.31 [13].

$$s_i(t) = f(rnn_j(t)) \quad (5.31)$$

A saída do neurônio k da camada de saída da RNN no tempo t , considerando a função de ativação $g(\cdot)$, que pode ser igual à função $f(\cdot)$ em arquiteturas homogêneas, é dada pela Definição 5.32 [13].

$$\begin{aligned} rnn_k(t) &= \sum c_{k,j}(t) s_i(t) \\ y_k(t) &= g(rnn_k(t)) \end{aligned} \quad (5.32)$$

A Figura 5.7 mostra a arquitetura de uma RNN desdobrada para análise das entradas nos tempos $t-2$, $t-1$ e t , na forma de um modelo de estados [14], onde A , B e C são matrizes de pesos das dimensões de ativação, entrada e saída da unidade, respectivamente.

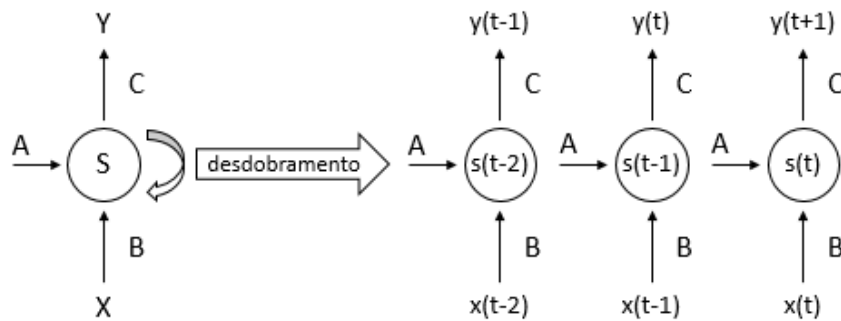


Figura 5.7: Arquitetura da rede RNN desdobrada no tempo [14].

Assumindo um modelo de estado s relacionado ao tempo t , a Equação 5.33 é definida como segue. O valor θ é um deslocamento fixo das variáveis de entrada x_t , sendo que f é

a função de ativação da rede. Desse modo, o sistema pode ser visto como um sistema de transição de estados dinâmico parcialmente observável [14], com $s_t \rightarrow s_{t+1}$ que também é impulsionado por forças externas x_t .

$$\begin{aligned} s(t+1) &= f(As(t) + Bx(t) + \theta) \\ y(t) &= g(Cs(t)) \end{aligned} \tag{5.33}$$

Normalmente, uma função não linear é usada como função de ativação [14] em uma RNN. A função linear *Rectified Linear Unit* (ReLU), apresentada na Equação 5.34, pode ser utilizada em abordagens mais simples. Entretanto, as funções sigmoidal logística e tangente hiperbólica, representadas nas Equações 5.35 e 5.36, respectivamente, são as funções de ativação comumente utilizadas em RNNs, assumindo que $v = \sum w_i x_i + b$, w_i são os pesos dos valores x_i de entrada do nó e b é uma constante.

$$\phi(v) = \max(v, 0) \quad (\text{Rectified Linear Unit (ReLU)}) \tag{5.34}$$

$$\sigma(v) = \frac{1}{1 + e^{-v}} \quad (\text{sigmoidal logística}) \tag{5.35}$$

$$\tanh(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}} \quad (\text{tangente hiperbólica}) \tag{5.36}$$

As funções sigmoidal logística e tangente hiperbólica são utilizadas devido a não linearidade e simplicidade computacional de suas derivadas [116].

5.3.2 Redes LSTM

As RNNs usam conexões recorrentes para armazenar representações de eventos recentes na forma de ativações (memória de curto prazo). Aprender a armazenar informações em intervalos de tempo estendidos por retropropagação recorrente de erros geralmente leva um tempo considerável. As redes neurais recorrentes *Long Short Term Memory* (LSTM) introduziram um método eficiente baseado em gradiente descendente (Seções 5.1.4, 5.1.5, 5.1.6 e 5.2) para aprender informações em intervalos de tempo estendidos [117].

A arquitetura da rede LSTM é similar a da RNN tradicional (Seção 5.3.1) e sua principal diferença reside no nó LSTM. A estrutura de nó LSTM mais comum, chamada *vanilla LSTM*, originalmente definida por Graves e Schmidhuber [118], é apresentada na Figura 5.8. A capacidade de adicionar ou remover informações do estado da célula é regulada por estruturas denominadas portas. Elas são responsáveis por fazer com que as informações mais relevantes prossigam através do nó. São compostas de uma função

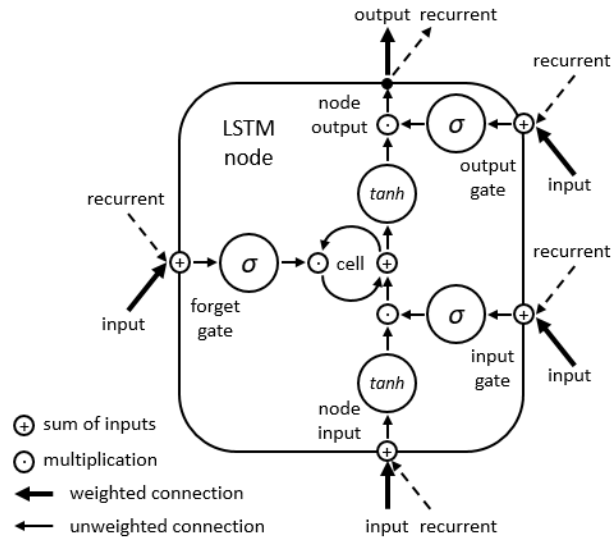


Figura 5.8: Nó LSTM utilizado nas camadas ocultas da rede neural [15].

de ativação sigmoidal logística (Equação 5.35) e de uma operação de multiplicação das matrizes de pesos.

A estrutura apresentada na Figura 5.8 é composta por portas de entrada, de esquecimento e de saída. Além das portas, a estrutura conta com bloco de entrada de dados, célula de “memória”, função de ativação e bloco de saída. Tanto o bloco de entrada quanto o de saída da estrutura usam a função de ativação tangente hiperbólica (Equação 5.36). Existem conexões recorrentes do bloco de saída para o bloco de entrada e também para cada uma das portas.

A ideia central por trás da estrutura LSTM é uma célula de memória, que pode manter seu estado ao longo do tempo, e unidades não lineares denominadas de portas, que regulam o fluxo de informações para dentro e fora da célula [15]. Inicialmente, a estrutura recebe valores de entrada, das camadas anteriores, e valores recorrentes, das camadas posteriores, que compõem um vetor único. É aplicada uma função de ativação tangente hiperbólica (Equação 5.36), que ajuda a regular os valores no intervalo -1 e 1.

A porta de esquecimento (*forget gate*) permite que a estrutura LSTM redefina o seu próprio estado em tarefas contínuas, cujo processamento trata de fluxos de entrada contínuos sem que o término da sequência seja explicitamente delimitado [119]. A redefinição de estado, em momentos apropriados, proporciona a retenção dos dados efetivamente relevantes em processos de aprendizado de longo prazo. Isso é possível graças ao uso da função de ativação sigmoidal (Equação 5.35) pelas portas, que produz valores intervalo entre 0 (descarte) e 1 (continuidade).

A porta de entrada define quais dados poderão ser inclusos ao estado atual da célula. Os dados de entrada e recorrentes são passados por uma função sigmoidal, que define

valores relevantes ou não. Nesse momento, o estado atual da célula é multiplicado (vetorialmente) pela saída da porta de esquecimento. Logo após, os dados adicionados passam por uma função de ativação tangente hiperbólica (Equação 5.36) para ajudar a regular os valores saída da rede, no intervalo entre -1 e 1.

Finalmente, a porta de saída influencia nos valores finais de saída resultantes da mudança de estado atual. Novamente, uma função sigmoideal é utilizada, cujos valores são multiplicados vetorialmente aos resultados processados pela célula, após a uma nova aplicação da função tangente hiperbólica. A saída da rede alimenta as conexões de saída para as camadas posteriores e as recorrentes para as camadas anteriores.

A memória de longo prazo é incorporada pela alteração lenta dos pesos das conexões. Assim como nas RNNs tradicionais, o treinamento é feito com algoritmos de cálculo do gradiente descendente. Uma variação do algoritmo gradiente descendente estocástico combinada com a técnica de cálculo do momento clássico é utilizada em [120], fazendo com que a rede neural produza resultados com desempenho satisfatório, desde que os pesos sejam inicializados corretamente. Como visto na Seção 5.2.2, a técnica de momento clássico é utilizada para acelerar a “descida” ou redução do gradiente por meio do acúmulo de um vetor de velocidade em direções de redução persistente, no decorrer iterações do algoritmo [108].

Capítulo 6

Precificação Dinâmica em Computação em Nuvem

A precificação de recursos computacionais, mais especificamente de instâncias de máquinas virtuais em provedores de computação em nuvem pública (Seção 2.5), é uma área de pesquisa bastante promissora e trata de diversos aspectos a serem analisados e explorados.

Este capítulo apresenta um levantamento sobre as pesquisas relacionadas a esse tema, com o objetivo de se identificar técnicas que proporcionem maior disponibilidade de recursos em ambientes dinâmicos, ao menor custo possível para os usuários. Também são considerados trabalhos que tratam da melhor eficiência na utilização da infraestrutura computacional disponibilizada pelos provedores de computação em nuvem.

Para cada trabalho, são analisados o problema, a técnica utilizada e os resultados quantitativos e qualitativos obtidos, com vistas à identificação dos trabalhos mais relevantes da literatura. Desse modo, na Seção 6.1 serão detalhados os trabalhos selecionados. Na Seção 6.2, uma análise comparativa dos trabalhos é apresentada, considerando as principais características identificadas.

6.1 Estado da Arte em Precificação

A seguir, serão analisados os trabalhos que representam o estado da arte em precificação dinâmica de recursos no contexto de computação em nuvem.

6.1.1 Lee e co-autores (2011)

Lee e co-autores [26] publicaram em 2011 um dos primeiros trabalhos que analisa as características dos tipos de instância *on demand* da Amazon EC2. O objetivo do trabalho era escalonar a execução de uma aplicação composta de tarefas independentes em um

ambiente de nuvem heterogêneo e compartilhado. Em 2011, foram utilizados apenas 11 tipos de instâncias EC2 e, mesmo com esse número reduzido de instâncias, os autores concluíram que a relação entre as variáveis preço, desempenho e quantidade de memória RAM na Amazon EC2 é complexa. Eles sugerem uma abordagem denominada de *ProgressShare*, que busca escalonar as tarefas em tipos diferentes de instâncias de forma que cada uma receba sua parcela justa (ou *fair*) de recursos, proporcionando bom desempenho da carga de trabalho como um todo.

A decisão de escalonamento de tarefas analisa diferentes combinações de tipos de instância, considerando custo e desempenho destas combinações, de modo que o progresso na execução das tarefas avance de forma uniforme em todas as instâncias participantes. No estudo de caso, foram executadas tarefas de criptografia e ordenação de dados, de forma paralela e independente, utilizando a ferramenta Apache Hadoop. A solução considerou características da aplicação do usuário que possuem melhor afinidade com os propósitos específicos das instâncias, como uso intensivo de CPU para execução de algoritmos de criptografia. Como resultado, a sugestão de instâncias para cada aplicação proporcionou alocação adequada dos recursos, indiretamente proporcionando menor tempo total de execução e redução de custos, se comparada com uma abordagem genérica, ou seja, que não considerasse as características da aplicação e das instâncias de computação em nuvem.

6.1.2 Zhang e co-autores (2012) - CloudRecommender

O CloudRecommender [27] usa uma abordagem declarativa baseada em ontologias com dois objetivos principais: (a) unificar a especificação de tipo de instância de vários provedores de nuvem e (b) sugerir instâncias *on demand* com preço razoável que respeitem os requisitos das aplicações. Foram consideradas três características de instâncias em vários provedores de nuvem: capacidade de processamento, largura de banda da rede e capacidade de armazenamento. Usando os requisitos das aplicações e a ontologia definida, o CloudRecommender seleciona instâncias que respeitam os requisitos e que oferecem custo agregado mínimo para o usuário.

A contribuição mais relevante do trabalho consiste na definição de uma ontologia que torna possível uma comparação objetiva entre instâncias de diferentes provedores de nuvem, além do fato de classificar características qualitativas que são mais apropriadas para determinados tipos de aplicações.

6.1.3 Ben-Yehuda e co-autores (2013)

No trabalho de Ben-Yehuda e co-autores [30], foram analisados os históricos de precificação de 8 diferentes tipos de instâncias *spot* do provedor de nuvem Amazon EC2, em 4

regiões geográficas distintas e considerando 2 sistemas operacionais (Linux e Windows), durante o período de novembro de 2009 até abril de 2010. Foi aplicado um mecanismo de engenharia reversa sobre os preços *spot* praticados e os autores constataram que os preços não eram orientados por um mecanismo de mercado até outubro de 2011, conforme alegava o provedor. Ao invés disso, os autores concluíram que os preços eram gerados aleatoriamente a partir de um estreito intervalo de preços definido pelo provedor, desconsiderando os lances oferecidos pelos usuários, de forma a evitar que as instâncias fossem vendidas a um preço muito baixo e, ao mesmo tempo, dando a falsa impressão de crescimento da demanda.

O trabalho utilizou resultados de simulação do modelo de preços definidos artificialmente pela Amazon e os comparou com simulações utilizando o modelo de mercado baseado em leilões selados de $(N + 1)$ preços (Seção 4.5). Desse modo, os autores conseguem comprovar que o histórico de preços publicado pelo provedor se assemelhava mais ao modelo de preços artificiais do que ao modelo baseado em mercado. Outro achado importante foi que os preços definidos artificialmente para as instâncias *spot* possuíam correlação com os preços *on demand*.

Cabe ressaltar que a Amazon mudou a estratégia de preços, sendo que a partir de 2012 os autores constataram a utilização de um mecanismo de mercado baseado em leilão na precificação de instâncias *spot*.

6.1.4 Javadi e co-autores (2013)

Em [29], Javadi e co-autores fazem uma análise estatística detalhada de várias instâncias *spot* em quatro zonas diferentes, usando dados históricos fornecidos pela Amazon, de fevereiro de 2010 a fevereiro de 2011. Os autores assumiram um modelo baseado em mercado e consideraram duas variáveis: o preço *spot* em um determinado momento e tempo inter-preço (ou seja, o tempo entre as mudanças de preços). Essa é a escolha natural, pois a Amazon disponibiliza histórico de preços nesse formato [24]. Com isso, os autores foram capazes de identificar correlações em termos de hora do dia e dia da semana. O preço *spot* e os tempos inter-preço foram modelados como distribuições de Mistura de Gaussianos (MoG) e essas distribuições foram usadas por um modelo de calibração para identificação de tendências. A principal desvantagem deste estudo é que ele assume que os preços *spot* são definidos por um modelo orientado a mercado, sem influência externa. No período considerado pelos autores (de fevereiro de 2010 a fevereiro de 2011), esta hipótese foi contrariada por Ben-Yehuda e co-autores (Seção 6.1.3), mostrando indicações de preços artificialmente estabelecidos pela Amazon até outubro de 2011. Portanto, não é claro que a distribuição MoG seja adequada.

Javadi e co-autores [29] afirmam que a distribuição da Mistura de Gaussianos (MoG) produz melhores aproximações de preços no modelo *spot* da Amazon, enquanto Karunakaran e Sundarraj (Seção 6.1.7) argumentam que a distribuição lognormal é mais adequada. Como pode ser visto, não há consenso sobre qual distribuição usar e não há indicação de que uma única distribuição se encaixe no modelo de mercado *spot* como um todo.

6.1.5 Tanaka e Murakami (2014)

A pesquisa de Tanaka e Murakami [42] propõe solução para o problema de seleção de serviços disponíveis em computação em nuvem no modelo SaaS, com o objetivo de obter a melhor composição de serviços que atenda aos requisitos de preço máximo e qualidade mínima definidos pelo usuário. Na modelagem, o usuário define sua aplicação como um *workflow* de serviços abstratos, sendo que cada serviço abstrato é implementado por um grupo ou *cluster* de serviços concretos. Com base nos requisitos de nível mínimo de qualidade de serviço da aplicação e preço máximo que o usuário está disposto a pagar, os serviços concretos são selecionados para cada serviço abstrato que faz parte do *workflow*.

Para a interação entre usuários e provedores, é descrito um modelo de mercado com base em leilão invertido ou pregão (Seção 4.5). De um lado, os provedores oferecem seus serviços a um determinado preço e qualidade. De outro, os usuários selecionam a combinação dos dois e pagam o preço proposto. O provedor obtém lucro quando o preço pago pelo serviço ofertado é superior ao seu custo e inferior aos demais serviços ofertados. Neste caso, é adotado o mecanismo Vickrey-Clarke-Groves (VCG), que foi objeto de estudo na Seção 4.6, no qual a estratégia dominante para o provedor de serviço consiste em reportar o custo real do seu serviço. Assim, a pesquisa utiliza o mecanismo VCG para mercado de serviços e propõe um algoritmo de programação dinâmica, cujo objetivo é a melhoria de desempenho para seleção de recursos e cálculo de pagamentos a serem realizados.

A avaliação na seleção de serviços e o cálculo de pagamentos é feita considerando os critérios de tempo de execução para finalização do processo e taxa de sucesso da composição de serviços. Os tempos de execução são investigados em função do número de *clusters* de serviços e do número de serviços concretos, que variam de 5 até 100. Ao final, é feita uma comparação da solução com o modelo de preço fixo, que mostra que a diminuição da taxa de lucro leva ao aumento da taxa de sucesso da seleção de uma composição. É mostrado também o equilíbrio da utilidade do sistema, que considera tanto a função do usuário (eficiência na utilização dos recursos) quanto a do provedor (lucro).

6.1.6 Huang e co-autores (2015) - Cumulon

Cumulon [28] é um sistema em nuvem que executa tarefas únicas e *workflows* de longo prazo, compostos de várias tarefas independentes. A solução primeiramente compõe um plano de execução básico usando apenas instâncias *on demand* e o otimiza criando um plano alternativo que usa instâncias *spot* adicionais. O objetivo é reduzir o custo financeiro geral e respeitar uma tolerância ao risco de indisponibilidade definida pelo usuário, sendo que tolerância ao risco somada à disponibilidade deve ser igual a 1 (um).

Os autores definem um modelo de preço de mercado e usam-no para estimar quais tipos de instâncias *spot* devem ser usadas em um determinado período de tempo. Mesmo que os autores afirmem que há periodicidade diária e semanal nas variações do preço, eles usam um modelo não periódico para coletar e analisar os resultados experimentais. A avaliação considera, por exemplo, o custo de conclusão da tarefa do usuário, o tempo de espera e a quantidade de interrupções para cada tipo de instância.

6.1.7 Karunakaran e Sundarraj (2015)

Karunakaran e Sundarraj [31] discutem quatro estratégias de lances para o mercado de leilões *spot* da Amazon, considerando a execução de *jobs* de longa duração (≥ 50 horas). Uma distribuição lognormal é usada para simular preços *spot*. Neste estudo, os autores mostram que os lances que são muito baixos ou muito próximos do preço *on demand* não proporcionam um bom equilíbrio entre o preço pago e o número de interrupções na execução. Eles sugerem que os valores intermediários entre esses dois extremos devem ser usados.

Além de analisar estratégias de lances a serem adotadas pelo usuário, a pesquisa analisa a influência do preço de lance no custo total de processamento, considerando diversos tipos de instâncias. É analisada também a influência no tempo de espera e na quantidade de interrupções no processamento. As estratégias sugeridas podem ser utilizadas em tipos específicos de aplicações, como *webcrawling* ou processamento analítico de dados [31].

6.1.8 Singh e co-autores (2015)

Singh e co-autores [32] propõem uma solução de previsão de preço *spot* para lances de curto prazo (1 dia à frente) e médio prazo (5 dias à frente). A solução usa uma técnica de otimização que emprega o algoritmo gradiente descendente tradicional (Seção 5.1.4) para prever o preço *spot* por hora. Os dados brutos são, em primeiro lugar, submetidos a um procedimento de normalização e de cálculo de média de preços por hora. O conjunto de dados de aprendizagem é composto pelas médias de preço das últimas 24 horas e pelas médias dos últimos 3 meses com relação à mesma hora a ser prevista. O algoritmo

gradiente descendente calcula os parâmetros de peso para os dois componentes de média de preço usados para a previsão.

Na avaliação experimental, os autores utilizaram os dados do histórico de preços de janeiro a abril de 2014 para o tipos de instâncias *m1.large*, *m2.2xlarge* e *g2.2xlarge*, nas regiões *ap-southeast*, *sa-east* e *us-west*. A métrica de avaliação utilizada para a previsão de preços foi o erro médio absoluto percentual ou *Mean Absolute Percent Error* (MAPE), cujos erros calculados resultaram em valores próximos a 9,4% para curto prazo e menos de 20% para previsões de médio prazo.

6.1.9 Lucas-Simarro e co-autores (2015)

Na pesquisa realizada por Lucas-Simarro e co-autores [33], é proposto um algoritmo de escalonamento capaz de tomar decisões baseadas em diferentes mecanismos de precificação. No caso específico de precificação dinâmica, o algoritmo considera a previsão de preços das VMs no próximo período, com o objetivo de minimizar o custo total de infraestrutura computacional para o usuário. Para isso, são utilizadas técnicas de médias móveis (Seção 4.3) para previsão de preços: média móvel simples ou *Simple Moving Average* (SMA), que usa dados passados para obter o melhor preço médio em um único momento, mas que não considera tendências de preços; média móvel ponderada ou *Weighted Moving Average* (WMA), que supera a SMA, permitindo valorizar dados mais recentes, atribuindo diferentes pesos a esses dados; média móvel exponencial ou *Exponential Moving Average* (EMA), bastante semelhante à WMA, sendo que os pesos atribuídos diminuem em progressão exponencial, ao invés de progressão aritmética.

A solução também considera restrições de desempenho, tais como quantidades mínimas de CPUs virtuais ou tamanho mínimo de memória; restrições de localização, que limitam à utilização de um determinado tipo de provedor; e restrições de unidade, que determinam a quantidade de tipos específicos de instâncias que devem ser utilizadas. Nas avaliações feitas, a técnica EMA proporcionou os melhores resultados, considerando o modelo de precificação *spot* do serviço Amazon EC2.

6.1.10 Ouyang e co-autores (2016) - SpotLight

O SpotLight [34] tem como objetivo auxiliar os usuários em compreender a dinâmica das políticas de precificação de computação em nuvem. Os autores propõem uma abordagem de sondagem ativa para monitorar a disponibilidade de instâncias *on demand* e *spot* do provedor AWS. Com esta abordagem, o SpotLight solicita a instância à Amazon e marca como "disponível", se a solicitação for concedida (passo 5 na Seção 2.5.2), ou "não disponível", caso contrário. Embora essa abordagem possa fornecer uma fotografia precisa

da disponibilidade das instâncias, a desvantagem óbvia é que a sondagem incorre em um custo financeiro. Para reduzir esse custo, os intervalos de monitoramento são definidos dinamicamente pelo usuário.

O trabalho aponta indícios de que o provedor coloca à disposição, para determinado tipo de instância, um *pool* de tamanho único para os tipos de precificação *reserved*, *on demand* e *spot*. Os autores também encontraram indícios de que a disponibilidade de instâncias *spot* segue tendência inversa a de instâncias *on demand*, o que está de acordo com a tese de que os mecanismos de precificação são complementares.

Finalmente, o trabalho indica que a premissa de que uma aplicação sempre pode utilizar instâncias *on demand*, como solução para tolerância a falhas no caso em que um servidor *spot* for revogado, não está correta. Isso porque revogações de instâncias *spot* muitas vezes correlacionam-se com períodos de indisponibilidade em instâncias *on demand*, quando há necessidade de provimento de instâncias *reserved* contratadas previamente ou quando o provedor atinge valores próximos à capacidade física de provisionamento.

6.1.11 Li e co-autores (2016)

O trabalho de Li e co-autores [121] fornece uma pesquisa extensiva sobre preços *spot*, considerando 61 artigos e mostrando os benefícios e limitações da precificação do provedor AWS. Os autores também afirmam que, posteriormente ao ano de 2011, os preços *spot* parecem não observar nenhuma lei e são totalmente incertos, dando a evidência de que, de fato, a partir do final desse ano os preços seguem um modelo orientado a mercado.

A pesquisa classifica os artigos que tratam do modelo de precificação *spot* de acordo com quatro tipos de teorias: (a) teoria descritiva, que fornece uma descrição dos preços com base em observações e análises estatísticas, considerando mecanismo de precificação como uma caixa preta; (b) teoria explicativa, que esclarece o problema de precificação *spot*, assim como as causas para as variações de preço com base em um mecanismo de mercado e leilões (Seção 4.5); (c) teoria preditiva, abordando a simulação de preços sem ter ciência dos detalhes causais de oferta e demanda, cujo mecanismo gerador também é compreendido como uma caixa preta; e (d) teoria prescritiva, que fornece uma prescrição explícita (por exemplo, métodos, técnicas, princípios, funções ou uma combinação deles) para funcionamento do mecanismo de preços *spot*. Na primeira classificação (descritiva), há uma predominância no uso de técnicas estatísticas. Na segunda (explicativa), são utilizados mecanismos modelagem econômica e teoria de jogos. Na terceira classificação (preditiva), são utilizados principalmente modelos de regressões, séries temporais (Seção 4.2) e técnicas de *machine learning*, de forma predominante as redes neurais artificiais (Seção 5.1). Na última classificação (prescritiva), são utilizados mecanismos de mercado,

análises estatísticas, soluções de otimização e métodos econômicos baseados em equilíbrio, como teoria de jogos (Seção 4.4).

Como importantes conclusões da pesquisa, podem ser destacadas as evidências de que o modelo de precificação dinâmica, no caso *spot*, é mais eficiente quanto à utilização de infraestrutura ociosa do que os modelos de precificação estática, contribuindo para uma melhor definição do valor de mercado dos recursos disponibilizados. A consequência direta desse fato é que o modelo dinâmico proporciona maior lucro ao provedor, em função da utilização mais eficiente dos recursos computacionais. Além disso, proporciona a oferta de recursos mais baratos ao usuário, caso o mesmo esteja preparado para possíveis interrupções na execução de sua aplicação.

6.1.12 Agarwal e co-autores (2017)

O artigo de Agarwal e co-autores [44] apresenta uma das primeiras abordagens de uso das redes neurais LSTM (Seção 5.3.2) para previsão de preços *spot*. O trabalho usa uma configuração de rede com apenas uma camada de nós LSTM. A média de todas as variações de preço durante um dia foi considerada para separar as etapas de tempo uniformemente na etapa de treinamento.

Neste trabalho, foram considerados dados históricos de preços de 6 tipos de instância, de março a abril de 2017. A proporção de dados de aprendizagem e teste foi de 67% e 33%, respectivamente. Os resultados mostram um erro de 1% a 8,6% para os tipos de instância avaliados. Os autores ainda argumentam sobre a obtenção de melhores resultados na previsão de preços *spot* utilizando redes LSTM em comparação com outras abordagens, como a aplicação do algoritmo de otimização gradiente descendente (Seção 5.1.4) e o uso de redes neurais tradicionais de alimentação direta (Seção 5.1.2).

6.1.13 Wolski e co-autores (2017) - DrAFTS

No trabalho de Wolski e co-autores [35], o objetivo consiste em definir uma sugestão de lance para o usuário de computação em nuvem *spot* da Amazon, de forma a garantir uma duração fixa de tempo de execução com uma determinada probabilidade associada. É aplicada uma técnica de previsão ao histórico de preços denominada DrAFTS (Durability Agreements from Time Series), que utiliza série temporal (Seção 4.2) para obtenção da estimativa de disponibilidade da instância. O valor de lance mínimo é ajustado de forma a garantir uma disponibilidade da instância *spot* por, pelo menos, 55 minutos, com confiança de 95%.

Na realização dos testes, foram usados dados de histórico de preços *spot* de diversos tipos de instâncias e regiões/zonas de disponibilidade. Nos resultados para o período de

outubro a dezembro de 2016, a economia de custo chegou a 68,53% em comparação com preços sob demanda.

6.1.14 Al-Theibat e co-autores (2018)

Em [36], o problema de previsão de preço *spot* é estudado a partir da comparação do modelo estatístico ARIMA (Seção 4.2) com uma abordagem de redes neurais LSTM (Seção 5.3.2). A arquitetura da rede é composta por 3 camadas: uma camada de entrada, uma camada LSTM oculta com 100 nós e uma camada densa de saída da rede. O algoritmo ADAM (Seção 5.2.6) é utilizado para o cálculo dos gradientes no processo de otimização dos pesos das conexões dos nós LSTM.

Os autores analisaram 3 meses de histórico de precificação *spot* de 4 tipos de instância (c3.2xlarge, c1.xlarge, m4.large e c3.4xlarge) em diferentes regiões, no período de dezembro de 2017 a março de 2018. Note que, nesse período, a Amazon já havia simplificado o seu modelo de mercado *spot* (Seção 6.1.19). A conclusão é que a abordagem LSTM supera o modelo estatístico, obtendo menor erro com as funções de erro *Root Mean Squared Error* (RMSE), *Mean Absolute Percentage Error* (MAPE) e *Mean Absolute Scaled Error* (MASE).

6.1.15 Baughman e co-autores (2018)

O trabalho de Baughman e co-autores [25] aborda a previsão de preços *spot* para o cálculo de garantias de durabilidade da instância. Os preços *spot* são tratados como uma série temporal (Seção 4.2) e uma arquitetura híbrida composta de duas camadas LSTM (Seção 5.3.2), cada uma com 32 nós de largura, mais uma camada densa, é usada para realização das previsões de preços.

O algoritmo ADAM (Seção 5.2.6) e a sua variação NADAM (Seção 5.2.7), são utilizadas, sendo que o cálculo da função MSE é a métrica de erro. Os resultados comparados a um modelo de regressão ARIMA (Seção 4.2) mostram que a arquitetura LSTM proposta fornece bons resultados, com baixo erro de estimativa. As desvantagens da rede LSTM, no entanto, são a necessidade de um tempo de treinamento extenso (~ 13 horas) e o fato de as análises ficarem restritas aos dados do histórico de preços para um mês (setembro de 2016), considerando unicamente a instância c3.2xlarge.

6.1.16 Sharma e co-autores (2018)

A solução denominada SpotCheck, proposta por Sharma e co-autores [37], propõe uma camada IaaS que oferece a ilusão de VMs sempre disponíveis, por um preço próximo ao preço *spot*. Para isso, as instâncias *on demand* (sobressalente) e *spot* são criadas em um

pool de instâncias, e várias instâncias sobressalentes são mantidas para a redução dos custos de migração.

Três políticas de alocação são formuladas, uma priorizando a redução de custo, outra o aumento da disponibilidade e uma terceira política que reduz as revogações simultâneas ao alocar instâncias em vários *pools*, permitindo que uma instância seja usada por um ou mais usuários simultaneamente. Os resultados mostram que usar mais de um *pool* contribui para reduzir o preço total pago. No entanto, a introdução de uma camada de virtualização adicional entre usuário e provedor pode levar a sobrecarga de processamento não desprezível e possíveis problemas dependência tecnológica.

6.1.17 Shastri e Irwin (2018)

O objetivo do trabalho de Shastri e Irwin [38] consiste em identificar tendências de longo prazo para o mercado *spot*. Os autores afirmam que uma única distribuição não é capaz de modelar todo o mercado *spot* e propõem uma métrica agregada (índice de preço) para toda uma família de instâncias ou região. Esta métrica é baseada na variação de preços da instância combinada com o número de núcleos virtuais e capacidade de memória. Para selecionar uma instância, é calculado o valor de ganho financeiro a partir do preço definido pelo índice em relação ao preço real de uma única VM.

O trabalho também explora possibilidade de migrações de instâncias, considerando aspectos como a diminuição de custos, o aumento de disponibilidade ou requisitos balanceados. No último caso, a variabilidade de preço e a disponibilidade da instância são combinadas usando o índice de Sharpe [122], que é uma métrica financeira usada para o entendimento do retorno de um investimento em relação ao seu risco. Os resultados de testes com aplicações de longa duração (até 6 meses), utilizando tipos de instâncias *m4.large*, *m4.2xlarge*, *c4.2xlarge* e *r4.xlarge*, em diversas regiões e zonas de disponibilidade, no decorrer do ano de 2017, mostram que a abordagem pode atingir disponibilidade de até 99% pelo preço médio da instância *spot* no período.

6.1.18 Wang e co-autores (2018)

O trabalho de Wang e co-autores [39] propõe preditores de preço para o mercado *spot* da Amazon com base em um conjunto de características relacionadas à utilização do tipo de instância, como tempo de vida médio (*up time*), preço médio, revogações simultâneas (instâncias que podem ser encerradas simultaneamente devido a falhas de lance coincidentes) e tempo de instância para iniciar (*time to start*).

A dimensão temporal (janela deslizante de curto prazo) e a dimensão espacial (região e zona de disponibilidade) são consideradas na definição dos preditores. O modelo se

baseia no problema de otimização que visa minimizar o custo de execução da aplicação, considerando penalidades no caso de falhas de valores de lances abaixo do preço *spot*. Usando os tipos de instância *m3.2xlarge*, *c3.4xlarge*, *r3.xlarge* e *r3.2xlarge*, em quatro regiões, ao longo do período de março a junho de 2017, os resultados mostram menos falhas de lance com preços ligeiramente mais altos do que uma abordagem ótima, em que os lances acompanhariam os valores exatos da variação de preço *spot*.

6.1.19 Wolski (2018), Baughman e co-autores (2019)

Nos trabalhos de Wolski e co-autores [123] e Baughman e co-autores [124], a mudança no mecanismo de precificação de instâncias *spot* da Amazon é estudada. Após esta alteração, ocorrida em novembro de 2017, verificou-se uma diminuição significativa do número de variações de preços, efeito conhecido como “suavização de preços”.

Outra mudança significativa reside no fato de que o valor de lance mínimo do usuário pela instância desejada passa a ser opcional, ainda que os preços sejam variáveis e definidos com base em um modelo de mercado suavizado. Os autores concluem que a previsão de lances para instâncias *spot* continua sendo útil. No entanto, como a sistemática de funcionamento do mercado *spot* foi alterada, as técnicas propostas antes da mudança devem ser avaliadas no novo mercado e, possivelmente, adaptadas a ele.

Os trabalhos comparam histórico de preços entre os períodos de agosto a novembro de 2017 (antes da mudança) e de dezembro de 2017 a fevereiro de 2018 (após a mudança). Os autores concluem também que a mudança causou um aumento não desprezível nos preços médios e uma redução na confiabilidade das instâncias *spot*. Além disso, os autores revisam técnicas probabilísticas para previsão de preços e disponibilidade de instâncias, como a proposta analisada na Seção 6.1.13.

6.1.20 Liu e co-autores (2019)

Um algoritmo de previsão de preços *spot* associado a um tempo mínimo de disponibilidade da instância é proposto por Liu e co-autores [40]. Os preços previstos são obtidos por uma técnica ARIMA (Seção 4.2) para selecionar recursos baratos e estáveis. O trabalho também se concentra em obter um intervalo mínimo de tempo antes da revogação para evitar a migração frequente de instâncias.

Assim sendo, foram considerados nos testes variações de preços durante o período de abril a julho de 2016, ou seja, antes da mudança no modelo *spot*. Os resultados mostraram que a previsão de preços *spot* e a suposição de intervalo mínimo de disponibilidade podem ser objetivos complementares, podendo alcançar economias de 5% a 20% em relação ao

modelo de precificação sob demanda, considerando 12 tipos de instância na região leste dos Estados Unidos.

6.1.21 Lu e co-autores (2019)

No trabalho de Lu e co-autores [125], é analisado o problema de previsão do preço de reciclagem de recursos renováveis. É um problema típico de previsão com séries temporais (Seção 4.2), pois o preço tende a ser instável e não linear, com tendência de longo prazo e volatilidade de curto prazo, simultaneamente.

Os autores propõem um modelo de análise de tendências utilizando séries temporais e redes neurais recorrentes. A abordagem divide o problema em duas partes: (a) análise de tendência, que fornece uma estimativa do preço usando médias móveis (Seção 4.3), e (b) previsão de preço e análise de erro residual, que usa uma rede LSTM (Seção 5.3.2) para melhorar a solução. A rede utilizada uma única camada LSTM com 16 nós e o algoritmo ADAM (Seção 5.2.6).

Os resultados mostram erro RMSE 72,19% menor em comparação com o uso da técnica *Support Vector Regression* (SVR) e 76,80% menor do que a rede neural LSTM de nó único, para a previsão diária do preço da sucata durante o período de agosto de 2015 a abril de 2019 em Suzhou, China.

6.1.22 Mishra e co-autores (2019)

No trabalho de Mishra e co-autores [41], uma abordagem de série temporal (Seção 4.2) combinada com análise probabilística é usada para previsão de curto prazo (em segundos ou minutos) de preços *spot*. O algoritmo proposto considera as probabilidades de transição de preço, a partir do histórico de preços observado, para prever o próximo preço.

O trabalho possui seu foco na minimização do *Mean Absolute Percent Error* (MAPE). Dados históricos de 6 tipos de instância de diferentes regiões, de agosto a outubro de 2017, são usados para avaliar a técnica, com MAPE igual a 7,72% para o melhor caso.

6.1.23 Baig e co-autores (2020)

No artigo [45], Baig e co-autores propõem um método para identificar dinamicamente o melhor tamanho de janela deslizante (Seção 4.3) para treinar o modelo de predição para estimar a utilização de recursos de um *data center*. Eles aplicam uma abordagem de aprendizado com base em RNNs (Seção 5.3) para selecionar de forma adaptativa a melhor janela de observação e minimizar o erro da estimativa.

A configuração da rede possui 4 camadas ocultas com 23, 15, 10 e 5 nós, com ReLU (Seção 5.3.1) como função de ativação, ADAM (Seção 5.2.6) como algoritmo de otimização

e MSE como métrica de erro. O conjunto original de dados é dividido em subconjuntos com 80% e 20% de tamanho para fins de treinamento e avaliação, respectivamente. São realizadas 500 épocas de treinamento. Os experimentos mostraram que as janelas adaptáveis, variando de 5 a 30 observações anteriores, melhoraram a precisão da previsão de utilização dos recursos da CPU de 16% para 54%, produzindo melhores resultados em comparação com janelas fixas e técnicas como regressão linear/polinomial, SVR e outras técnicas de previsão.

6.1.24 Khandelwal e co-autores (2020)

Na abordagem utilizada por Khandelwal e co-autores [46], é apresentado um modelo de previsão de preços *spot* para auxiliar os usuários de computação em nuvem na definição de lances. O estudo compara a precisão de abordagens de aprendizado de máquina, como SVR, RNN (Seção 5.3) e *Regression Random Forest* (RRF).

Na análise foram considerados 10 tipos de instâncias otimizadas para computação, em 8 regiões geográficas diferentes, no período de abril de 2015 a março de 2016, para avaliação de previsões de um dia e uma semana de antecedência. A abordagem usa conjuntos de dados históricos de preços de 7 dias para o treinamento.

Os resultados mostram que RRF apresenta melhor precisão e menor tempo de treinamento do que outras abordagens, com $MAPE \leq 10\%$ para um dia e $MAPE \leq 15\%$ para previsões de uma semana de antecedência. Como aspecto negativo, incide o fato de que as análises ficaram restritas a instâncias de um único propósito de utilização (otimizadas apenas para computação), além considerar tipos de instâncias de gerações antigas e um período de tempo anterior à mudança no mercado *spot*.

6.2 Análise Comparativa

Nesta seção, são apresentados, resumidamente, objetivos, técnicas e parâmetros utilizados pelos mecanismos de precificação, com vistas em comparar os diversos trabalhos analisados na Seção 6.1. Na primeira coluna da Tabela 6.1, são relacionados os trabalhos publicados e os anos de publicação, sendo que foram analisados somente trabalhos a partir de 2011, ano em que o mecanismo de precificação *spot* da Amazon foi proposto como um mecanismo de mercado, conforme mencionado em [121].

Em seguida, na segunda coluna da Tabela 6.1, são mostrados quais mecanismos de precificação foram abordados pelo trabalho em questão, se *on demand* ou *spot*, ou se ambos os mecanismos. Os principais objetivos de cada trabalho são apresentados na terceira coluna. A quarta coluna exhibe uma classificação das técnicas utilizadas, objetivando agrupar os trabalhos que usam técnicas estatísticas (EST), modelos econômicos (ECO),

machine learning (ML) ou outras (OUT). As técnicas propriamente ditas de cada trabalho são apresentadas na quinta coluna. Na sexta coluna são exibidos os parâmetros de análise utilizados pelas técnicas de cada trabalho.

Na última coluna da Tabela 6.1, os trabalhos são classificados de acordo com o comportamento do mercado *spot* considerado, sendo o tipo 1 referente ao mercado artificial (antes de 2011), tipo 2 referente ao mercado real (de 2011 até novembro de 2017) e tipo 3 referente ao mercado suavizado (de dezembro de 2017 em diante). Essa análise foi feita observando, principalmente, as datas dos históricos de precificação das instâncias analisadas em cada trabalho.

Inicialmente, em 2011 e 2012, são analisados trabalhos sobre precificação *on demand* e sua relação com parâmetros de infraestrutura. Normalmente, os parâmetros analisados possuem correlação com o preço em razão do tipo de instância disponibilizada para comercialização pelo provedor. Por exemplo, instâncias de propósito geral com quantidades expressivas de memória e processador são comercializadas com preços maiores do que instâncias com configurações mais simples. Outro fator que impacta no preço é utilização da instância para um propósito específico, como uso intensivo de memória ou CPU. Após o ano de 2017, houve uma concentração muito grande das pesquisas no modelo de precificação *spot*. Em relação aos trabalhos que abordam esse modelo de precificação, os principais parâmetros considerados são relacionados ao histórico de preços, que é disponibilizado pela Amazon para os últimos 3 meses (preços e datas).

Os trabalhos de Li e co-autores [121] (Seção 6.1.11) e de Lu e co-autores [125] (Seção 6.1.21) não são apresentados na Tabela 6.1, uma vez que o primeiro trata de uma pesquisa ampla sobre publicações relacionadas à precificação *spot* e o segundo não trata diretamente desse mercado. Os trabalhos de Wolski e co-autores [123] e Baughman e co-autores [124] (Seção 6.1.19) também não são apresentados na Tabela 6.1, uma vez que tratam da mudança de preços do modelo *spot* ocorrida no final do ano de 2017.

As técnicas aplicadas nos trabalhos variam muito. Elas compreendem probabilidade e estatística, modelagem de leilões, otimização combinatória, séries temporais, redes neurais artificiais e outras técnicas de *machine learning*. Diversos trabalhos utilizam análise estatística combinada com outras técnicas. A modelagem de séries temporais e o uso de redes neurais, principalmente redes LSTM, são as técnicas predominantes para a previsão de preços *spot* recentemente. As técnicas utilizadas nesta Tese (estatística, médias móveis com janelas deslisantes e redes neurais LSTM) são, certamente, as mais utilizadas no cenário de pesquisa sobre precificação em computação em nuvem.

A análise do mercado *spot* (última coluna da Tabela 6.1) aponta que maioria dos trabalhos refere-se ao modelo de mercado puro (fase 2). Alguns trabalhos recentes ainda usam dados desse modelo de mercado. Apenas o trabalho de Al-Theibat e co-autores

(Seção 6.1.14) efetivamente analisa histórico de precificação de instâncias *spot* após 2017, ou seja, no modelo de mercado *spot* suavizado da Amazon. Esse trabalho se destaca também pela comparação de técnicas estatísticas e de *machine learning* aplicadas ao problema de previsão de preços no mercado *spot*.

Apesar de não tratar especificamente o mercado *spot* da Amazon, outro trabalho que também se destaca pela utilização da técnica de médias móveis combinada com redes neurais LSTM é o de Lu e co-autores (Seção 6.1.21). O objetivo de uso da rede neural é bastante diferente das demais abordagens e não trata da previsão de preços, mas sim da análise residual (erro) e de sua influência.

Ao nosso conhecimento, não há trabalho semelhante na literatura que usa as técnicas estatísticas, econômicas e de *machine learning* de forma complementar e em momentos diferentes, ou seja, para identificação de tendências e sugestão de preço máximo, com o objetivo de dar ao usuário de computação em nuvem os subsídios necessários para análise de custos e disponibilidade de instâncias *spot* da Amazon.

Tabela 6.1: Comparação entre os trabalhos sobre precificação em computação em nuvem.

| Ref. (Ano) | Precificação | Objetivo | Grupo | Técnica | Parâmetros | Mercado |
|-------------|--------------------------|--|----------|---|----------------------------------|---------|
| [26] (2011) | <i>on demand</i> | escalonamento de tarefas em nuvem IaaS | OUT | <i>delay scheduling, progress share</i> | preço, memória RAM | 1 |
| [27] (2012) | <i>on demand</i> | seleção de instâncias virtuais de acordo com requisitos da aplicação | OUT | <i>ontology reasoning</i> | CPU, rede, preço, armazenamento | 1 |
| [30] (2013) | <i>spot</i> | modelagem do mercado <i>spot</i> | EST | análise de série temporal, simulação, distribuição de Pareto | preço, data | 2 |
| [29] (2013) | <i>spot</i> | modelagem do mercado <i>spot</i> | EST | análise de série temporal, análise estatística (MoG), modelagem de mercado | preço, data | 2 |
| [42] (2014) | extensível a <i>spot</i> | seleção de serviços com base em restrições de custo e qualidade | ECO | mecanismo de mercado baseado em leilões VCG | qualidade, preço, data | - |
| [28] (2015) | <i>on demand, spot</i> | leilão por instâncias <i>spot</i> | ECO | análise de série temporal, e modelagem de mercado, | preço, data | 2 |
| [31] (2015) | <i>spot</i> | estratégias de leilão por instâncias <i>spot</i> | EST | análise de série temporal, análise estatística (lognormal) | preço, data | 2 |
| [32] (2015) | <i>spot</i> | leilão por instâncias <i>spot</i> | ML | análise de série temporal, algoritmo de otimização <i>gradient descending</i> | preço, data | 2 |
| [33] (2015) | <i>on demand, spot</i> | escalonamento de instâncias com restrições | EST | análise de série temporal, médias móveis (simples, ponderada e exponencial) | preço, data, CPU, memória RAM | 2 |
| [34] (2016) | <i>on demand, spot</i> | modelagem do mercado <i>spot</i> | ECO | simulação e monitoramento de mercado | preço, data | 2 |
| [44] (2017) | <i>spot</i> | previsão de preços <i>spot</i> | ML | redes LSTM | preço, data | 2 |
| [35] (2017) | <i>spot</i> | estimativa de lance e disponibilidade | EST | série temporal, probabilidade e estatística | preço, data | 2 |
| [36] (2018) | <i>spot</i> | previsão de preços <i>spot</i> | ML | redes LSTM | preço, data | 3 |
| [25] (2018) | <i>spot</i> | previsão de preços <i>spot</i> | ML | redes LSTM | preço, data | 2 |
| [37] (2018) | <i>spot</i> | garantia de disponibilidade | EST | paralelismo, redundância e tolerância a falhas | preço, data | 2 |
| [38] (2018) | <i>spot</i> | identificação de tendências | ECO | indicadores financeiros e modelagem de riscos | preço, data | 2 |
| [39] (2018) | <i>spot</i> | previsão de preços <i>spot</i> | ECO | modelagem de série temporal | preço, data, <i>start/uptime</i> | 2 |
| [40] (2019) | <i>spot</i> | previsão de preços <i>spot</i> | EST | análise estatística e série temporal | preço, data | 2 |
| [41] (2019) | <i>spot</i> | previsão de preços <i>spot</i> | EST | análise estatística e série temporal | preço, data | 2 |
| [45] (2020) | extensível a <i>spot</i> | utilização de infraestrutura | EST & ML | análise estatística e redes LSTM | histórico de uso | - |
| [46] (2020) | <i>spot</i> | previsão de preços <i>spot</i> | ML | RRF | preço, data | 2 |

Parte II

Contribuições

Capítulo 7

Modelagem Estatística de Precificação *On demand* e *Spot*

A contribuição inicial da Tese consiste na modelagem estatística das precificações *on demand* e *spot*, que são utilizadas por provedores de computação em nuvem pública para provimento de instâncias de máquinas virtuais. A primeira modelagem teve como objetivo determinar a influência das características de processador e memória na composição do preço *on demand*, considerando diferentes tipos de instância dos provedores AWS (Seção 3.2.1) e GCP (Seção 3.3.1). A segunda modelagem objetivou determinar uma faixa razoável de lances para instâncias disponíveis no modelo de precificação dinâmica *spot* da AWS (Seção 3.2.3), de modo que o usuário de computação em nuvem conseguisse estimar a disponibilidade da instância em relação ao seu custo previsto. Nesse estudo, foram utilizados dados de 2016 e, portanto, o modelo de mercado real foi considerado

O restante deste capítulo possui a seguinte estrutura. A Seção 7.1 apresenta a modelagem estatística da precificação *on demand*. A modelagem da precificação *spot* é detalhada na Seção 7.2. Ao final deste capítulo, na Seção 7.3, são apresentadas as considerações finais sobre as modelagens estatísticas realizadas.

7.1 Modelagem da Precificação *On demand*

Conforme discutido na Seção 3.1.1, a precificação *on demand* é considerada estática, uma vez que as instâncias são comercializadas a um preço fixo, dependendo do tipo de instância. O custo final para o usuário considera esse preço e o tempo de utilização da instância. O usuário tem a garantia de que a instância não será revogada pelo provedor, ou seja, a instância é permanente, o que torna o seu custo elevado se comparado com outras estratégias de precificação.

7.1.1 Modelagem Estatística Aplicada

Inicialmente, é analisada a relação entre custo, número de núcleos (ou *cores*) de processamento e quantidade de memória dos diversos tipos de instâncias *on demand* oferecidas por dois provedores de nuvem pública, de maneira a gerar informações que auxiliem tanto os usuários na seleção do tipo da instância, quanto os provedores de nuvem no estabelecimento do custo de novos tipos de instância.

Do ponto de vista do provedor de nuvem, o estabelecimento da relação na Equação 7.1 busca contribuir para o aumento da eficiência na utilização dos recursos de infraestrutura, assim como para a evolução dos modelos econômicos e de precificação de recursos. Na perspectiva do usuário, o modelo pode ser utilizado em estratégias eficientes de seleção de tipos de instâncias e de modelos de precificação disponíveis, a depender das características de suas aplicações.

Como segunda contribuição, o trabalho tem como objetivo identificar quais outros fatores podem influenciar nesta relação, e se diferentes provedores praticam estratégias semelhantes de precificação para seus recursos. As técnicas de análise quantitativa [86] aplicadas são regressão multilinear, análise de variância com intervalo de confiança e clusterização. O objetivo consiste em identificar a relação entre o custo de uma instância *on demand*, a quantidade de núcleos do processador e o tamanho da memória. Nossa primeira hipótese considerou essa relação como multilinear, usando dados sobre o processador e a memória, conforme a Equação 7.1.

$$Custo = \alpha + \beta * Processador + \gamma * Memória \quad (7.1)$$

Na Equação 7.1, a variável *Custo* é o valor monetário (em \$/hora); *Processador* é a quantidade de núcleos de processamento (*virtual cores*); *Memória* é o tamanho da memória (em GB); α é o coeficiente linear da equação; β é o coeficiente angular do fator Processador; γ é o coeficiente angular do fator Memória. Nos experimentos da Seção 7.1.2 foram exploradas as várias regressões multilineares, visando obter valores apropriados para α , β e γ , bem como novas equações.

7.1.2 Dados Utilizados e Experimentos Realizados

Os dados coletados se referem a medições de custos de utilização de instâncias *on demand* de computação em nuvem, sendo que cada tipo de instância possui uma configuração específica de processamento e memória. Para isso, foram coletados dados dos provedores AWS e GCP, originários de 2 fontes distintas. A primeira origem contém dados utilizados na pesquisa referenciada em [16], coletados em 2014. A segunda origem contém dados do serviço EC2 da AWS [17], coletados em junho de 2016.

Os experimentos foram planejados de acordo com a origem dos dados, técnica de análise e provedor de computação em nuvem. Em todos os experimentos, foram considerados os custos no modelo de precificação *on demand* para instâncias disponíveis na costa leste dos Estados Unidos (US East), utilizando o sistema operacional Linux. Na Tabela 7.1 são apresentados os experimentos realizados. Foram aplicadas as técnicas de regressão multilinear, além da análise de variância com grau de confiança de 90%, tendo como objetivo a avaliação da relevância estatística do coeficiente linear α , e dos coeficientes angulares β (núcleos de processamento) e γ (quantidade de memória). A técnica de clusterização foi utilizada com a finalidade de compor grupos representativos de dados. No caso, foi utilizado o algoritmo *Minimal Spanning Tree*, considerando a distância euclidiana mínima entre os pontos que farão parte de um grupo [86].

Tabela 7.1: Experimentos com as instâncias *on demand* dos provedores AWS e GCP.

| Id | Técnica Aplicada | Descrição do Experimento | Origem dos Dados |
|----------------------------|--|---|------------------|
| mlr | Regressão Multilinear (MLR) | Custo (USD/hora) em função de quantidade de núcleos de CPU e Memória (GB). | AWS e GCP [16] |
| c-spt | Clusterização (C) com filtro <i>spanning tree</i> (SPT) | Custo (USD/hora) em função de quantidade de núcleos de CPU e Memória (GB), após aplicação do algoritmo <i>Minimal Spanning Tree</i> . | AWS [16] |
| mlr-spt | Regressão Multilinear (MLR) com agrupamento <i>spanning tree</i> | Custo (USD/hora) em função de quantidade de núcleos de CPU e Memória (GB), agrupando-se as instâncias com o algoritmo <i>Minimal Spanning Tree</i> | AWS [16] |
| mlr-ecu | Regressão Multilinear (MLR) | Custo (USD/hora) em função de quantidade de núcleos de CPU, Memória (GB) e ECU. | AWS [17] |
| mlr-ecu-restricted | Regressão Multilinear (MLR) | Custo (USD/hora) em função de quantidade de núcleos de CPU, Memória (GB) e ECU, somente instâncias <i>compute-optimized</i> e <i>memory-optimized</i> . | AWS [17] |
| mlr-ecu-restricted-no-core | Regressão Multilinear (MLR) | Custo (USD/hora) em função de quantidade de Memória (GB) e ECU, somente instâncias <i>compute-optimized</i> e <i>memory-optimized</i> . | AWS [17] |

Os resultados são mostrados no formato de quadro-resumo, que fornece a identificação, a descrição e as observações sobre o experimento. Além disso, são mostrados também a equação da regressão multilinear, o coeficiente de determinação e os dados relacionados à relevância estatística obtidos por meio da análise de variância e da aplicação do *F-Test* [86]. Para análise visual, são mostrados 2 tipos de gráfico: Quantil Normal X Residual (erro) e Homocedasticidade. O primeiro gráfico deve se assemelhar a uma reta, de forma a demonstrar que existe uma relação linear entre os parâmetros. O segundo gráfico evidencia a dispersão dos pontos da regressão dentro de bons limites.

Regressão Multilinear (MLR)

A Figura 7.1 mostra o quadro-resumo com os resultados da regressão multilinear *mlr*, que define Custo (USD/hora) em função de quantidade de núcleos de CPU (*virtual cores*) e Memória (GB), para tipos de instâncias da Amazon EC2 coletados em [16].

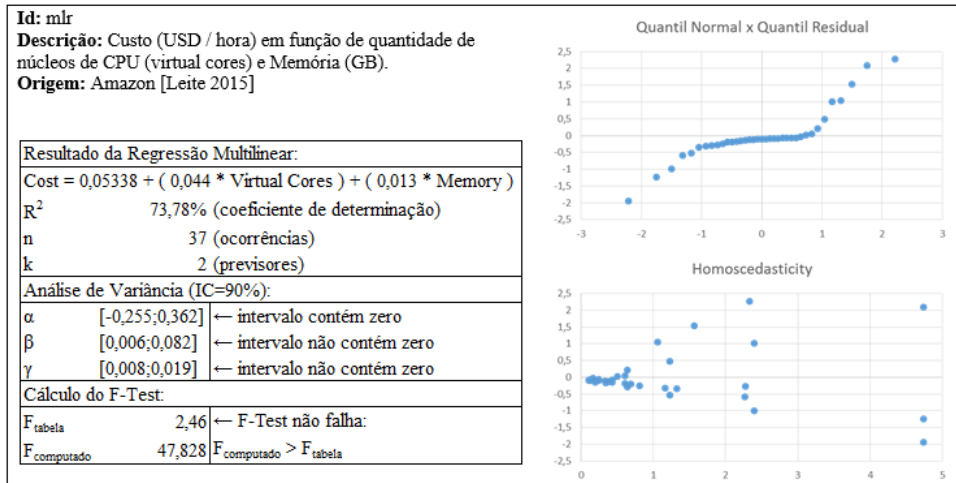


Figura 7.1: Resumo do experimento *mlr* com dados da Amazon EC2 coletados em [16].

Neste experimento, o coeficiente de determinação (R^2) é igual a 73,78%. O coeficiente linear α não é estatisticamente relevante na análise de variância, pois contém valor zero no intervalo, ou seja, não se pode afirmar com 90% de confiança que o coeficiente representa o comportamento linear esperado. Outro problema apresentado é o gráfico de homoscedasticidade, que apresenta uma tendência de aglomeração no formato de cone horizontal, sugerindo a necessidade de uma transformação dos dados (e.g. logarítmica). O gráfico dos quantis evidencia que a relação não é linear. Apesar de passar no *F-Test*, conclui-se que a regressão possui variáveis externas não consideradas. Dessa forma, o custo de um tipo de instância não deve ser calculado considerando apenas número de núcleos e quantidade de memória para o serviço Amazon EC2.

Análise semelhante foi realizada com dados do Google, conforme mostrado na Figura 7.2. Os resultados obtidos para as instâncias Google GCP foram melhores, do ponto de vista estatístico, do que os obtidos para as instâncias Amazon EC2, principalmente em relação ao coeficiente de determinação. Entretanto, permanecem os problemas relacionados à relevância estatística do coeficiente linear α e ao gráfico de homoscedasticidade. Sendo assim, não é possível afirmar que há uma relação de linearidade entre os fatores para se estimar custos de instâncias nesse provedor.

Clusterização *Minimal Spanning Tree* (MLR-SPT)

Com o objetivo de estruturar melhor os fatores, em seguida, foi realizada uma caracterização com base nos tipos de instâncias da Amazon EC2. Para tanto, foi aplicado o algoritmo de clusterização *Minimal Spanning Tree* no conjunto de 37 tipos de instâncias. A Tabela 7.2 mostra os dados dos centroides obtidos para cada um dos 6 grupos representativos e a Figura 7.3 exibe o dendograma resultante dos agrupamentos.

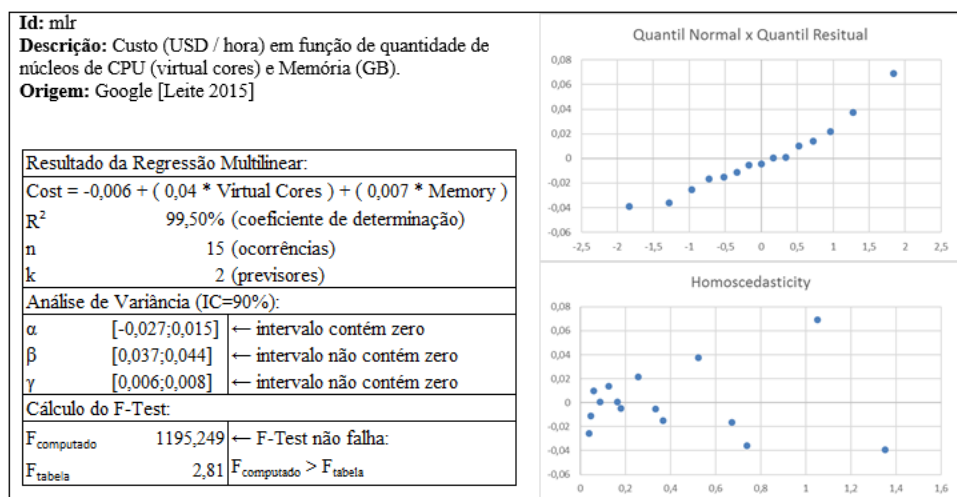


Figura 7.2: Resumo do experimento *mlr* com dados do Google GCP coletados em [16].

Tabela 7.2: Grupos de Instâncias Amazon EC2 após aplicação do algoritmo de clusterização *Minimal Spanning Tree*.

| Cluster | Tipos de Instâncias | Virtual Cores | Memória (GB) | Custo / Hora |
|---------|--|---------------|--------------|--------------|
| A | c3.8xlarge,cc2.8xlarge | 32,00 | 60,25 | 1,84 |
| B | i2.8xlarge,r3.8xlarge,cr1.8xlarge | 32,00 | 244,00 | 4,99 |
| C | hs1.8xlarge,r3.4xlarge,i2.4xlarge | 16,00 | 119,50 | 3,50 |
| D | hi1.4xlarge,m2.4xlarge,r3.2xlarge,i2.2xlarge | 12,00 | 62,60 | 2,10 |
| E | c3.2xlarge,g2.2xlarge,m2.xlarge,r3.large,m3.xlarge, m1.xlarge,m3.medium,m1.medium,c3.large,t2.medium, t2.micro,t1.micro,c1.medium,t2.small,m1.small, c1.xlarge,c3.xlarge,m3.large,m1.large | 4,34 | 10,21 | 0,30 |
| F | m3.2xlarge,m2.2xlarge,r3.xlarge,i2.xlarge, c3.4xlarge,cg1.4xlarge | 11,00 | 28,71 | 1,01 |

Com base nas configurações resultantes da aplicação do algoritmo de clusterização e exibidas na Tabela 7.2, buscou-se então analisar uma possível relação de linearidade entre os fatores. Os valores de *virtual cores*, memória e custo/hora são referentes às médias aritméticas dos tipos de instância pertencentes a cada grupo.

Os resultados da Figura 7.4 não são estatisticamente relevantes, uma vez que o coeficiente linear α e o coeficiente angular β apresentam o valor zero em seus intervalos de confiança. Além disso, não há como determinar homocedasticidade dos dados, por serem poucos pontos no gráfico, apesar de haver uma melhora significativa do coeficiente de determinação em comparação com o experimento *mlr* sem uso de clusterização. Apesar disso, não é possível uma avaliação conclusiva, não sendo recomendável o uso desta regressão para se estimar custos de instâncias.

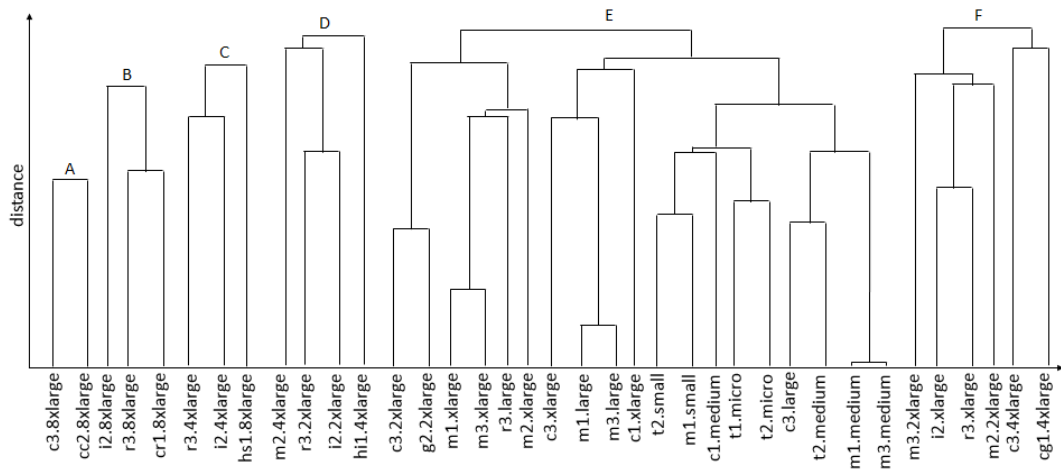


Figura 7.3: Dendrograma de representação dos grupos em função da distância entre instâncias, após aplicação do algoritmo de clusterização *Minimal Spanning Tree*.

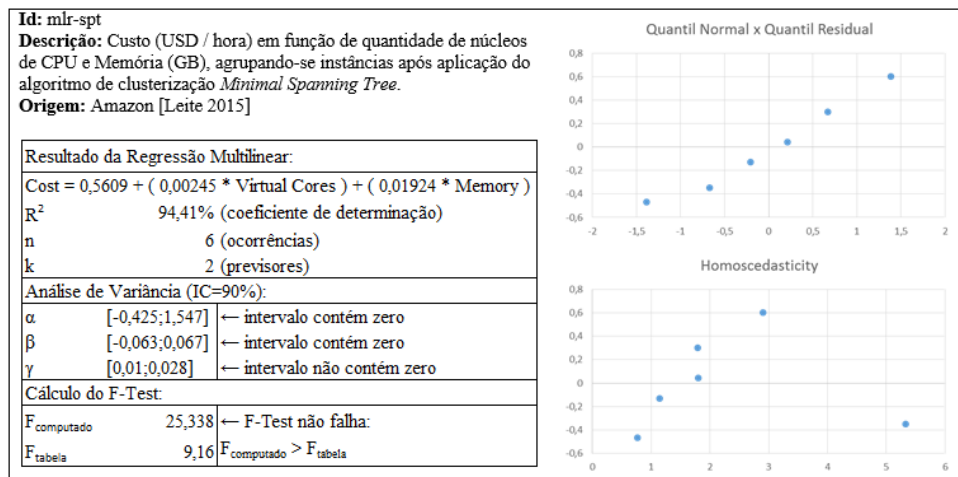


Figura 7.4: Resumo do experimento *mlr-spt* com dados da Amazon EC2 coletados em [16], após aplicação do algoritmo de clusterização *Minimal Spanning Tree*.

Regressão Multilinear com ECU e vCPU (MLR-ECU-Restricted)

Melhores resultados foram obtidos nos experimentos que utilizaram os dados mais recentes da Amazon EC2, coletados de [17]. O experimento *mlr-ecu-restricted* aplica a técnica de regressão multilinear, considerando um subconjunto de 19 tipos de instâncias cujo propósito é processamento ou memória (*compute-optimized* e *memory-optimized*), ou seja, instâncias do tipo *c.** ou *m.**, respectivamente. Neste subconjunto, fica clara a alta influência das quantidades de núcleos de CPU e de memória na composição do custo por meio do valor do coeficiente de determinação R^2 obtido.

Outro diferencial do experimento *mlr-ecu-restricted* foi o uso de mais um previsor, a

medida de ECU (*EC2 Compute Unit*), no qual 1 ECU é equivalente à capacidade de um processador Opteron 2007 ou Xeon 2007, com velocidade de 1,0 a 1,2 GHz [17]. Dessa forma, a Equação 7.1 é modificada para considerar o valor do previsor escalar ECU, cujo coeficiente é θ , sendo assim definida a Equação 7.2.

$$Custo = \alpha + \beta * Processador + \gamma * Memória + \theta * ECU \quad (7.2)$$

A Figura 7.5 mostra os resultados obtidos no experimento *mlr-ecu-restricted*. Apesar do coeficiente de determinação R^2 obtido ter sido alto (99,8%) e da significância estatística de todos os fatores, o gráfico da homocedasticidade contém uma tendência de não espalhamento. A Equação 7.3 apresenta a regressão multilinear obtida no experimento *mlr-ecu-restricted*. É importante destacar o maior peso do coeficiente β , referente ao fator processador, em relação aos demais coeficientes. Isso indica a maior importância deste fator na composição do custo das instâncias.

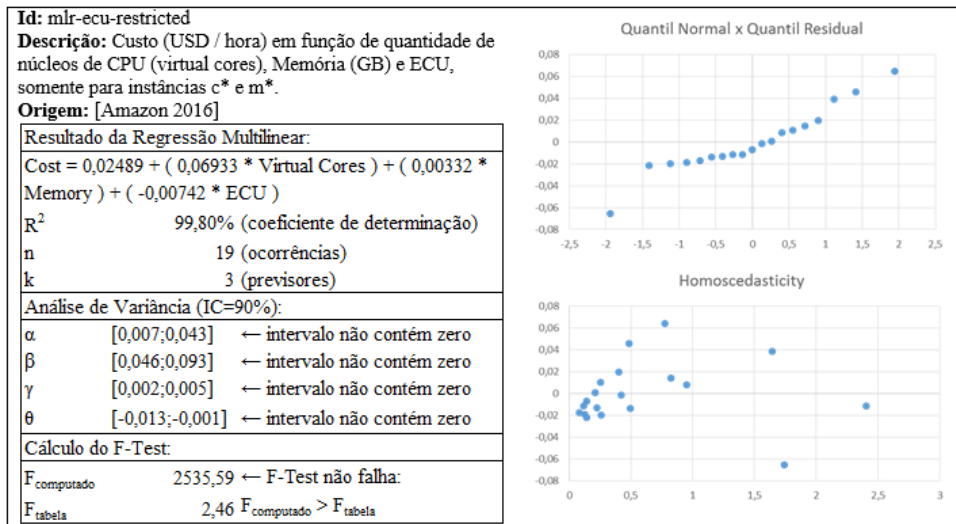


Figura 7.5: Resumo do experimento *mlr-ecu-restricted* com dados da Amazon EC2 de [17] com uso da métrica ECU.

$$Custo = 0,02498 + 0,0633 * Processador + 0,00332 * Memória - 0,00742 * ECU \quad (7.3)$$

Regressão Multilinear somente com ECU (MLR-Restricted-no-core)

Assim, buscando amenizar a tendência revelada no gráfico de homocedasticidade e considerando a alta correlação entre os fatores *Virtual Cores* e ECU, foi realizado o experimento

mrl-restricted-no-core com apenas os fatores *Memory* e *ECU*. Outro ponto considerado foi a característica de progressão geométrica de razão 2 no fator *ECU*, o que indica uma potencial necessidade de uso de uma transformação logarítmica. Além disso, foi também utilizada a transformação logarítmica na variável resposta *Custo*, obtendo-se a Equação 7.4. Os resultados do experimento podem ser vistos na Figura 7.6.

$$\ln(Custo) = \alpha + \gamma * Memória + \theta * \ln(ECU) \quad (7.4)$$

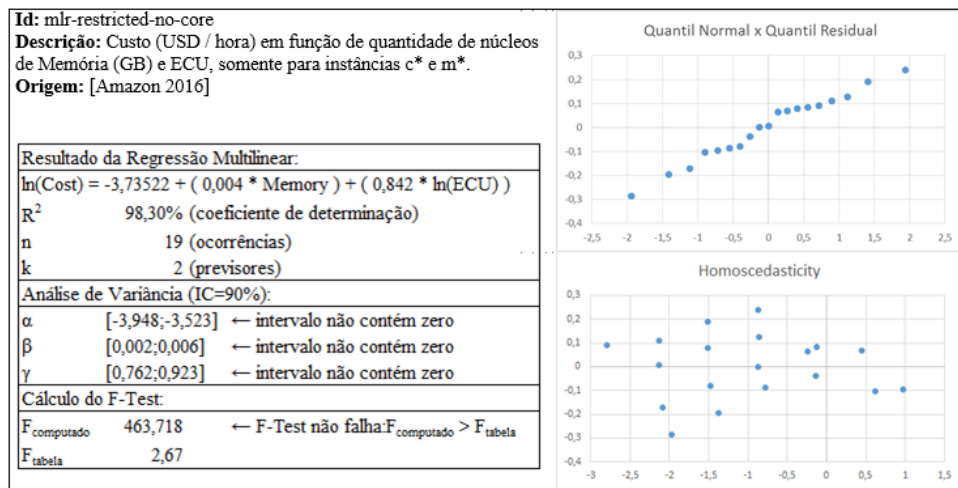


Figura 7.6: Resumo do experimento *mrl-estricted-no-core* com dados da Amazon EC2 de [17], desconsiderando o fator *Virtual Cores*.

Previsão com MLR-ECU-Restricted

Finalmente, foi aplicada a equação resultante do experimento *mrl-ecu-restricted* (Figura 7.5) em outros tipos de instâncias *memory optimized large* da Amazon EC2, para tentar determinar a aplicabilidade da mesma para tipos de instâncias com propósitos semelhantes ao da equação, ou seja, para uso de processador ou memória. Essa comprovação pode ser visualizada na Tabela 7.3, especificamente nas colunas que demonstram o erro (escalar e percentual), ou seja, a diferença entre o valor real cobrado pela Amazon e o valor estimado usando a equação. Como pode ser visto, a fórmula possui um erro pequeno (<5%) para a maior e menor instância consideradas (*x1.32xlarge* e *r3.large*), respectivamente, e erro razoavelmente pequeno (<15%) para as demais instâncias.

Tabela 7.3: Aplicação da equação de regressão multilinear do experimento *mlr-ecu-restricted* em tipos de instância Amazon EC2 *memory optimized*.

| Tipo da Instância | vCPU | ECU | Memória (GB) | On demand (\$/hora) | Estimado (\$/hora) | Erro (Escalar) | Erro (%) |
|-------------------|------|-----|--------------|---------------------|--------------------|----------------|----------|
| x1.32xlarge | 128 | 349 | 1952,0 | 13,338 | 12,79398 | 0,54402 | 4% |
| r3.large | 2 | 6,5 | 15,0 | 0,166 | 0,165142 | 0,00086 | 1% |
| r3.xlarge | 4 | 13 | 30,5 | 0,333 | 0,307052 | 0,02595 | 8% |
| r3.2xlarge | 8 | 26 | 61,0 | 0,665 | 0,58921 | 0,07579 | 11% |
| r3.4xlarge | 16 | 52 | 122,0 | 1,330 | 1,153528 | 0,17647 | 13% |
| x1.32xlarge | 32 | 104 | 244,0 | 2,660 | 2,282162 | 0,37784 | 14% |

7.2 Modelagem da Precificação *Spot*

A modelagem estatística da precificação *spot* objetiva a redução de custos para o usuário de computação em nuvem. Esta modelagem também possibilita uma estimativa da disponibilidade da instância em relação ao seu custo. O projeto de modelagem estatística e os dados utilizados nas análises são apresentados na Seção 7.2.1. As análises propriamente ditas são detalhadas na Seção 7.2.2.

7.2.1 Projeto de Modelagem Estatística

As instâncias de máquinas virtuais comercializadas no modelo de precificação *spot* da Amazon (Seção 3.2.3) possuem um preço dinâmico e determinado por um mecanismo de mercado. Tais instâncias podem ser revogadas pelo provedor, o que causa baixa disponibilidade do recurso computacional para o usuário. Para analisar os preços dinâmicos desse modelo de precificação, foi proposto o uso de técnicas de séries temporais (Seção 4.2) e médias móveis (Seção 4.3), considerando intervalos fixos de tempo de 12 horas, com o objetivo de proporcionar boa disponibilidade, mantendo a redução do custo para o usuário de computação em nuvem.

Na modelagem, foram utilizados dados referentes a três meses de histórico de preços (de setembro a novembro de 2016), de cinco tipos de instâncias *spot* da Amazon EC2. Os dados referem-se a instâncias localizadas na região leste dos Estados Unidos (*US-East*), com o sistema operacional Linux. Para os três meses, foram consideradas mais de 140.000 entradas para cada tipo de instância, totalizando 723.276 ocorrências de variações de preço.

Dentre os tipos de instâncias analisadas, há duas instâncias de propósito geral (t1.micro e m4.10xlarge), uma instância otimizada para uso de CPU (c3.8xlarge), uma instância otimizada para memória (r3.2xlarge) e uma instância otimizada para GPU (g2.2xlarge). Portanto, a análise considerou a influência do propósito de utilização da instância na formulação da estratégia de precificação. A Tabela 7.4 mostra os tipos de instâncias *spot*

analisadas, com os preços *on demand* e de reserva, referentes ao mês de novembro de 2016, para fins de comparação.

Tabela 7.4: Tipos de instâncias Amazon EC2 analisadas.

| Tipo de Instância | Propósito | vCPU | ECU | Memória (GB) | Preço On Demand (USD/hora) | Preço Reserved (USD/hora) |
|-------------------|-------------------|------|-------|--------------|----------------------------|---------------------------|
| c3.8xlarge | Compute Optimized | 32 | 108,0 | 60,0 | 1,680 | 1,168 |
| g2.2xlarge | GPU Instance | 8 | 26,0 | 15,0 | 0,650 | 0,474 |
| m4.10xlarge | General Purpose | 40 | 124,5 | 160,0 | 2,394 | 1,645 |
| r3.2xlarge | Memory Optimized | 8 | 26,0 | 61,0 | 0,665 | 0,418 |
| t1.micro | General Purpose | 1 | 2,0 | 0,6 | 0,020 | N/A |

A primeira tentativa de caracterizar o modelo de precificação *spot* consistiu em aplicar a mesma estratégia para instâncias *on demand*, conforme abordagem adotada na Seção 7.1. Tentou-se, então, obter equações de regressão multilinear a partir dos dados de histórico de variação de preços. As equações obtidas para a instância c3.8xlarge apresentaram coeficientes de determinação R^2 abaixo de 6%, o que é obviamente inaceitável. Portanto, optou-se por usar outra abordagem.

Os preços de instâncias *spot* em 2016 eram definidos por um modelo de mercado real (Seção 3.2.3) e, por esse motivo, a variação nos preços de uma instância era bastante complexa. Devido à ausência de uma tendência linear e à alta volatilidade dos preços ao final de 2016, a determinação de valores através da técnica de regressão multilinear não produz resultados satisfatórios. Além disso, como o modelo de mercado é utilizado, observa-se uma autocorrelação temporal dos valores, sugerindo que o preço *spot* em um dado momento depende de seus valores anteriores.

Assim, foi assumida a hipótese de que a análise de séries temporais (Seção 4.2) seria a melhor opção. Em seguida, foi aplicado um método de suavização da média móvel (Seção 4.3), como melhor estimativa para os preços de instância ao longo do período estudado. Neste trabalho, uma janela de 12 horas foi considerada para estimativa, ou seja, o valor previsto da instância *spot* é determinado por meio dos valores comercializados nas últimas 12 horas.

Sejam $X_{t1}, X_{t2}, \dots, X_{tm}$ variáveis aleatórias que representam os valores comercializados no intervalo de tempo $[t - 12h, t]$. Assumindo uma distribuição normal desses valores, a estimativa do valor comercializado no momento t e seu respectivo intervalo de confiança de 95% (95%CI) são dados, respectivamente, pelo conjunto de Equações em 7.5, onde s_t é o desvio padrão de amostra de valores no intervalo de tempo $[t - 12h, t]$. Para o valor 95%CI é considerado o quantil da distribuição z , ou seja, $z_{(1-\alpha/2)}$ de uma distribuição normal, (onde α é o nível de significância de 5%), assim $z_{(0,975)} = 1,96$. O preço máximo estimado no tempo t é definido como o limite superior do intervalo de confiança de 95% apresentado pelo conjunto de Equações em 7.5.

$$\begin{aligned}\bar{X}_t &= \frac{1}{m} \sum_{i=1}^m X_{ti} \\ 95\%CI_t &= [\bar{X}_t - 1.96 \times s_t; \bar{X}_t + 1.96 \times s_t] \\ s_t &= \sqrt{\frac{1}{m-1} \sum_{i=1}^m (X_{ti} - \bar{X}_t)^2}\end{aligned}\tag{7.5}$$

Dessa forma, foi utilizado o preço estimado \bar{X}_t para calcular a disponibilidade da instância *spot* durante o período de três meses. De um modo semelhante a Ben-Yehuda e co-autores [30] (Seção 6.1.3), a disponibilidade foi calculada como o número de preços *spot* abaixo (n_{below}) do preço máximo estimado para o período, dividido pelo número total de preços *spot* (n_{all}), conforme mostrado na Equação 7.6.

$$availability = \frac{n_{below}}{n_{all}}\tag{7.6}$$

7.2.2 Análises Realizadas

Neste estudo foram realizadas análises para os tipos de instância c3.8xlarge, g2.2xlarge, m4.10xlarge, r3.2xlarge e t1.micro, conforme seções a seguir.

Análise do Tipo de Instância c3.8xlarge

A Figura 7.7 mostra os históricos de preços *spot* observados (cruzes cinza claro) para setembro, outubro e novembro de 2016, considerando o tipo de instância c3.8xlarge. Além disso, são mostrados também os limites superior e inferior do intervalo de confiança com o nível de 95% (95%CI) nas linhas cinza escuras. A linha preta mostra o preço *spot* estimado deste tipo de instância específico em cada momento.

Por exemplo, suponha que o usuário quisesse estimar um lance para a instância c3.8xlarge às 00:00 de 1º de outubro de 2016. Os dados anteriores de 12 horas resultaram em uma média de 0,375 USD/hora e um 95%CI de [0,288; 0,462]. Assim, o preço máximo estimado para 12:00 de 1º de outubro de 2016 foi igual a 0,462 USD/hora e o preço observado foi de 0,331 USD/hora, garantindo a alocação de instância para o lance oferecido.

A Tabela 7.5 mostra a análise de disponibilidade da instância c3.8xlarge. Nessa tabela, são mostrados a média e o desvio padrão (DP) para cada mês (setembro, outubro e novembro) e para os dias da semana. Conforme essa tabela, em todas as segundas-feiras nesse período, a média do preço máximo estimado foi de 0,582 USD/hora, com um desvio padrão de 0,362. Assim, é possível notar ver que a melhor disponibilidade da

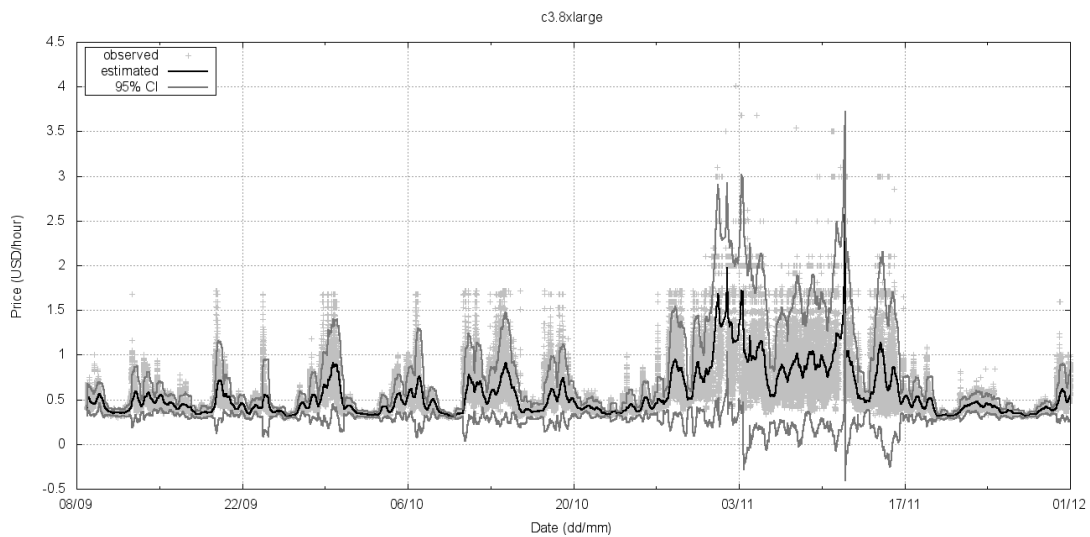


Figura 7.7: Preços observados e estimados para a instância c3.8xlarge durante os meses de setembro, outubro e novembro de 2016.

Tabela 7.5: Análise de custo e disponibilidade da instância c3.8xlarge.

| | | Estimativa de Preço Máximo em USD/hora Média (DP) | Disponibilidade |
|-----------------------------------|---------------|--|-----------------|
| Período completo (3 meses) | | 0,813 (0,467) | 91,3% |
| Mês | | | |
| | Setembro | 0,607 (0,260) | 90,0% |
| | Outubro | 0,726 (0,316) | 91,4% |
| | Novembro | 1,106 (0,637) | 92,2% |
| Dia da semana | | | |
| | Segunda-feira | 0,582 (0,362) | 82,5% |
| | Terça-feira | 0,834 (0,440) | 89,6% |
| | Quarta-feira | 0,917 (0,460) | 91,3% |
| | Quinta-feira | 0,830 (0,420) | 93,0% |
| | Sexta-feira | 0,900 (0,543) | 91,9% |
| | Sábado | 0,931 (0,551) | 98,9% |
| | Domingo | 0,629 (0,346) | 92,4% |

instância c3.8xlarge nesses três meses ocorre aos sábados, quando o preço do lance é de 0,931 USD/hora. Considerando que o preço *on demand* da instância c3.8xlarge é de 1,68 USD/hora, conforme a Tabela 7.4, se o cliente da nuvem utilizar o preço estimado pela nossa modelagem, ele gastará 48,39% do preço *on demand*, com disponibilidade muito alta (98,9%). Um bom *trade-off* entre preço e disponibilidade também ocorre aos domingos, quando o valor estimado de lance é de 0,629 USD/hora (37,44% do preço *on demand*), com uma disponibilidade superior a 92%.

Análise do Tipo de Instância g2.2xlarge

A Figura 7.8 mostra o preço *spot* real e estimado de setembro a novembro de 2016, considerando a instância g2.2xlarge. Nesse caso, há um grande número de picos no preço

real, resultando em uma grande variação de curto e longo prazo no preço *spot*. A estimativa de valor para essa instância é claramente mais complexa que a estimativa para a instância c3.8xlarge (Seção 7.2.2). Mesmo assim, a análise de média móvel suavizada conseguiu manter os valores estimados dentro de um intervalo 95%CI razoável.

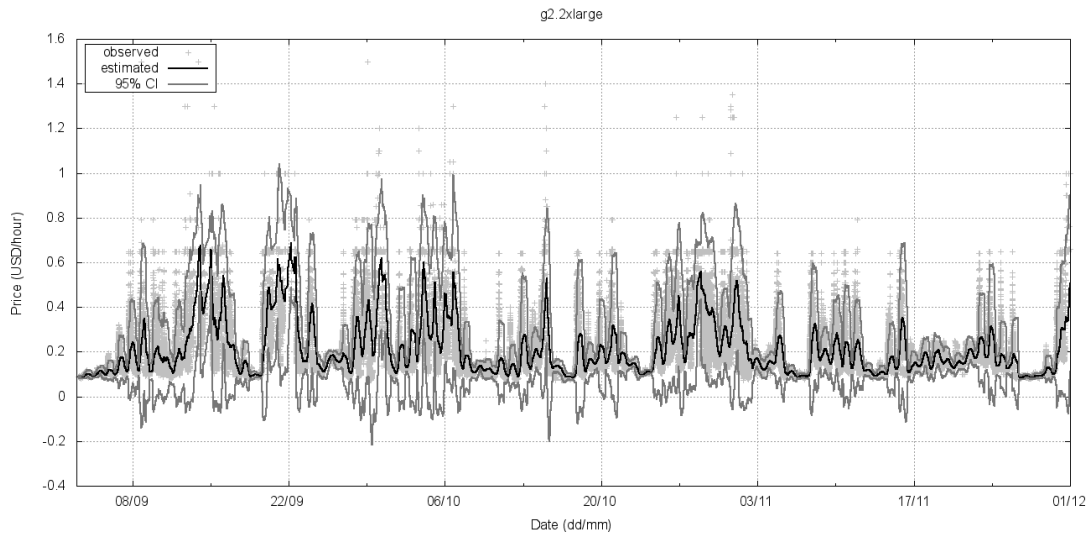


Figura 7.8: Preços observados e estimados para a instância g2.2xlarge durante os meses de setembro, outubro e novembro de 2016.

Tabela 7.6: Análise de custo e disponibilidade da instância g2.2xlarge.

| | | Estimativa de Preço Máximo em USD/hora | Disponibilidade |
|-----------------------------------|---------------|--|-----------------|
| | | Média (DP) | |
| Período completo (3 meses) | | 0,345 (0,211) | 87,6% |
| Mês | | | |
| | Setembro | 0,368 (0,236) | 87,3% |
| | Outubro | 0,380 (0,219) | 87,6% |
| | Novembro | 0,280 (0,166) | 87,8% |
| Dia da semana | | | |
| | Segunda-feira | 0,189 (0,128) | 80,6% |
| | Terça-feira | 0,360 (0,211) | 86,5% |
| | Quarta-feira | 0,385 (0,195) | 88,1% |
| | Quinta-feira | 0,433 (0,220) | 88,6% |
| | Sexta-feira | 0,421 (0,213) | 87,6% |
| | Sábado | 0,399 (0,205) | 92,9% |
| | Domingo | 0,224 (0,155) | 88,9% |

A Tabela 7.6 apresenta a análise de disponibilidade para o tipo de instância g2.2xlarge. Como na Tabela 7.5, a menor disponibilidade também ocorre às segundas-feiras para este tipo de instância, o que pode indicar que os finais de semana têm um comportamento distinto em comparação aos dias da semana. Nestes 3 meses, a maior cotação média ocorreu às quintas-feiras, quando o preço estimado foi de 0,433 USD/hora (66,62% do preço *on demand*), com 88,6% de disponibilidade. Se, no entanto, o usuário decidir fazer um lance para a mesma instância aos domingos, o preço médio estimado será de 0,224

USD/hora (34,46% do preço *on demand*) com 88,9% de disponibilidade. Mesmo com a alta variação nos preços observada para esta instância, se um cliente de nuvem usasse o preço estimado como oferta em um domingo de setembro a novembro de 2016, ele teria pago muito menos do que o preço *on demand*, com alta disponibilidade.

Análise do Tipo de Instância m4.10xlarge

A Figura 7.9 mostra o preço *spot* real e estimado para setembro, outubro e novembro de 2016, considerando a instância m4.10xlarge. Comparado com os resultados descritos nas Figuras 7.7 e 7.8, pode-se notar que há menos picos, dando uma indicação de que os preços *spot* das instâncias m4.10xlarge estavam mais estáveis durante o período de tempo analisado.

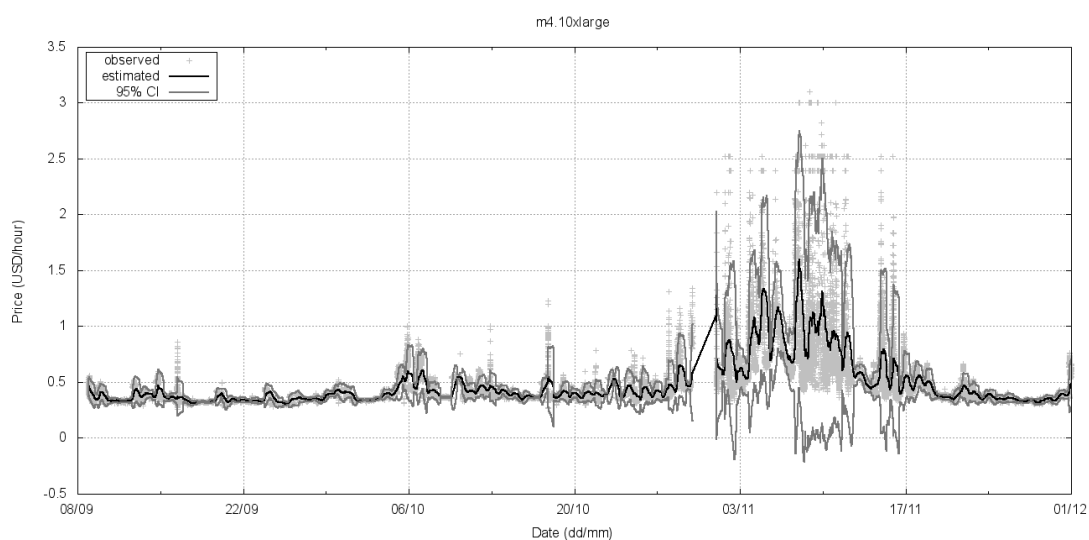


Figura 7.9: Preços observados e estimados para a instância m4.10xlarge durante os meses de setembro, outubro e novembro de 2016.

Tabela 7.7: Análise de custo e disponibilidade da instância m4.10xlarge.

| | | Estimativa de Preço Máximo em USD/hora Média (DP) | Disponibilidade |
|-----------------------------------|---------------|--|-----------------|
| Período completo (3 meses) | | 0,643 (0,439) | 90,7% |
| Mês | | | |
| | Setembro | 0,391 (0,106) | 92,1% |
| | Outubro | 0,511 (0,113) | 89,0% |
| | Novembro | 0,895 (0,590) | 91,2% |
| Dia da semana | | | |
| | Segunda-feira | 0,512 (0,353) | 85,0% |
| | Terça-feira | 0,690 (0,501) | 91,1% |
| | Quarta-feira | 0,673 (0,432) | 89,5% |
| | Quinta-feira | 0,718 (0,514) | 90,0% |
| | Sexta-feira | 0,602 (0,418) | 91,4% |
| | Sábado | 0,605 (0,367) | 95,4% |
| | Domingo | 0,609 (0,407) | 95,6% |

Isso também pode ser visto na Tabela 7.7, na qual a estratégia proposta conseguiu estimar preços de lance muito baixos, com alta disponibilidade. Por exemplo, o lance médio sugerido aos domingos (0,609 USD/hora) equivale a 25,44% do preço *on demand* (vide Tabela 7.4), com uma disponibilidade estimada em 95,6%. O mês de setembro apresentou os melhores resultados dentre os períodos mensais analisados, com uma taxa de disponibilidade média de 92,1% e um valor de lance estimado médio igual a 0,391 USD/hora.

Análise do Tipo de Instância r3.2xlarge

A Figura 7.10 mostra o preço *spot* real e estimado para o tipo de instância r3.2xlarge, de setembro a novembro de 2016. Como na análise de g2.2xlarge, há muitos picos na série de preços reais, mostrando uma grande variação de curto e longo prazo. A Tabela 7.8 mostra os resultados consolidados por mês e os dias da semana, mantendo a tendência de baixa disponibilidade para as segundas-feiras e taxas mais altas para os finais de semana.

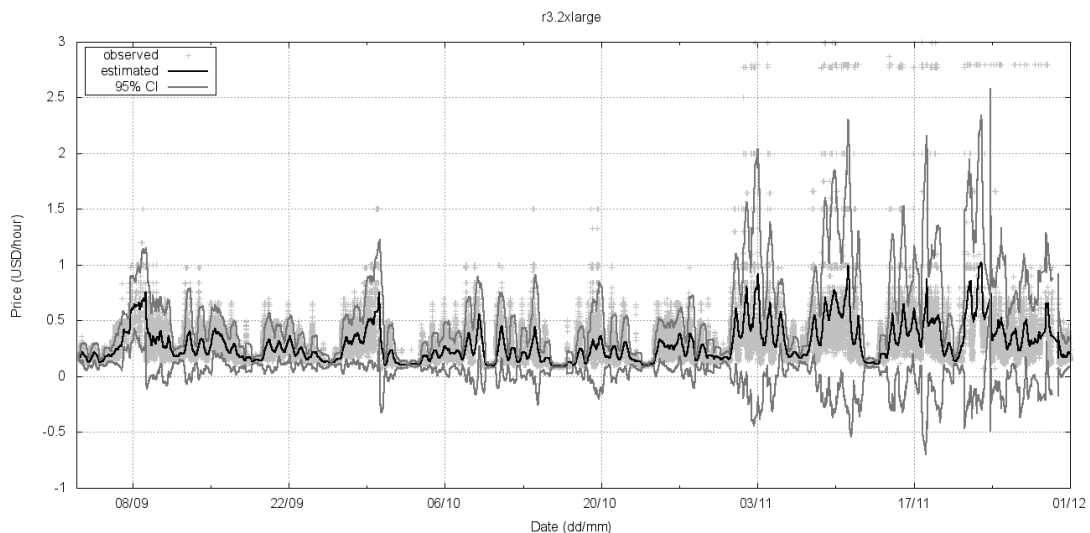


Figura 7.10: Preços observados e estimados para a instância r3.2xlarge durante os meses de setembro, outubro e novembro de 2016.

Conforme pode ser visto na Tabela 7.8, os valores de lance estimado para domingos (0,348 USD/hora) e segundas-feiras (0,226 USD/hora) apresentam os valores mais baixos, cerca de 52,33% e 33,98% do preço *on demand*, respectivamente. Entretanto, as maiores taxas de disponibilidade são registradas para quartas-feiras e sábados, com valores iguais a 91,5% e 95,7%, respectivamente. Há uma intensa variação de preços no mês de novembro, sendo que em alguns momentos o preço *spot* ultrapassa o preço *on demand*. Tais variações impactaram nas sugestões de lance, fazendo com que atingissem o valor médio de 0,790 USD/hora, que é mais elevado em comparação com os meses de setembro e outubro,

Tabela 7.8: Análise de custo e disponibilidade da instância r3.2xlarge.

| | | Estimativa de Preço Máximo em USD/hora Média (DP) | Disponibilidade |
|-----------------------------------|---------------|--|-----------------|
| Período completo (3 meses) | | 0,527 (0,364) | 89,4% |
| Mês | | | |
| | Setembro | 0,414 (0,219) | 89,1% |
| | Outubro | 0,368 (0,189) | 87,3% |
| | Novembro | 0,790 (0,458) | 92,2% |
| Dia da semana | | | |
| | Segunda-feira | 0,226 (0,125) | 77,2% |
| | Terça-feira | 0,484 (0,275) | 90,7% |
| | Quarta-feira | 0,611 (0,366) | 91,5% |
| | Quinta-feira | 0,725 (0,446) | 91,2% |
| | Sexta-feira | 0,717 (0,410) | 89,7% |
| | Sábado | 0,572 (0,321) | 95,7% |
| | Domingo | 0,348 (0,222) | 89,6% |

cujos valores foram de 0,414 USD/hora e 0,368 USD/hora. Apesar disso, a estimativa de disponibilidade média se manteve alta para o mês de novembro, com taxa igual a 92,2%.

Análise do Tipo de Instância t1.micro

Por fim, a Figura 7.11 mostra o preço *spot* atual e estimado para setembro, outubro e novembro de 2016, considerando o tipo de instância t1.micro. Nela é possível observar que os preços estavam relativamente estáveis até a última semana de outubro de 2016, quando ocorreu uma alta variação. Isso mostra que, mesmo para tipos de instâncias antigas e bem estabelecidas, como t1.micro, grandes variações podem ocorrer de forma inesperada. Apesar desse fenômeno, o modelo se adaptou em razão da janela flutuante de 12h ter conseguido captar a alta variação de preços.

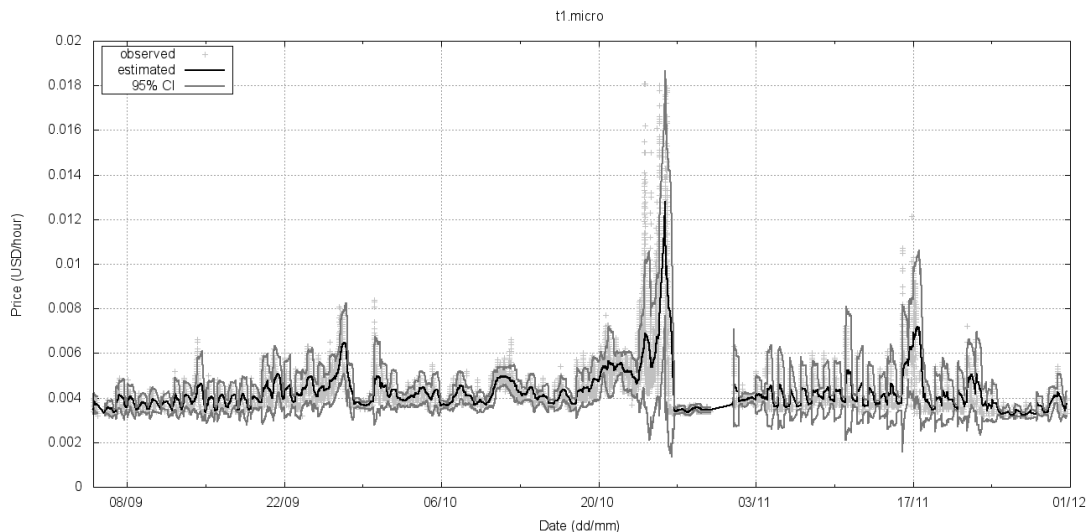


Figura 7.11: Preços observados e estimados para a instância t1.micro durante os meses de setembro, outubro e novembro de 2016.

Tabela 7.9: Análise de custo e disponibilidade da instância t1.micro.

| Estimativa de Preço Máximo em USD/hora | | Disponibilidade |
|--|---------------|-----------------|
| Média (DP) | | |
| Período completo (3 meses) | | 91,4% |
| Mês | | |
| | Setembro | 89,3% |
| | Outubro | 92,4% |
| | Novembro | 90,9% |
| Dia da semana | | |
| | Segunda-feira | 89,5% |
| | Terça-feira | 91,0% |
| | Quarta-feira | 91,6% |
| | Quinta-feira | 89,3% |
| | Sexta-feira | 93,6% |
| | Sábado | 93,7% |
| | Domingo | 91,5% |

A tabela 7.9 mostra os resultados consolidados para cada mês e dia da semana. As taxas de disponibilidade média variam de 89,5% a 93,7% para os dias da semana, com destaque para sextas-feiras, sábados e domingos, quando as taxas ficaram mais altas, com valores iguais a 93,6%, 93,7% e 91,5%, respectivamente, sendo que a sugestão de lance médio se manteve em apenas 0,005 USD/hora, valor equivalente a 25% do preço *on demand*.

7.3 Considerações sobre as Modelagens Estatísticas

A partir da modelagem estatística da precificação *on demand* foi possível mostrar que existe uma influência dos recursos de infraestrutura computacional, como processamento e memória, no custo das instâncias nos provedores Amazon EC2 e Google GCP. Assim, acredita-se que as equações 7.2 (MLR-ECU-Restricted) e 7.4 (MLR-Restricted-no-core) podem ser utilizadas com sucesso em estratégias de seleção de instâncias *on demand* no provedor Amazon EC2, cujo propósito é o uso intensivo de processador ou memória. Em ambas equações, notou-se que a quantidade de memória contribui menos para o preço do que o poder de computação. Entretanto, é importante mencionar que a generalização dessa abordagem requer análises para tipos de instâncias especializadas em outros propósitos, como por exemplo uso intensivo de *GPU* ou *storage*. Nestes casos, outras variáveis devem ser consideradas na composição dos custos dos tipos de instâncias.

A modelagem estatística da precificação *spot* proporcionou as estimativa de custos reduzidos e com boa disponibilidade das instâncias, em 2016, com o modelo de mercado real. Considerando o limite superior do intervalo de confiança de 95% da média móvel como estratégia para sugestão de lances, a disponibilidade pôde alcançar mais de 87% para o tipo de instância g2.2xlarge (pior caso), e cerca de 91% para t1.micro (melhor

caso). O valor do lance médio para este último tipo de instância foi definido em torno de 30% do preço da demanda, revelando a eficácia da estratégia.

Como pode ser visto na Figura 7.7, o limite superior dos valores de intervalo de confiança 95% se adaptam para períodos de alta variação de preços para o tipo de instância *c3.8xlarge*, como ocorrido da primeira metade de novembro de 2016. Como as instâncias *spot* seguem um modelo baseado no mercado, existem variáveis externas não mapeadas que causam variação de preço, mas mesmo nesses períodos específicos o limite superior rapidamente se adapta aos picos de cobertura dos preços observados nas últimas 12 horas. Um fenômeno semelhante pode ser observado para os outros tipos de instância analisados (Figuras 7.7, 7.9, 7.10 e 7.11). Portanto, conclui-se que a estratégia estatística baseada em médias móveis suavizadas foi bem-sucedida na modelagem do comportamento de instâncias *spot*, bem como na previsão de preços com manutenção da disponibilidade da instância.

Ao término da modelagem e da realização das análises estatísticas das instâncias *spot*, foi possível estabelecer uma forma de reduzir o custo financeiro de uma instância *spot*, mantendo estatisticamente a disponibilidade dentro de um intervalo de confiança. Priorizou-se assim a minimização do custo. Em seguida, foi levantada a hipótese de formulação de um mecanismo voltado ao balanceamento de custo e disponibilidade, o que deu origem à abordagem detalhada no Capítulo 8.

Capítulo 8

Mecanismo de Utilidade para Instâncias *Spot* da Amazon EC2

A primeira contribuição primária desta Tese consiste na proposta e avaliação de um mecanismo baseado em utilidade do ponto de vista do usuário de computação em nuvem. Tal mecanismo busca equilibrar a diminuição do custo da instância *spot* e o aumento da sua disponibilidade, proporcionando um balanceamento desses dois objetivos conflitantes. A proposta do mecanismo, assim como as análises realizadas e os resultados obtidos, foram publicados em [49], usando dados de 2016. O mecanismo foi também aplicado ao modelo de mercado suavizado, com dados de 2020 e 2021, conforme descrito no Capítulo 9.

A estrutura deste capítulo se inicia pela Seção 8.1, que detalha o mecanismo baseado em utilidade. Nesta seção são apresentados também a função e o algoritmo de utilidade utilizados no mecanismo. Na Seção 8.2, os dados de histórico de precificação utilizados e os resultados da aplicação do mecanismo de utilidade no mercado *spot* são discutidos. Ao final deste capítulo, as considerações finais sobre o mecanismo são apresentadas na Seção 8.3.

8.1 Projeto do Mecanismo

Do ponto de vista do usuário de computação em nuvem, é importante equilibrar a disponibilidade e o custo da instância *spot*. Para projetar o mecanismo baseado em utilidade e considerando os conceitos da Seção 4.1, primeiramente foi definida uma função de utilidade para equilibrar a disponibilidade e os fatores de custo. Em seguida, utilizando uma modelagem de série temporal, conforme abordado na Seção 4.2, definiu-se um algoritmo que usa uma janela deslizante no tempo com base na função de utilidade, a fim

de determinar o valor ideal de disponibilidade e custo da instância que maximize essa utilidade.

8.1.1 Função de Utilidade

O usuário que contrata instâncias *spot* precisa lidar com dois objetivos conflitantes, que são o custo e a disponibilidade das instâncias *spot*. O primeiro objetivo refere-se a minimizar o custo, enquanto o segundo objetivo visa maximizar a disponibilidade. Supondo que os dados do histórico estejam disponíveis por um período fixo de tempo antes do uso da instância, foi proposto uma estratégia que calcula (a) a utilidade esperada u com base em um lance estimado b , até θ vezes o valor do preço *on demand* p_{od} ; e (b) um valor estimado da disponibilidade a de cada lance. Assim, a função de utilidade é apresentada na Equação 8.1.

$$u = a\sqrt{(\theta p_{od} - b)} \quad (8.1)$$

Nesta equação, a disponibilidade a é calculada como o número de variações de preço *spot* abaixo do lance estimado b , definido como n_{below} , sobre o número total de variações de preços, definido como n_{all} , considerando uma janela de tempo predefinida antes do momento da avaliação do lance/custo (Equação 8.2).

$$a = \frac{n_{below}}{n_{all}} \quad (8.2)$$

A função de utilidade (Equação 8.1) privilegia a disponibilidade sobre o custo, por isso a raiz quadrada no último termo. O objetivo é estabelecer os valores de lance e disponibilidade que maximizem a utilidade esperada, conforme mostrado na Fórmula 8.3. Tais valores balanceiam o custo e a disponibilidade da instância, do ponto de vista do usuário.

$$a, b \rightarrow \max u \quad (8.3)$$

Neste cenário, considerando as Equações 8.1 e 8.2, assim como a Fórmula 8.3, a utilidade u começa com um valor baixo, que tende a aumentar à medida que o lance estimado aumenta, até atingir um valor de u ideal, em que a disponibilidade a e o lance b são os valores equilibrados.

8.1.2 Algoritmo de Utilidade

Para calcular a função de utilidade apresentada na Equação 8.1, proposta na Seção 8.1.1, foi definido o Algoritmo 5. Os parâmetros esperados são o preço *on demand* referente

ao tipo de instância (p_{od}), o fator de multiplicação do preço *on demand* (θ), a precisão de estimativa dos lances (σ) e a janela de tempo da série temporal em que ocorreram as variações de preços *spot* (P_w).

Algorithm 5 *UtilityFunction*($p_{od}, \theta, \sigma, P_w$)

```

1:  $B_e \leftarrow [0, \theta p_{od}]$ , de maneira que  $b_{i+1} - b_i = \sigma$ 
2: for each  $b_i \in B_e$  do
3:    $n_{below} \leftarrow 0$ 
4:   for  $j \leftarrow (t - w)$  to  $t$ ,  $t \in P_w$  do
5:     if  $p_j \leq b_i$  then
6:        $n_{below} \leftarrow n_{below} + 1$ 
7:     end if
8:   end for
9:    $a_i \leftarrow n_{below} / \text{sizeof}(P_w)$ 
10:   $u_i \leftarrow a_i \sqrt{(\theta p_{od} - b_i)}$ 
11: end for
12: return  $u, a, b \rightarrow \max u$ 

```

Na linha 1 do Algoritmo 5, os valores dos lances estimados variam de forma crescente, de 0 até θ vezes o preço *on demand* p_{od} , de forma incremental e de acordo com o valor de precisão definido por σ . Estes valores são armazenados no conjunto B_e . Por exemplo, se σ é igual a 10^{-2} , os valores dos lances estimados são 0,01, 0,02, 0,03 e assim por diante, até o limite θp_{od} . Assim, o conjunto B_e pode conter mais ou menos elementos de acordo com o valor de precisão definido por σ . Os valores desse conjunto são avaliados no laço de iteração que compreende as linhas 2 e 11. Na linha 3, a variável n_{below} é inicializada com valor 0 para ser utilizada no laço que compreende as linhas 4 a 8, para contar a quantidade de vezes em que o valor do preço *spot* p_j é menor do que o valor do lance estimado $b_i \in B_e$ no decorrer da janela de tempo, ou seja, entre $(t - w)$ e t . Na linha 9, o valor da disponibilidade a_i referente ao lance estimado b_i é calculado utilizando a Equação 8.2. Em seguida, a utilidade u_i é calculada como uma função do lance b_i e da disponibilidade a_i , conforme mostrado na linha 10. Ao final, na linha 12, o algoritmo retorna o valor da utilidade, assim como os valores do lance e da disponibilidade que maximizam a utilidade calculada, conforme estabelecido na Fórmula 8.3.

Em seguida, foi projetado o algoritmo *CalculateUtilitySuggestion* (Algoritmo 6), que define os parâmetros necessários para a execução do Algoritmo 5, de forma a obter a sugestão balanceada de lance e disponibilidade da instância. Inicialmente, o valor do preço *on demand* p_{od} referente à instância é recuperado na linha 1. Nas linhas 2, 3 e 4 são definidos os valores de θ , σ e do tamanho da janela de tempo w . Assumindo t como o momento corrente da análise, o histórico de variação de preços *spot* é delimitado no intervalo $[p_{t-w}; p_t]$ e armazenado na variável P_w (linha 5). Na linha 6, o algoritmo *UtilityFunction* (Algoritmo 5) é chamado com os parâmetros previamente definidos p_{od} ,

P_w , θ e σ . Os valores da utilidade u , da disponibilidade a e do lance b são retornados na linha 7.

Algorithm 6 *CalculateUtilitySuggestion(instance)*

```

1:  $p_{od} \leftarrow$  on-demand price for instance
2:  $\theta \leftarrow 1$ 
3:  $\sigma \leftarrow 10^{-2}$ 
4:  $w \leftarrow 12h$ 
5:  $P_w \leftarrow [p_{t-w}; p_t]$ , such that  $p_i \in P$ 
6:  $u, a, b \leftarrow UtilityFunction(p_{od}, \theta, \sigma, P_w)$ 
7: return  $u, a, b$ 

```

O algoritmo 5 possui uma complexidade de tempo quadrática ($O(B_e P_w)$), onde B_e refere-se ao número de lances a serem avaliados, que depende da precisão σ e P_w está relacionado ao número de variações de preços na janela de tempo, neste caso, definido como 12 horas antes do momento atual. Na prática, o cálculo da utilidade, da disponibilidade e do lance em um determinado momento t , considerando uma janela de tempo de 12 horas antes de t , é da ordem de poucos milissegundos.

8.1.3 Extensão do Algoritmo de Utilidade

Para auxiliar o usuário a decidir o melhor dia para contratar uma instância *spot*, considerando o momento atual e as variações de preços nos dias da semana anterior, propomos uma extensão do Algoritmo 6. O objetivo principal é dar uma sugestão ao usuário sobre o momento do lance, ou seja, se deve contratar a instância imediatamente ou aguardar um determinado dia da semana. O benefício se traduz em uma instância *spot* com custo menor e com maior disponibilidade esperada.

Assim, para fornecer a sugestão do momento ideal, foi projetado o Algoritmo 7, que usa o cálculo baseado em utilidade do Algoritmo 5. A ideia principal consiste em utilizar uma janela composta, formada pelas variações de preço das últimas 12 horas, como ocorre no Algoritmo 6, e pelas variações ocorridas 24 horas de cada dia da semana anterior ao momento no qual a análise está sendo realizada. Portanto, o intervalo de tempo $[p_{wd_{t_0}}, p_{wd_{t_n}}]$ refere-se às variações de preço de um dia inteiro da semana anterior. Por exemplo, se a análise for realizada na segunda-feira às 14h, serão consideradas a janela de 12 horas de segunda-feira (2h às 13h59) e as janelas dos dias da semana anterior, referentes a terça-feira (0h às 23h59), quarta-feira (0h às 23h59) e assim por diante, até domingo (um dia antes da análise).

No algoritmo *CalculateUtilityDayWeekSuggestion* (Algoritmo 7), os valores de utilidade, disponibilidade e lance são calculados primeiramente considerando apenas a janela das últimas 12 horas (linha 5). Os valores são adicionados ao conjunto U (linha 6). A

Algorithm 7 *CalculateUtilityDayWeekSuggestion(instance)*

```
1:  $p_{od} \leftarrow$  on-demand price for instance
2:  $\theta \leftarrow 1$ 
3:  $\sigma \leftarrow 10^{-2}$ 
4:  $w \leftarrow 12h$ 
5:  $u_0, a_0, b_0 \leftarrow UtilityFunction(p_{od}, \theta, \sigma, P_w)$ 
6:  $U[0] \leftarrow u_0, a_0, b_0$ 
7:  $D \leftarrow \{Mon, Tue, Wed, Thu, Fri, Sat, Sun\}$ 
8: for  $wd$  in  $D - today$  do
9:    $P_w \leftarrow [p_{t-w}, p_t] + [p_{wd_{t_0}}, p_{wd_{t_n}}]$ , such that  $p_i \in P$ 
10:   $u_i, a_i, b_i \leftarrow UtilityFunction(p_{od}, \theta, \sigma, P_w)$ 
11:   $U[i] \leftarrow u_i, a_i, b_i$ , such that  $1 \geq i \leq 6$ 
12: end for
13: return (today or  $wd_k \in D$ ),  $(a_k, b_k \rightarrow \max u_k$ , such that  $u_k, a_k, b_k \in U$ )
```

variável wd assume os valores de segunda a domingo, considerando cada dia da semana, exceto o dia atual. A utilidade é então calculada para cada dia da semana (linhas 8 a 12) e os valores de utilidade, disponibilidade e lance também são adicionados ao conjunto U (linha 11). Na linha 13, são recuperados os valores de disponibilidade (a_k) e lance (b_k) que maximizam a utilidade (u_k), assim como a indicação de hoje ou do dia da semana em que isso acontece (*today* ou $wd_k \in D$).

8.2 Resultados dos Experimentos

8.2.1 Dados Utilizados

Os experimentos consideraram um conjunto de dados de três meses de históricos de preços de instância *spot* fornecidos pela Amazon [75]. Os dados referem-se a instâncias virtuais localizadas na região *US-East* com o sistema operacional Linux. Entre os tipos de instância analisados, há dois tipos de instâncias de propósito geral (t1.micro e m4.10xlarge) e outros três tipos com propósitos específicos, como uso otimizado de CPU (c3.8xlarge), memória (r3.2xlarge) e GPU (g2.2xlarge). A Tabela 8.1 apresenta os tipos de instâncias com seus respectivos propósitos e preços *on demand*. A tabela também mostra o número de variações de preço do histórico para cada tipo de instância, durante o período de setembro a novembro de 2016 considerado em nossa análise.

8.2.2 Implementação e Disponibilidade de Código

O algoritmo 5 (Seção 8.1.2) foi avaliado preliminarmente usando a linguagem de programação R para computação estatística e, em seguida, implementado em linguagem Java,

Tabela 8.1: Tipos de instância, preços *on demand* e número de variações de preço *spot* de setembro a novembro de 2016.

| # | Tipo de Instância | Propósito de Utilização | <i>On demand</i> (USD/hora) | # de Variações de Preço <i>spot</i> |
|---|-------------------|-------------------------|-----------------------------|-------------------------------------|
| 1 | t1.micro | General Purpose | 0,020 | 38.460 |
| 2 | c3.8xlarge | Compute Optimized | 1,680 | 212.840 |
| 3 | r3.2xlarge | Memory Optimized | 0,665 | 218.495 |
| 4 | g2.2xlarge | GPU Instance | 0,650 | 187.049 |
| 5 | m4.10xlarge | General Purpose | 2,394 | 66.427 |

na versão 8. Todas as implementações estão disponíveis no repositório *Accurate Spot Analysis and Prediction (ASAP) Framework*¹ do GitHub.

8.2.3 Experimentos com Mecanismo de Utilidade

Os resultados da análise baseada em utilidade para o tipo de instância *c3.8xlarge* são mostrados na Figura 8.1. O experimento usou um valor de θ igual a 2. A taxa média de disponibilidade por três meses foi de 95,89%, com um valor médio de lance de 0,8456 USD/hora, o que equivale a 50,33% do preço *on demand*. Esta figura mostra a variação de preços (cruzes em cinza), os valores de lance/custo (pontos azuis relacionados ao eixo Y esquerdo) e as taxas de disponibilidade (pontos vermelhos relacionados ao eixo Y direito). Como pode ser visto, há uma enorme variação de preço nas duas primeiras semanas de novembro, com muitas situações em que o preço *spot* supera o preço *on demand* (linha preta horizontal), o que afeta os valores estimados de lance e disponibilidade, o que explica o θ escolhido. Nos últimos dez dias de novembro, as taxas de disponibilidade atingiram 99,99%, com estimativas de lances de 0,5 USD/hora (29,76% do preço *on demand*), devido às menores variações de preço durante esse período.

Para o tipo de instância *t1.micro* (Figura 8.2), os resultados mostram uma taxa de disponibilidade média de 97,8%, com um valor de lance médio de 0,0064 USD/hora, o que equivale a 32,71% do preço *on demand*. Este foi o único experimento que usou um valor σ igual a 10^{-3} , já que o preço *on demand* (0,002 USD/hora) é o menor em comparação com as demais instâncias. O valor de θ foi definido como 1, uma vez que não foram observadas ocorrências de variações dos preços *spot* acima do preço *on demand*. Porém, um cenário de grande variação de preços ocorreu no final de outubro, afetando negativamente os valores estimados. No entanto, a abordagem foi capaz de reduzir o impacto nas estimativas, sugerindo lances de cerca de 0,0101 USD/hora, o que equivale a 50,5% do preço *on demand*.

¹Disponível em <http://github.com/gjportella/asap>

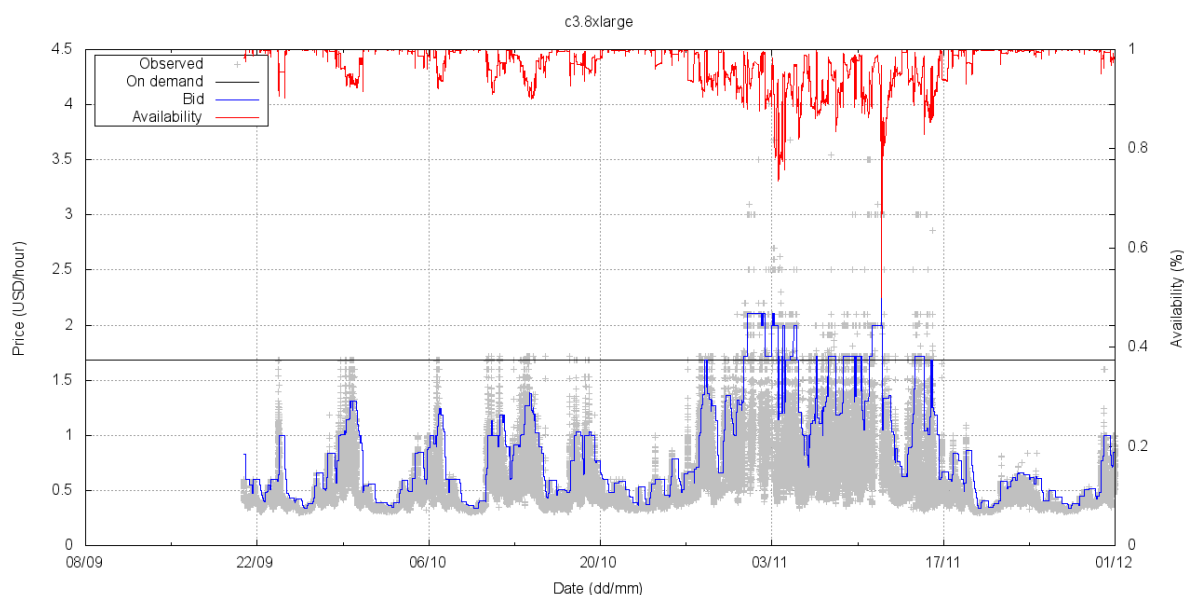


Figura 8.1: Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância *c3.8xlarge* ($\theta = 2$).

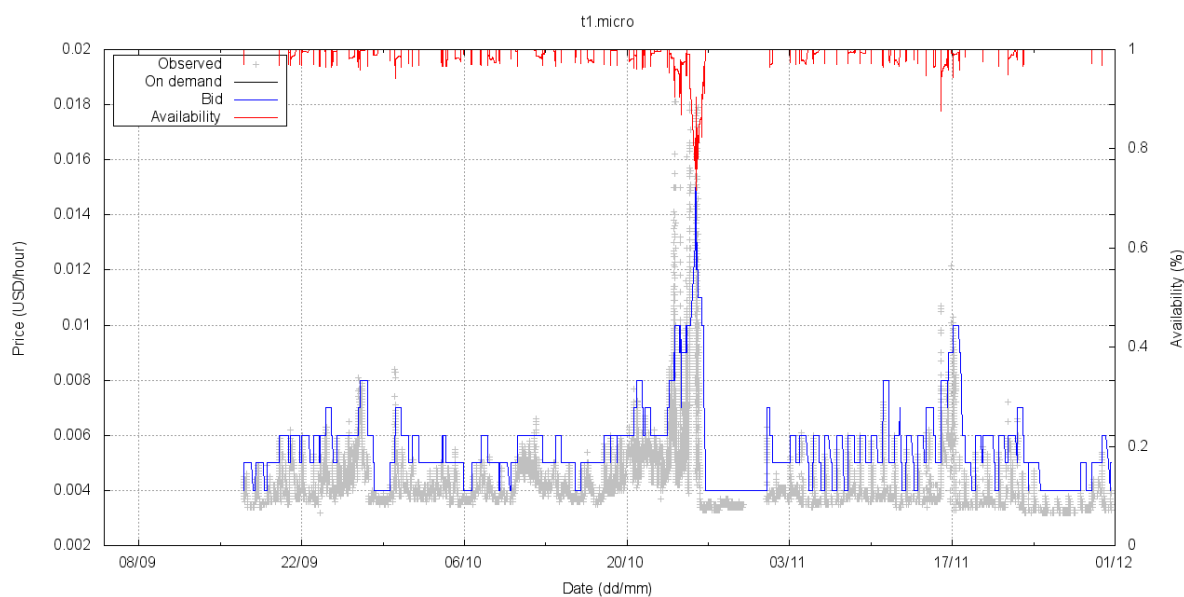


Figura 8.2: Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância *t1.micro* ($\theta = 1$).

Os cenários mais complexos foram observados para os tipos de instância *r3.2xlarge* (Figura 8.3) e *g2.2xlarge* (Figura 8.4), em que grandes variações de preços ocorreram em vários períodos de tempo. Para a instância *r3.2xlarge*, a taxa média de disponibilidade foi de 94,99%, com um valor médio de lance igual a 0,4790 USD/hora (72,03% do preço *on demand*). O θ usado para este experimento foi igual a 3, pois as variações de preço *spot* atingem cerca de três vezes o preço *on demand* em alguns casos. Para o tipo de

instância *g2.2xlarge*, definimos θ igual a 2, uma vez que foram observadas variações de preços *spot* até duas vezes o preço *on demand*. A taxa média de disponibilidade foi de 94,14%, com um valor médio de lance de 0,3435 USD/hora, o que equivale a 52,85% do preço *on demand*.

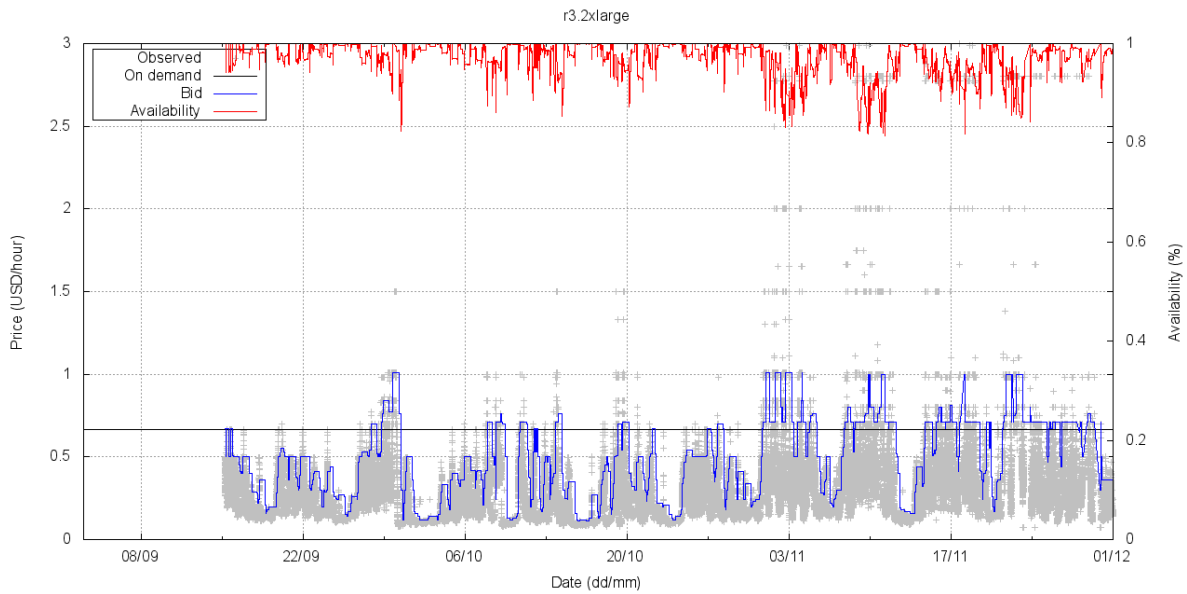


Figura 8.3: Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância *r3.2xlarge* ($\theta = 3$).

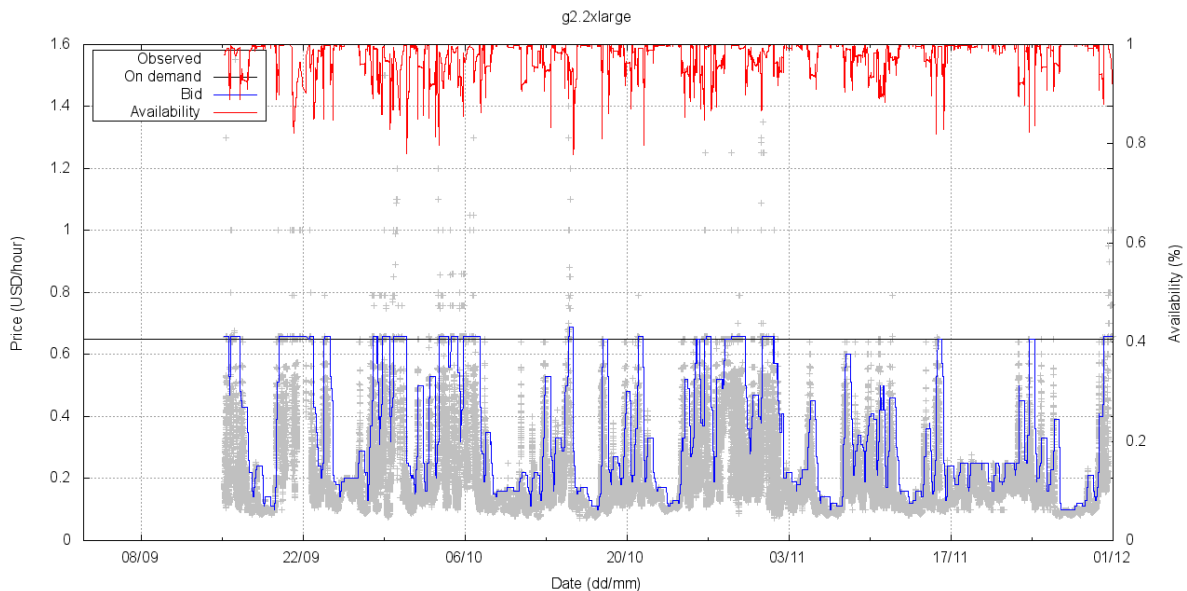


Figura 8.4: Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância *g2.2xlarge* ($\theta = 2$).

Por fim, a análise de lances e disponibilidade da instância *m4.10xlarge* é apresentada na Figura 8.5. A taxa média de disponibilidade foi de 98,19%, com um valor médio de lance igual a 0,689 USD/hora, o que equivale a 28,82% do preço *on demand*. O experimento usou um valor de θ igual a 2, pois neste caso também ocorreram variações de preço *spot* que excederam o preço *on demand*.

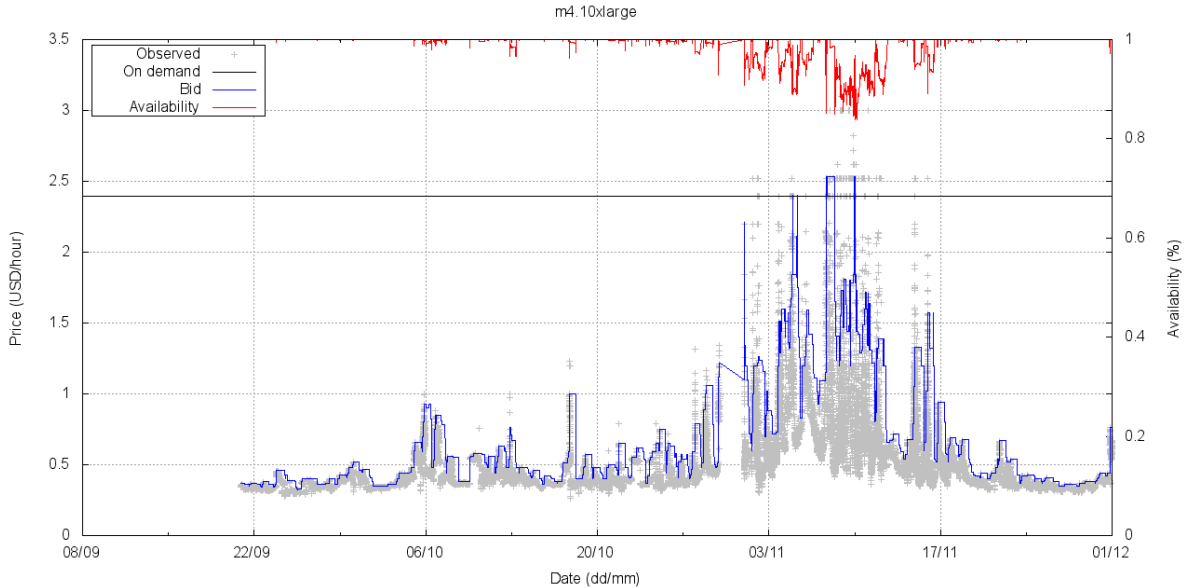


Figura 8.5: Análise de utilidade para equilíbrio de custo e disponibilidade para o tipo de instância *m4.10xlarge* ($\theta = 2$).

8.2.4 Experimentos com Mecanismo Estendido para Dia da Semana

A Tabela 8.2 mostra a análise de sugestões para o tipo de instância *t1.micro*, de setembro a novembro de 2016. A primeira coluna da tabela especifica em qual dia da semana as variações de preço foram analisadas. As oito colunas subsequentes mostram as porcentagens de sugestões para contratar a instância imediatamente (agora) ou aguardar o dia da semana especificado (de segunda a domingo).

Por exemplo, a segunda linha da Tabela 8.2 mostra as análises realizadas às segundas-feiras. Nesse dia da semana, em 41,8% dos momentos em que houve variação de preço *spot*, o Algoritmo 7 sugeriu contratar a instância imediatamente. Em 23,7% dos momentos em que houve variação de preços, nesse dia da semana, o mesmo algoritmo sugere esperar até domingo. Na terceira linha da Tabela 8.2, é feita a análise para as variações de preços ocorridas às terças-feiras no período de três meses. Para esse dia da semana, em 47,9% dos casos o algoritmo sugeriu a contratação da instância imediatamente. Em 24,5% dos casos, a sugestão do algoritmo foi por aguardar até domingo.

Para o tipo de instância *t1.micro*, contratar imediatamente é a melhor opção, embora haja percentuais relevantes para outros dias da semana, por exemplo, às terças-feiras e aos domingos. Como pode ser observado, às quintas-feiras (quinta linha da Tabela 8.2) as sugestões para contratar a instância imediatamente chegam a 46,2%. Já em 45,5% dos casos, o algoritmo sugere esperar até a próxima terça-feira.

Tabela 8.2: Percentuais de sugestões de dias da semana para *t1.micro*.

| Dia | Agora | Segunda | Terça | Quarta | Quinta | Sexta | Sábado | Domingo |
|---------|--------------|---------|-------|--------|--------|-------|--------|---------|
| Segunda | 41,8% | - | 0,0% | 0,1% | 16,8% | 14,4% | 3,3% | 23,7% |
| Terça | 47,9% | 5,1% | - | 5,2% | 5,6% | 3,8% | 7,8% | 24,5% |
| Quarta | 49,1% | 1,6% | 28,1% | - | 2,5% | 0,3% | 5,3% | 13,0% |
| Quinta | 46,2% | 1,2% | 45,5% | 0,1% | - | 4,6% | 0,0% | 2,5% |
| Sexta | 42,1% | 8,8% | 27,6% | 5,3% | 4,0% | - | 0,2% | 12,0% |
| Sábado | 49,4% | 3,6% | 36,6% | 0,0% | 4,6% | 0,0% | - | 5,7% |
| Domingo | 65,9% | 2,2% | 7,8% | 0,0% | 10,2% | 8,5% | 5,4% | - |

Os percentuais de sugestões de dias da semana para o tipo de instância *c3.8xlarge* são apresentados na Tabela 8.3. Nesse caso, os maiores valores percentuais se concentram nas sugestões de contratação imediata aos finais de semana (sábado e domingo), quando ocorrem menos variações de preços. Apenas nas segundas-feiras (segunda linha da tabela), a contratação imediata possui percentual de sugestão maior (36,4%) do que em relação aos finais de semana.

Tabela 8.3: Percentuais de sugestões de dias da semana para *c3.8xlarge*.

| Dia | Agora | Segunda | Terça | Quarta | Quinta | Sexta | Sábado | Domingo |
|---------|--------------|---------|-------|--------|--------|-------|--------------|--------------|
| Segunda | 36,4% | - | 0,0% | 0,8% | 8,2% | 0,0% | 19,8% | 34,8% |
| Terça | 13,6% | 0,9% | - | 0,1% | 3,5% | 3,9% | 43,7% | 34,2% |
| Quarta | 17,0% | 11,8% | 0,0% | - | 0,0% | 2,3% | 39,6% | 29,3% |
| Quinta | 28,3% | 7,2% | 12,0% | 3,1% | - | 0,0% | 18,4% | 31,0% |
| Sexta | 16,2% | 1,3% | 7,5% | 16,7% | 11,7% | - | 14,2% | 32,4% |
| Sábado | 61,5% | 0,0% | 0,0% | 0,2% | 12,4% | 0,0% | - | 25,9% |
| Domingo | 76,7% | 0,0% | 0,0% | 0,0% | 12,7% | 0,0% | 10,6% | - |

A Tabela 8.4 apresenta a análise de dia da semana para o tipo de instância *r3.2xlarge*. Observe que existem grandes variações de preços (Figura 8.3) ao longo do período de três meses. Nesse caso, o mecanismo sugere a contratação de instâncias *spot* aos domingos para a maioria dos casos. De acordo com a análise de recomendação, se um usuário contratasse uma instância em 3 de novembro (quinta-feira) às 18h20, ele pagaria um valor de lance próximo a 0,71 USD/hora, com uma taxa de disponibilidade estimada em 98,37%. No entanto, a recomendação para o usuário seria aguardar até o próximo domingo, 6 de

novembro, quando às 18h20 ele poderia dar um lance de 0,21 USD/hora (cerca de 29% do lance de quinta-feira), com uma taxa de disponibilidade de aproximadamente 99,99%.

Tabela 8.4: Percentuais de sugestões de dias da semana para *r3.2xlarge*.

| Dia | Agora | Segunda | Terça | Quarta | Quinta | Sexta | Sábado | Domingo |
|---------|--------------|---------|-------|--------|--------|-------|--------|--------------|
| Segunda | 67,9% | - | 0,0% | 0,2% | 0,0% | 0,2% | 0,0% | 31,7% |
| Terça | 15,7% | 2,4% | - | 6,8% | 0,0% | 5,3% | 0,0% | 69,7% |
| Quarta | 29,5% | 1,1% | 6,6% | - | 2,3% | 0,5% | 0,0% | 59,9% |
| Quinta | 20,8% | 5,2% | 1,5% | 3,1% | - | 3,0% | 2,2% | 64,3% |
| Sexta | 19,7% | 2,6% | 0,0% | 3,6% | 1,2% | - | 7,5% | 65,3% |
| Sábado | 39,0% | 1,9% | 0,7% | 0,0% | 0,0% | 7,0% | - | 51,4% |
| Domingo | 97,4% | 1,7% | 1,0% | 0,0% | 0,0% | 0,0% | 0,0% | - |

8.3 Considerações sobre o Mecanismo de Utilidade

Neste capítulo, foi apresentado um modelo baseado em utilidade que equilibra o custo e a disponibilidade da instância *spot* para os usuários do Amazon EC2. Os resultados experimentais mostraram altas taxas de disponibilidade com baixos custos quando comparados aos preços *on demand* para diferentes tipos de instâncias.

Para o tipo de instância de propósito geral *m4.10xlarge*, as recomendações baseadas no nosso mecanismo atingiram uma taxa média de disponibilidade de 98,19% para um lance médio igual a 28,82% do preço *on demand*. O mecanismo de recomendação também apoia a decisão do usuário de contratar a instância imediatamente ou aguardar o melhor dia da semana para fazê-lo. Para o tipo de instância *t1.micro*, a sugestão de contratação imediata prevalece. Para as instâncias *c3.8xlarge* e *r3.2xlarge*, as sugestões de contratação direcionam para os finais de semana, quando ocorrem menos variações de preços.

Em razão das análises do mecanismo de utilidade utilizarem dados de histórico de precificação *spot* referentes ao período de 2016, ou seja, antes da mudança para o mercado suavizado, conforme detalhado na Seção 6.1.19, as análises de utilidade são rediscutidas no Capítulo 9.

Capítulo 9

Mecanismo Combinado para Análise de Tendência e Previsão de Preços *Spot* da Amazon EC2

A segunda contribuição primária desta Tese compreende um mecanismo que combina a análise de utilidade apresentada no Capítulo 8 com uma análise de tendências de preços de instâncias *spot* da Amazon EC2. De forma mais específica, o mecanismo combinado compreende:

- A análise de utilidade, que equilibra custo e disponibilidade de instância *spot* a curto prazo, de forma a oferecer sugestão de lances para o usuário de computação em nuvem;
- A utilização de aprendizado de máquina (ou *machine learning*) para previsão de preço *spot* a longo prazo utilizando *Long Short Term Memory* (LSTM), que é uma técnica de rede neural recorrente adaptada para previsão em séries temporais, conforme explicado na Seção 5.3.2.

O mecanismo combinado aplica a técnica de LSTM para analisar tendências de preços *spot* a longo prazo, de forma a ajudar o usuário a selecionar uma instância apropriada. Em seguida, o mecanismo baseado em utilidade é aplicado para definir estimativas do lance e disponibilidade esperada da instância escolhida. Também são apresentadas a arquitetura e as configurações mais eficazes das redes LSTM para esta finalidade.

Diversos experimentos são realizados com o mecanismo combinado, nos quais são analisados diferentes tipos de instâncias e períodos de histórico de preços *spot* no decorrer do ano de 2020, ou seja, após a mudança do mercado *spot* para o modelo suavizado (Seção 6.1.19).

Finalmente, para avaliar o mecanismo combinado, foi elaborado e desenvolvido um estudo de caso com base em uma aplicação de Bioinformática, que compara milhares de sequências de DNA do SARS-CoV-2 (hospedeiro humano). O SARS-CoV-2 é o vírus que causa a doença covid-19, responsável pela pandemia que se alastrou pelo mundo em 2020. Assim sendo, foi mostrado que, com a nossa análise combinada, é possível selecionar instâncias apropriadas e definir lances ou preços máximos *spot* muito mais baixos do que o preço *on demand*, com disponibilidade próxima a proporcionada neste último modelo de precificação.

A proposta do mecanismo combinado, os resultados dos experimentos e o estudo de caso, assim como o conteúdo restante deste capítulo, foram submetido a um periódico prestigioso em Outubro de 2020. Uma versão revisada foi submetida em fevereiro de 2021 e, atualmente, está em processo de revisão final [50].

O restante do capítulo possui a seguinte estrutura. A Seção 9.1 detalha o mecanismo baseado em redes neurais LSTM para análise de tendências de preços. O conceito de disponibilidade real baseado na sugestão de lance fornecida pelo mecanismo de utilidade é definido na Seção 9.2. Os experimentos realizados com os mecanismos LSTM e de utilidade, com dados de histórico de precificação de instâncias *spot* de 2020, são detalhados na Seção 9.3. A Seção 9.4 apresenta a metodologia de uso do mecanismo combinado. Na Seção 9.5, é apresentado o estudo de caso que aplica o mecanismo combinado (redes LSTM e utilidade). Finalmente, na Seção 9.6, as considerações finais são apresentadas.

9.1 Projeto do Mecanismo LSTM

O mecanismo proposto no presente capítulo combina análise de tendências de preço *spot* a longo prazo com previsão e balanceamento de custo e disponibilidade a curto prazo. A análise de tendências de preços *spot* a longo prazo se baseia na utilização de redes LSTM conforme detalhado no Capítulo 5.

As redes LSTM têm sido empregadas em diversas aplicações e, recentemente, amplamente aplicadas à previsão de valores reais em séries temporais [127]. Conforme apresentado no Capítulo 6, existem vários trabalhos que utilizam redes LSTM para prever preços no modelo de precificação *spot* da Amazon EC2. Estes trabalhos, entretanto, definem uma arquitetura única de rede neural, com uma quantidade fixa de camadas e de nós dentro de cada camada, assim como valores fixos para hiperparâmetros de aprendizado da rede. Por exemplo, [36] e [25] propõem arquiteturas de rede e configurações de parâmetros distintos para tratar do problema de previsão de preço *spot*. Desse modo, não se pode concluir que existe uma arquitetura LSTM única ou configuração de hiperparâmetros definitiva que darão bons resultados para todas as instâncias e em diferentes períodos de tempo.

O mecanismo LSTM proposto se baseia em uma abordagem flexível, que permite ao usuário explorar várias configurações de rede e diferentes valores de hiperparâmetros. O objetivo da abordagem consiste em fornecer uma análise de preço *spot* de longo prazo, mostrando claramente as variações e as tendências de preço esperadas em períodos iguais ou superiores a uma semana. Essa análise visa ajudar o usuário na escolha de um tipo de instância apropriado. Isso permite que o mecanismo LSTM possa ser usado de forma complementar ao mecanismo de utilidade para análise e sugestão de custo e disponibilidade a curto prazo, proposto no Capítulo 8.

9.1.1 Arquiteturas da Rede LSTM

No mecanismo LSTM apresentado nesta Tese foram propostos três *layouts* de arquitetura de rede neural, dependendo do número de camadas e dos tipos de elementos presentes nas camadas ocultas, conforme conceitos abordados na Seção 5.3.1. O primeiro *layout* (tipo 1) define uma arquitetura básica de rede neural recorrente, composta por apenas uma camada oculta de unidades LSTM, além das camadas de entrada e de saída da rede. O segundo *layout* (tipo 2) se caracteriza por uma arquitetura de rede intermediária, composta por duas camadas ocultas LSTM totalmente conectadas. O terceiro *layout* (tipo 3) define uma arquitetura mais sofisticada em relação às anteriores, com duas camadas LSTM e uma camada densa adicional e totalmente conectada.

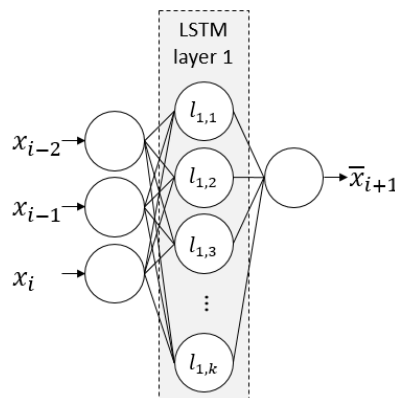


Figura 9.1: *Layout* arquitetural básico (tipo 1).

Os *layouts* básico e intermediário são mostrados nas Figuras 9.1 e 9.2, respectivamente. A Figura 9.3 mostra o terceiro *layout* de arquitetura, com duas camadas ocultas LSTM e uma camada oculta densa. Os valores históricos de preços compõem uma lista de *features*, que é submetida à rede através da camada de entrada. As camadas LSTM e densa têm o mesmo número de nós cada uma. A camada densa filtra e consolida o valor de previsão com a maior probabilidade, o que aumenta o tempo de treinamento.

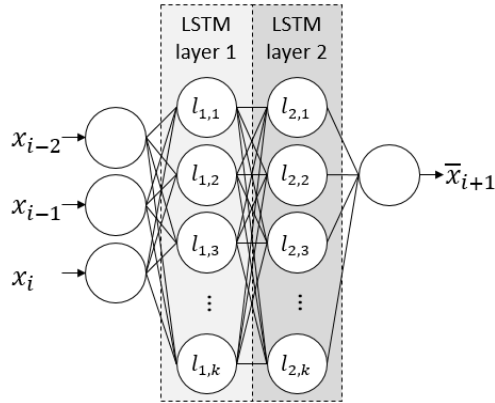


Figura 9.2: *Layout* arquitetural intermediário (tipo 2).

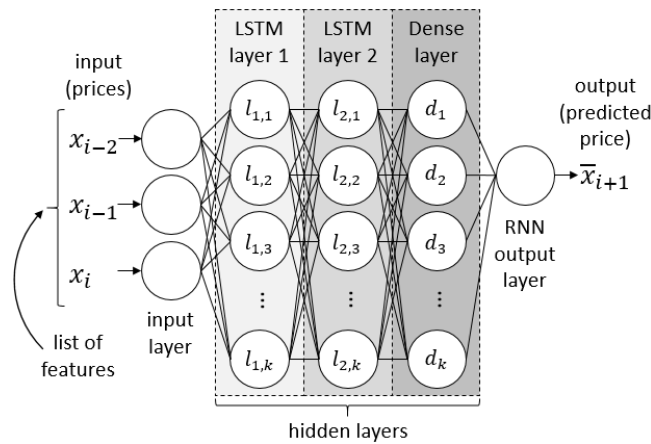


Figura 9.3: *Layout* arquitetural sofisticado (tipo 3).

9.1.2 Hiperparâmetros da Rede LSTM

Em relação à configuração dos parâmetros de treinamento da rede neural, conforme definições discutidas na Seção 5.1.7, foram analisadas variações discretas de quatro tipos de hiperparâmetros, que são (1) algoritmo de otimização, (2) quantidade de nós nas camadas ocultas, (3) quantidade de épocas de treinamento e (4) taxa de aprendizado, conforme detalhado na Tabela 9.1. Estes hiperparâmetros de (1) a (4) são referenciados pelas letras A, H, E e L, respectivamente. O MSE (Seção 4.3) é a função de perda usada para avaliar a precisão do treinamento da rede e dos valores de previsão.

Na Tabela 9.1, o primeiro hiperparâmetro define o algoritmo de otimização em relação ao cálculo do gradiente descendente para treinamento da rede. No mecanismo proposto, podem ser utilizados o algoritmo padrão *Stochastic Gradient Descent* SGD (Seção 5.1.5), algoritmo ADAM (Seção 5.2.6) ou NADAM (Seção 5.2.7). O segundo hiperparâmetro define a quantidade de nós das camadas ocultas LSTM e densa (no caso da arquitetura tipo 3) e pode assumir os valores de 16 ou 32 nós. O terceiro hiperparâmetro define a quantidade de épocas nas quais a rede neural processa todos os dados de treinamento e

Tabela 9.1: Valores possíveis dos hiperparâmetros de configuração.

| # | Hiperparâmetro | Símbolo | Valores Possíveis |
|---|-------------------------|---------|-----------------------------------|
| 1 | Algoritmo de otimização | A | A1 (SGD), A2 (ADAM) ou A3 (NADAM) |
| 2 | Nós nas camadas ocultas | H | 16 ou 32 |
| 3 | Épocas de treinamento | E | 50, 100 ou 200 |
| 4 | Taxa de aprendizado | L | 0,005 ou 0,01 |

calcula os pesos das conexões através do algoritmo do gradiente descendente (Seção 5.1.4). Poucas épocas podem ser insuficientes para reter as informações relevantes e, por outro lado, muitas épocas podem causar ajustes excessivos dos pesos e perda de precisão (Seção 5.2). Esses dois últimos hiperparâmetros impactam diretamente no tempo de treinamento da rede. O quarto hiperparâmetro define o valor da taxa de aprendizado usada pelos nós LSTM. Essa taxa controla como o modelo se adapta em resposta à propagação reversa dos erros (Seção 5.1.3), cada vez que os pesos das conexões são atualizados.

A combinação dos valores dos quatro hiperparâmetros, considerando ainda os 3 *layouts* de arquitetura (Seção 9.1.1), resulta em 108 configurações possíveis de rede. Nos experimentos realizados, o espaço de possibilidades foi reduzido para 10 configurações através da observação dos valores de MSE em testes empíricos, que consideraram diferentes tipos de instâncias e períodos do histórico de preços durante o ano de 2020. Tais dados foram obtidos da Amazon EC2 [24] e possuem os seguintes campos: nome do tipo de instância, nome do produto (sistema operacional), região/zona de disponibilidade, preço e registro de data e hora. Esses dois últimos campos são processados como uma série temporal e os outros campos são descartados.

Para a utilização da rede neural recorrente, os dados foram primeiramente regularizados considerando as variações de preços uniformemente distribuídas em intervalos de tempo de quatro horas. Os dados foram então separados em conjuntos de treinamento e teste, com as proporções de 70% e 30%, respectivamente, em relação ao conjunto de dados original. Finalmente, um procedimento de normalização Min-Max (Seção 5.1.7) é aplicado aos dados em ambos os conjuntos, convertendo os preços em valores entre 0 e 1.

9.2 Extensão do Mecanismo de Utilidade para Cálculo da Disponibilidade Real

O mecanismo de análise de utilidade com média móvel e janelas deslissantes foi detalhado no Capítulo 8. Esta Seção define a proposta de avaliação real da disponibilidade da ins-

tância *spot* em função do lance, estendendo assim o mecanismo de utilidade anteriormente definido.

A avaliação de disponibilidade real em função do lance consiste em analisar o tempo de disponibilidade e o custo da instância durante uma janela de tempo predefinida no futuro, caso o usuário utilize o valor do lance sugerido pelo Algoritmo 5 (Capítulo 8) para solicitar uma instância *spot*. Esta avaliação é realizada em termos de durabilidade [35], neste caso expressa como o período de tempo em que a instância ficou disponível antes que uma variação do preço *spot* ultrapassasse o valor da oferta ou fosse atingido o período limite predefinido. A disponibilidade da instância é considerada dentro de um período de tempo w , e, no caso do mecanismo de utilidade, é igual a 12 horas ou o mesmo tamanho de janela usado pelo Algoritmo 6.

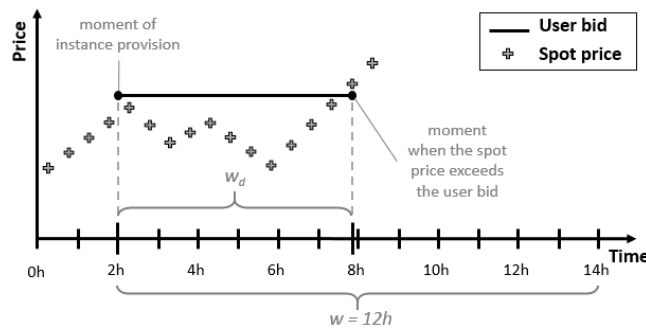


Figura 9.4: Análise futura de lance e disponibilidade.

A Figura 9.4 mostra um momento hipotético inicial de disponibilização de instância, quando a solicitação do usuário é atendida devido a um lance bem sucedido. A janela de tempo w_d compreende esse instante inicial até o momento em que a instância é revogada, devido ao fato do preço *spot* ultrapassar o valor de lance estabelecido inicialmente na solicitação. Com essa informação é possível avaliar as garantias de disponibilidade da instância dentro de um intervalo de confiança (Seção 4.3). A avaliação também é útil para responder à pergunta sobre qual percentual dos lances proporcionam disponibilidade em tempo integral como parte de um acordo de nível de serviço, considerando um período de avaliação e uma janela de tempo pré-definida.

9.3 Experimentos Realizados

Os experimentos realizados com mecanismos LSTM e de utilidade usaram dados de 8 tipos de instância disponíveis nos modelos de preços *on demand* e *spot* da Amazon EC2. As instâncias possuem diferentes propósitos de utilização, como processamento intensivo, uso de memória e otimização para armazenamento. Os históricos de preços se referem ao período de janeiro a junho de 2020, na região US-East-1 (North Virginia, EUA), com o

sistema operacional Linux/UNIX. Os tipos de instância e suas características computacionais (vCPU e memória), objetivos, preços *on demand* (OD) e o número de ocorrências de variação do preço *spot* durante o período de avaliação são detalhados na Tabela 9.2.

Tabela 9.2: Tipos de instância utilizados nos experimentos com o mecanismo combinado.

| Tipo de Instância | vCPU | Mem. (GB) | Propósito | Preço OD (USD/h) | # Variações de Preço <i>Spot</i> |
|--------------------------|-------------|------------------|------------------|-------------------------|---|
| c5n.2xlarge | 8 | 21 | Compute | 0,432 | 565 |
| c5n.9xlarge | 36 | 96 | Compute | 1,944 | 526 |
| i3.2xlarge | 8 | 61 | Storage | 0,624 | 540 |
| i3.8xlarge | 32 | 244 | Storage | 2,496 | 616 |
| m5.2xlarge | 8 | 32 | General | 0,384 | 580 |
| m5.4xlarge | 16 | 64 | General | 0,768 | 600 |
| r4.2xlarge | 8 | 61 | Memory | 0,532 | 434 |
| r5.2xlarge | 8 | 64 | Memory | 0,504 | 514 |

9.3.1 Experimentos com Mecanismo LSTM

A avaliação do mecanismo LSTM considerou períodos mensais, de janeiro a junho de 2020. Em cada período mensal, 70% dos dados do histórico de precificação são usados para treinar a rede e 30% para teste dos valores previstos. As configurações e os tipos de rede foram avaliados para cada mês e tipo de instância, de forma a possibilitar, em cada execução, a coleta dos valores de erro (MSE) de treinamento e teste. Assim, o conjunto original de combinações de cenários levou em consideração os valores de hiperparâmetros de configuração descritos na Tabela 9.1, os tipos de *layout* de arquitetura de rede definidos na Seção 9.1, os tipos de instância relacionados na Tabela 9.2 e os seis primeiros meses de 2020.

Para comparar as configurações e tipos de rede, foram identificadas as combinações com os menores valores de MSE de treinamento e teste por tipo de instância e período mensal. A primeira coluna da Tabela 9.3 apresenta a configuração e o tipo de rede, conforme a codificação dos hiperparâmetros (Tabela 9.1). Adicionalmente, os parâmetros T2 e T3 referem-se aos *layouts* arquiteturais de rede dos tipos 2 (Figura 9.2) e 3 (Figura 9.2), respectivamente. A segunda coluna mostra o número de ocorrências em que cada configuração/tipo de rede obteve o menor MSE de treinamento para um tipo de instância distinto e período mensal. A terceira coluna apresenta o número de vezes em que cada configuração/tipo de rede atingiu o MSE mais baixo durante o período de teste. As colunas 2 e 3 totalizam 48 ocorrências cada (ou 6 meses vezes 8 tipos de instância).

Ainda na Tabela 9.3, uma classificação mais ampla que cobre todos os tipos de instância e períodos mensais foi aplicada ao espaço de possíveis configurações e tipos de rede,

Tabela 9.3: Análise de MSE e MSE-limite por configuração e tipo de rede LSTM.

| Configuração e Tipo de rede (AHELT) | # Menor MSE de Treinamento | # Menor MSE de Teste | # Abaixo do Limite de Treinamento | # Abaixo do Limite de Teste | # Abaixo de Ambos os Limites |
|-------------------------------------|----------------------------|----------------------|-----------------------------------|-----------------------------|------------------------------|
| A1H16E50L0.005T2 | 0 | 0 | 0 | 17 | 0 |
| A1H16E50L0.005T3 | 0 | 1 | 5 | 30 | 5 |
| A2H16E100L0.005T2 | 8 | 7 | 27 | 39 | 23 |
| A2H16E100L0.005T3 | 10 | 2 | 23 | 35 | 18 |
| A2H32E200L0.005T2 | 14 | 11 | 30 | 40 | 27 |
| A2H32E200L0.005T3 | 15 | 6 | 23 | 31 | 18 |
| A3H16E100L0.005T2 | 0 | 16 | 0 | 34 | 0 |
| A3H16E100L0.005T3 | 1 | 2 | 7 | 30 | 3 |
| A3H32E200L0.005T2 | 0 | 2 | 0 | 26 | 0 |
| A3H32E200L0.005T3 | 0 | 1 | 2 | 27 | 0 |
| Total | 48 | 48 | 117 | 309 | 94 |

considerando limites aplicados aos valores de MSE de treinamento e teste. Desta forma, a quarta coluna da Tabela 9.3 mostra o número de ocorrências para cada configuração de rede cujo valor de MSE de treinamento é inferior ao limite 0,05. Este conjunto reduzido possui um total de 117 ocorrências. A quinta coluna apresenta a quantidade de vezes em que o valor do MSE de teste é inferior ao limite 0,0001 para cada configuração de rede, totalizando 309 ocorrências. A última coluna mostra a quantidade de vezes em que os valores de MSE de treinamento e teste estão abaixo de ambos os limites para cada configuração, totalizando 94 ocorrências. Por exemplo, a configuração A2H32E200L0.005T3 aparece 15 vezes com o menor valor de MSE de treinamento, e 6 vezes com o menor valor MSE de teste dentre todas as demais configurações. Em 44 combinações de meses e tipos de instância, esta configuração apresenta MSE de treinamento abaixo de 0,05, e em 31 combinações apresenta o MSE de teste abaixo de 0,0001. Finalmente, esta configuração apresenta valores de MSE de treinamento e teste abaixo de ambos os limites em 18 combinações de tipos de meses e tipos de instância.

Os algoritmos ADAM (A2) e NADAM (A3), assim como as redes T2 (intermediário) e T3 (sofisticado), compõem as configurações predominantes na Tabela 9.3. Os melhores resultados de MSE de teste, considerando ambos os limites, são apresentados na Tabela 9.4. Nas colunas 2, 3, 4 e 5, os meses e os valores de MSE de treinamento e teste, assim como as configurações de rede, são apresentados para cada tipo de instância. Janeiro e junho são os meses que mais aparecem com menores erros de previsão de preços. A Tabela 9.5 apresenta os piores resultados de MSE durante o período de teste, considerando valores abaixo do limite de 0,0001, para cada tipo de instância. Março, abril e maio são os meses que mais aparecem com o maior erro de previsão.

Não há configuração de rede predominante entre os resultados da Tabela 9.4, ou configuração que deva ser descartada a partir dos resultados da Tabela 9.5. Embora trabalhos como [25] tenham fixado uma configuração semelhante a A3H32E200L0.005T3, não foi possível estabelecer uma configuração de rede definitiva para todos os cenários.

Tabela 9.4: Melhores resultados de MSE no período de teste por tipo de instância.

| Tipo de Instância | Melhores Resultados de MSE de Teste | | | |
|-------------------|-------------------------------------|--------------------|--------------|----------------------|
| | Mês | MSE de Treinamento | MSE de Teste | Configuração da Rede |
| c5n.2xlarge | Janeiro | 0,01246518 | 0,00000050 | A2H16E100L0.005T2 |
| c5n.9xlarge | Janeiro | 0,01250304 | 0,00003097 | A2H32E200L0.005T3 |
| i3.2xlarge | Janeiro | 0,02640755 | 0,00000006 | A3H16E100L0.005T3 |
| i3.8xlarge | Janeiro | 0,00742033 | 0,00000515 | A2H32E200L0.005T3 |
| m5.2xlarge | Janeiro | 0,00620938 | 0,00000028 | A2H32E200L0.005T2 |
| m5.4xlarge | Junho | 0,01128134 | 0,00000044 | A2H16E100L0.005T2 |
| r4.2xlarge | Janeiro | 0,00266161 | 0,00000006 | A2H32E200L0.005T2 |
| r5.2xlarge | Junho | 0,02391096 | 0,00000013 | A3H16E100L0.005T3 |

Tabela 9.5: Piores resultados de MSE no período de teste abaixo do limite pré-estabelecido ($MSE \leq 0,0001$).

| Tipo de Instância | Piores Resultados de MSE de Teste Abaixo do Limite ($MSE \leq 0,0001$) | | | |
|-------------------|--|--------------------|--------------|----------------------|
| | Mês | MSE de Treinamento | MSE de Teste | Configuração da Rede |
| c5n.2xlarge | Abril | 0,00654776 | 0,00007528 | A2H16E100L0.005T3 |
| c5n.9xlarge | Janeiro | 0,01651889 | 0,00009087 | A2H32E200L0.005T2 |
| i3.2xlarge | Maior | 0,01329307 | 0,00009633 | A2H16E100L0.005T3 |
| i3.8xlarge | Maior | 0,02622799 | 0,00007793 | A1H16E50L0.005T3 |
| m5.2xlarge | Março | 0,00343944 | 0,00008868 | A2H32E200L0.005T2 |
| m5.4xlarge | Março | 0,00773649 | 0,00007488 | A2H32E200L0.005T2 |
| r4.2xlarge | Abril | 0,00895758 | 0,00007304 | A1H16E50L0.005T3 |
| r5.2xlarge | Março | 0,00825468 | 0,00009342 | A2H16E100L0.005T2 |

A avaliação do tipo de instância r4.2xlarge com a configuração A2H32E200L0.005T2, durante o mês de janeiro de 2020, é detalhada na Figura 9.5. A linha azul mostra as variações de preços no período de treinamento. A linha vermelha apresenta os preços reais durante o período de teste e a linha verde mostra os valores previstos pela rede LSTM neste mesmo período. O cálculo do MSE de teste se dá pela Equação 4.14, considerando os valores reais e previstos apresentados nas linhas vermelha e verde, respectivamente. Na fase de previsão, o MSE de teste é igual a $6e-8$, conforme listado na Tabela 9.4. A variação de preço *spot* inicia com uma tendência de queda e muda para um cenário de alta de preços a partir da segunda semana. A previsão capta essa tendência e acompanha a alta dos preços até o final do mês.

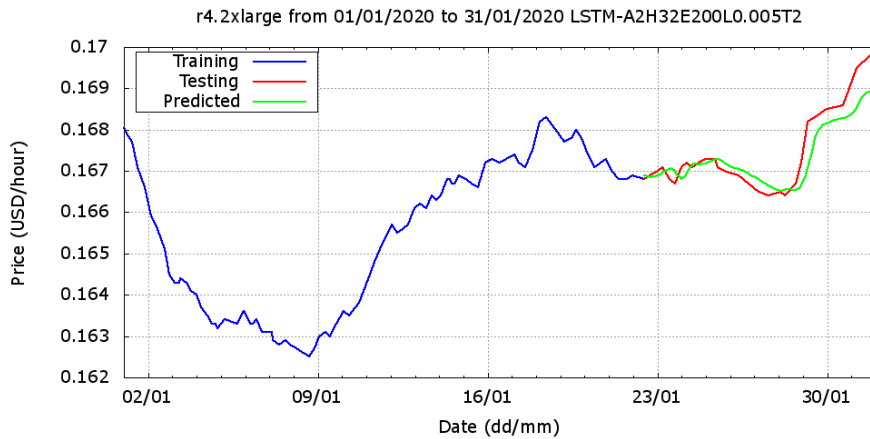


Figura 9.5: Previsão LSTM para a instância r4.2xlarge em Janeiro de 2020.

Um cenário mais complexo durante o mês de fevereiro de 2020 para r5.2xlarge é ilustrado na Figura 9.6. A configuração de rede A2H16E100L0.005T2 apresenta o MSE de teste mais baixo para esta combinação de mês e tipo de instância. Os preços apresentam trajetória ascendente até o 10º dia, quando os valores caem para patamares inferiores aos preços registrados no início do mês. O terço restante do mês mostra uma recuperação dos preços, e a previsão do LSTM segue essa tendência até o último dia.

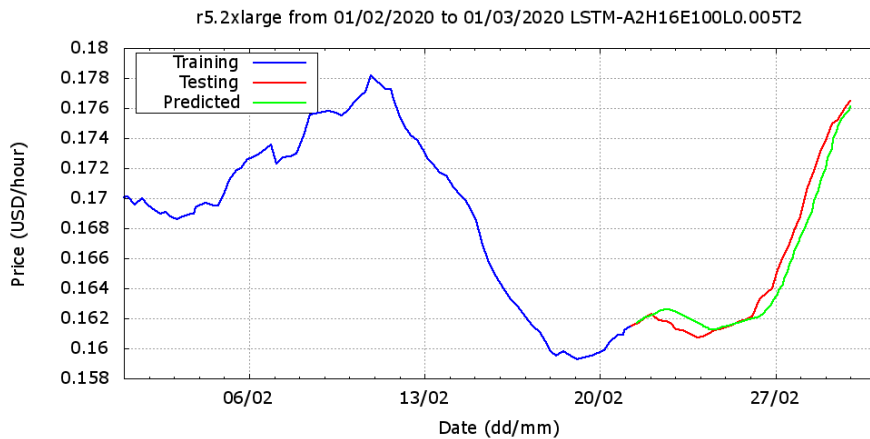


Figura 9.6: Previsão LSTM para a instância r5.2xlarge em Fevereiro de 2020.

A avaliação do tipo de instância m5.4xlarge durante o mês de março de 2020 é ilustrada na Figura 9.7. A configuração A2H32E200L0.005T3 apresenta o menor MSE de teste para este cenário de instância e mês, que é praticamente uniforme com tendência de aumento até os últimos dias do mês, quando há variações consideráveis de preços que terminam com uma queda subsequente. A previsão fornecida pela rede LSTM acompanha as variações dos preços e a tendência de queda ocorrida no final do mês.

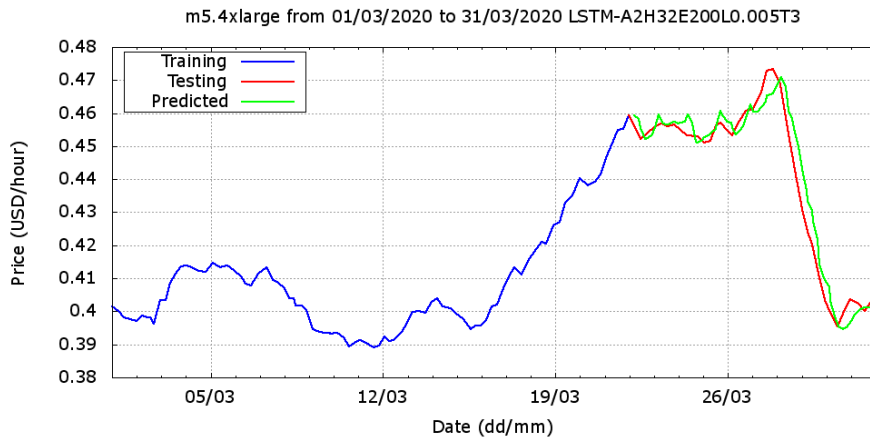


Figura 9.7: Previsão LSTM para a instância m5.4xlarge em Março de 2020.

A Figura 9.8 ilustra o pior cenário de teste para o tipo de instância c5n.2xlarge. Ele acontece em abril de 2020 com a configuração da rede A2H16E100L0.005T3, conforme mostrado na Tabela 9.5, com valor de MSE de teste igual a $7,35e-5$. Mesmo sendo o pior cenário sob os limites de treinamento e teste predefinidos, a previsão LSTM é capaz de capturar a tendência de alta dos preços até meados da última semana, quando ocorre queda de os preços, causando aumento do MSE de teste no final do mês.

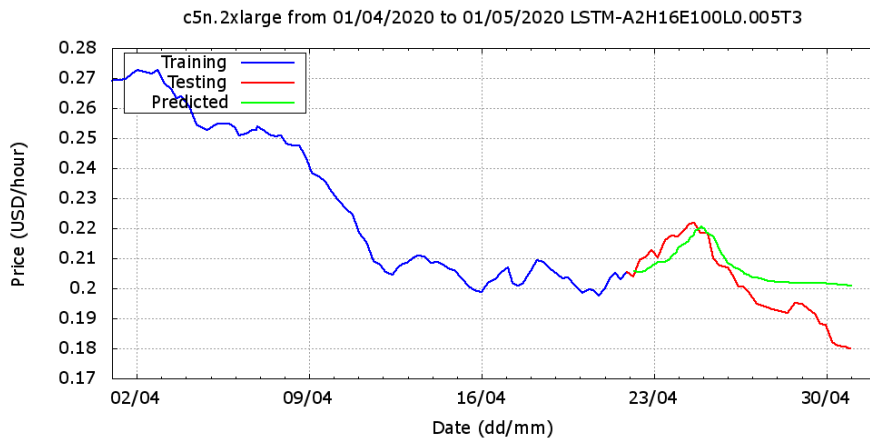


Figura 9.8: Previsão LSTM para a instância c5n.2xlarge em Abril de 2020.

A avaliação do tipo de instância i3.8xlarge, durante o mês de maio de 2020, apresenta um cenário altamente variável. É usada a configuração de rede A2H32E200L0.005T3. Conforme mostrado na Figura 9.9, as variações com tendência de alta ocorrem até o final da terceira semana, quando se inicia uma tendência de queda. A previsão para o restante do mês segue esta tendência, apesar da detecção de variações bruscas.

A Figura 9.10 mostra o cenário de avaliação para o tipo de instância r5.2xlarge usando a configuração A3H16E100L0.005T3, durante o mês de junho de 2020. O valor de MSE de

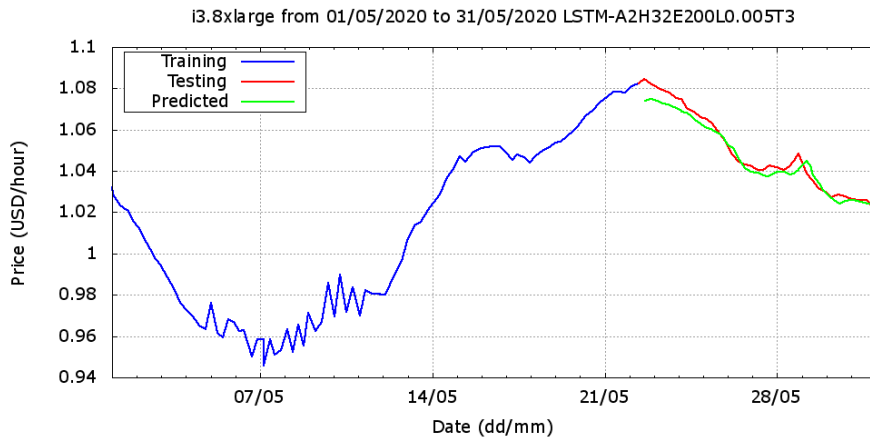


Figura 9.9: Previsão LSTM para a instância i3.8xlarge em Maio de 2020.

teste é igual a $1,3e-7$, como listado na Tabela 9.4. Este também é um cenário complexo, com tendência de crescimento do início do mês até o final da terceira semana. A previsão do LSTM é capaz de prever as variações a partir desse patamar, quando os preços *spot* começam a cair.

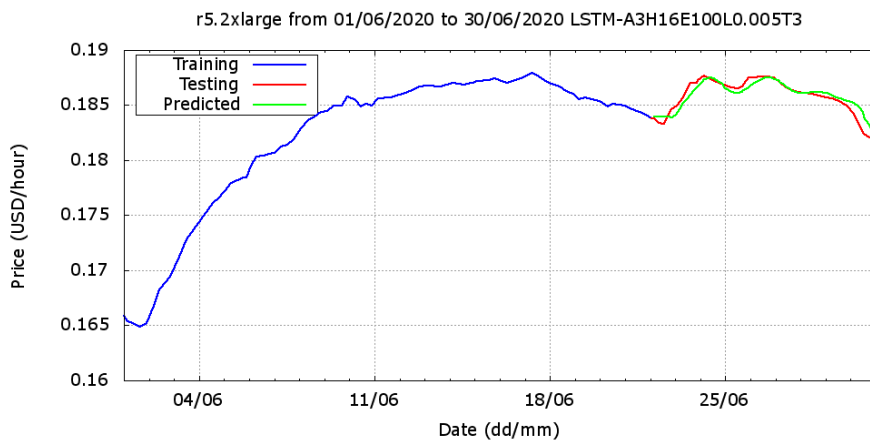


Figura 9.10: Previsão LSTM para a instância r5.2xlarge em Maio de 2020.

De forma geral, os resultados dos experimentos mostram que a rede LSTM é eficaz para a previsão de tendências preços no mercado *spot* da Amazon EC2. Apesar da necessidade de tempo e esforço consideráveis para definir a arquitetura e configurar os hiperparâmetros, os resultados apresentam erro reduzido para previsões de preços de longo prazo. O mecanismo é capaz de detectar tendências de aumentos e quedas de preços, bem como grandes variações em cenários de grande incerteza.

9.3.2 Experimentos com Mecanismo de Utilidade (Preços de 2020)

A avaliação do mecanismo baseado em utilidade considera históricos de precificação de instância divididos em dois conjuntos de dados: para o primeiro (Q1) e o segundo (Q2) trimestres de 2020. As Tabelas 9.6 e 9.7 detalham os resultados experimentais para cada tipo de instância, durante o primeiro e segundo trimestres, respectivamente.

Tabela 9.6: Resultados experimentais do mecanismo de utilidade durante o primeiro trimestre (Q1) de 2020.

| Tipo de Instância | Preço Médio (USD/h) | Lance Preço (USD/h) | w_d 5% (95%CI) Percentil (h) | w_d 10% (95%CI) Percentil (h) | Durabilidade $\geq 12h$ |
|-------------------|---------------------|---------------------|--------------------------------|---------------------------------|-------------------------|
| c5n.2xlarge | 0,176 | 0,176 | 5,31 (4,89 - 5,58) | 5,87 (5,57 - 6,16) | 63,56% |
| c5n.9xlarge | 0,786 | 0,789 | 4,92 (4,75 - 5,17) | 5,31 (5,17 - 5,46) | 53,48% |
| i3.2xlarge | 0,214 | 0,213 | 5,44 (4,76 - 5,85) | 6,33 (5,72 - 6,70) | 73,75% |
| i3.8xlarge | 0,862 | 0,858 | 4,91 (4,75 - 5,18) | 5,57 (5,16 - 5,73) | 58,70% |
| m5.2xlarge | 0,183 | 0,183 | 5,17 (4,89 - 5,34) | 5,73 (5,30 - 6,14) | 69,32% |
| m5.4xlarge | 0,382 | 0,380 | 5,02 (4,75 - 5,17) | 5,58 (5,17 - 5,72) | 58,63% |
| r4.2xlarge | 0,185 | 0,184 | 5,17 (4,89 - 5,70) | 5,87 (5,59 - 6,28) | 68,39% |
| r5.2xlarge | 0,193 | 0,192 | 4,76 (4,62 - 5,03) | 5,30 (5,02 - 5,54) | 57,75% |

As médias de preços *spot* são apresentados na segunda coluna. As sugestões de oferta e disponibilidade são calculadas a cada momento em que ocorre uma variação do preço *spot* utilizando o Algoritmo 5. As sugestões de lance médio são apresentadas na terceira coluna, enquanto a disponibilidade esperada com base na Equação 8.2 é sempre 100% para os cenários analisados. A quarta e quinta colunas apresentam os percentis de 5% e 10% da janela de durabilidade w_d em horas, definida na Seção 9.2, com seus respectivos intervalos de confiança de 95% em Q1 e Q2. A sexta e última coluna das tabelas apresenta os percentuais de lances que proporcionaram 12 horas ou mais de disponibilidade da instância nos trimestres.

Para o tipo de instância c5n.9xlarge, as sugestões de lance médio no primeiro e segundo trimestres foram, respectivamente, 0,789 (Tabela 9.6) e 0,788 USD/hora (Tabela 9.7). Tais valores estão ligeiramente acima dos preços *spot* médios e aproximadamente equivalente a 40% do preço *on demand*. A avaliação da disponibilidade do lance durante esses períodos indicam, com 95% de confiança, que 95% das sugestões de lances fornecidas pelo mecanismo da concessionária garantem pelo menos 4,75 e 4,61 horas de disponibilidade de instâncias para o primeiro e segundo trimestre, respectivamente. Também é possível concluir, com 95% de confiança, que 90% das sugestões de lance garantem pelo menos 5,17 e 4,9 horas para o primeiro e segundo trimestre, respectivamente. A avaliação da

Tabela 9.7: Resultados experimentais do mecanismo de utilidade durante o segundo semestre (Q2) de 2020.

| Tipo de Instância | Preço Médio (USD/h) | Lance Preço (USD/h) | w_d 5% (95%CI) Percentil (h) | w_d 10% (95%CI) Percentil (h) | Durabilidade $\geq 12h$ |
|-------------------|---------------------|---------------------|--------------------------------|---------------------------------|-------------------------|
| c5n.2xlarge | 0,234 | 0,238 | 5,06 (4,75 - 5,31) | 5,45 (5,30 - 5,60) | 58,54% |
| c5n.9xlarge | 0,791 | 0,788 | 4,76 (4,61 - 4,91) | 5,18 (4,90 - 5,32) | 48,14% |
| i3.2xlarge | 0,257 | 0,257 | 5,38 (5,17 - 5,59) | 5,88 (5,59 - 6,44) | 70,90% |
| i3.8xlarge | 1,014 | 1,014 | 5,09 (4,76 - 5,31) | 5,59 (5,31 - 5,87) | 61,19% |
| m5.2xlarge | 0,186 | 0,190 | 5,45 (5,03 - 5,74) | 6,36 (5,73 - 6,99) | 77,27% |
| m5.4xlarge | 0,361 | 0,365 | 5,17 (4,89 - 5,59) | 6,01 (5,59 - 6,46) | 73,88% |
| r4.2xlarge | 0,200 | 0,204 | 5,59 (5,02 - 5,91) | 6,43 (5,73 - 6,85) | 78,26% |
| r5.2xlarge | 0,213 | 0,218 | 5,45 (5,18 - 5,72) | 6,14 (5,67 - 6,71) | 74,65% |

durabilidade da instância indica que 53,48% e 48,14% das sugestões de lance para Q1 e Q2 garantem 12 horas ou mais de disponibilidade da instância, respectivamente. Esses cenários são ilustrados nas Figuras 9.11 e 9.12.

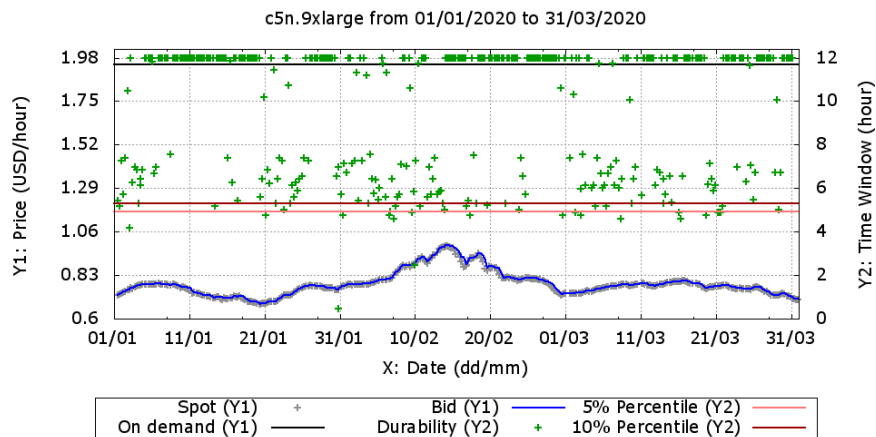


Figura 9.11: Lance e disponibilidade da instância c5n.9xlarge no 1º semestre de 2020.

Na avaliação para o tipo de instância r4.2xlarge, as sugestões de lance médio foram de 0,184 e 0,204 USD/hora no primeiro e segundo trimestre, valores aproximadamente equivalentes a 35% e 38%, respectivamente, do preço *on demand*. Com 95% de confiança, é possível concluir que 95% das sugestões de lance garantem pelo menos 4,89 e 5,02 horas de disponibilidade da instância para o primeiro e segundo trimestres. Também pode ser concluído, com 95% de confiança, que 90% das sugestões de lance garantem pelo menos 5,59 e 5,73 horas para o primeiro e segundo trimestres. Além disso, 68,39% e 78,26% de sugestões de lance garantem 12 horas ou mais de disponibilidade para o primeiro e segundo trimestre. Esses foram os maiores valores de durabilidade entre os tipos de

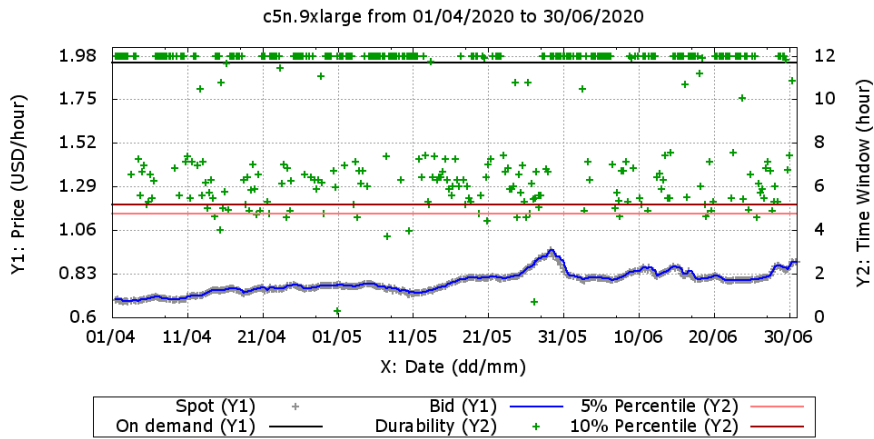


Figura 9.12: Lance e disponibilidade da instância c5n.9xlarge no 2º semestre de 2020.

instância analisados, embora tenha havido grandes variações de preços no final do Q1, de 17 de março até meados de maio, conforme detalhado nas Figuras 9.13 e 9.14.

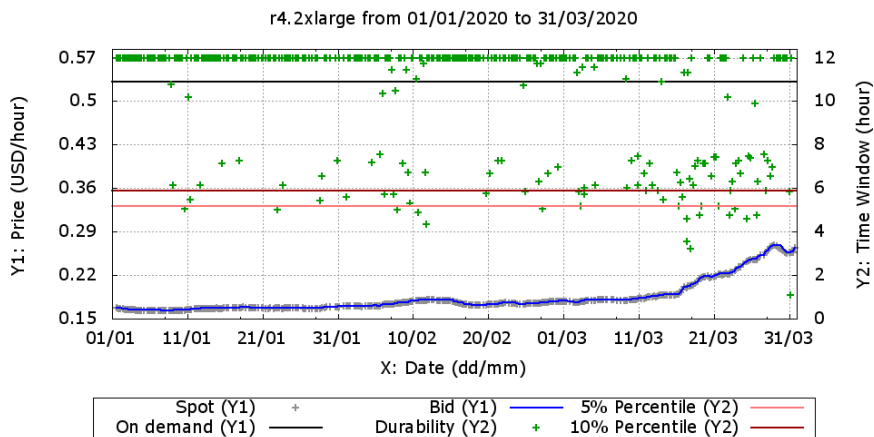


Figura 9.13: Lance e disponibilidade da instância r4.2xlarge no 1º semestre de 2020.

Por um lado, o tipo de instância r4.2xlarge apresenta maiores valores percentuais, portanto com maior capacidade de prover disponibilidade real. A durabilidade de r4.2xlarge também é maior do que outros tipos de instância no 2º trimestre de 2020. Por outro lado, a avaliação do tipo de instância c5n.9xlarge mostra os valores mais baixos para os percentis e durabilidade nas Tabelas 9.6 e 9.7. Na avaliação geral, os tipos de instância otimizadas para uso intensivo de memória e de propósito geral apresentaram percentis e valores de durabilidade mais altos do que as instâncias otimizadas para computação e armazenamento, exceto para a instância i3.2xlarge, que teve a maior durabilidade no primeiro trimestre.

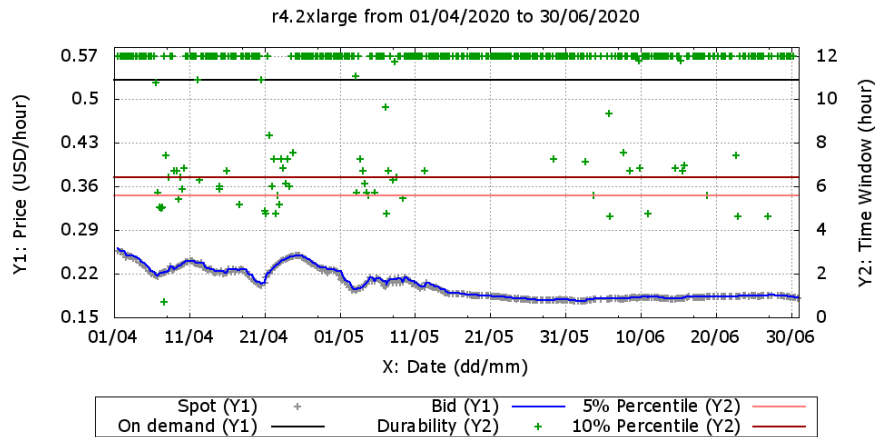


Figura 9.14: Lance e disponibilidade da instância r4.2xlarge no 2º semestre de 2020.

9.4 Metodologia para Uso Conjunto dos Mecanismos LSTM e Utilidade

Para o uso combinado do mecanismo baseado em utilidade para balanceamento de custo e disponibilidade, apresentado no Capítulo 8, com o mecanismo para análise de tendências de preços baseado em redes LSTM, proposto na Seção 9.1, foi definida uma metodologia para contratação de um tipo de instância *spot*, composta pelos seguintes passos:

1. **Pré-seleção de instâncias:** realizar a pré-seleção de um subconjunto de tipos de instância que atendem às necessidades da aplicação do usuário (por exemplo a quantidade mínima de CPU e memória, o custo máximo por hora, a zona de disponibilidade ou a região específica);
2. **Previsão LSTM:** para cada tipo de instância pré-selecionada, aplicar o mecanismo LSTM (Seção 9.1) para previsão de preços de longo prazo, neste caso uma semana (7 dias) à frente, utilizando dados de histórico de precificação da instância dos últimos 23 dias;
3. **Análise de tendências:** para as previsões geradas por meio do mecanismo LSTM, realizar uma análise de tendência de preços considerando critérios como amplitude de variações e tendência geral de queda de preços, dentre outros critérios;
4. **Seleção da instância apropriada:** seleção do tipo de instância que melhor se adequa aos critérios utilizados na análise de tendências (passo 3);
5. **Análise de utilidade:** obtenção do valor atual de sugestão de lance a partir do mecanismo baseado em utilidade, considerando o histórico de precificação da instância nas últimas 12 horas;

6. **Requisição da instância *spot*:** registro da requisição para o tipo de instância selecionado, utilizando a sugestão de lance (passo 5) como valor máximo a ser pago por hora.

A realização da previsão LSTM (passo 2) pode ser simplificada, com a utilização de configurações de rede LSTM pré-determinadas, como as mencionadas na Tabela 9.4. A análise de tendências (passo 3) pode considerar também o valor de MSE de treinamento da rede LSTM como critério para escolha do tipo de instância, apesar de não haver relação direta entre o erro na etapa de treinamento e na etapa de teste. Na análise de utilidade (passo 5), pode ser utilizada a sugestão de lance médio ou a maior sugestão de lance nos últimos 30 dias, como alternativas mais conservadoras, ao invés da sugestão de lance com base nas últimas 12 horas, conforme previsto no Algoritmo 6 (Capítulo 8).

9.5 Estudo de Caso: Comparação de Sequências Biológicas SARS-CoV-2

A doença COVID-19, causada pelo vírus SARS-CoV-2, se espalhou por todo o mundo em 2020 e se tornou uma pandemia de enormes proporções. Em setembro de 2020, uma linhagem mais contagiosa de SARS-CoV-2 chamada B.1.1.7 foi identificada no Reino Unido. Para entender esta doença, os pesquisadores desenvolveram vários estudos, que incluem, dentre outras técnicas, a comparação de sequências biológicas do SARS-CoV-2. Nesta seção, foi aplicada a nossa abordagem combinada de LSTM e utilidade para executar uma aplicação de Bioinformática real que compara sequências de DNA da linhagem SARS-CoV-2 B.1.1.7, usando instâncias *spot* do serviço Amazon EC2.

A ferramenta MASA-OpenMP¹ foi executada para obter alinhamentos globais ótimos entre as sequências de DNA SARS-CoV-2 (no hospedeiro humano), usando todos os núcleos disponíveis [128]. Para isso, foi usada a sequência de referência SARS-CoV-2 (número de acesso NC_045512.2)² para comparação com as sequências SARS-CoV-2 da linhagem B.1.1.7³. Primeiro, foram comparadas todas as sequências B.1.1.7 com a sequência de referência (4279 comparações). Em seguida, comparou-se as 50 sequências B.1.1.7 mais recentes com outras sequências da mesma linhagem (213900 comparações). O objetivo do estudo de caso é (a) recuperar alinhamentos entre a sequência de referência SARS-CoV-2 e a linhagem B.1.1.7, e (b) recuperar alinhamentos entre as sequências B.1.1.7. Para isso, definiu-se o tempo máximo de execução para 12 horas para esta aplicação (218179 comparações). A aplicação cliente/servidor desenvolvida para a execução

¹Disponível em <http://github.com/edanssandres/MASA-OpenMP>

²Disponível em <http://www.ncbi.nlm.nih.gov>

³Disponível em <http://www.gisaid.org>

de todas essas comparações em instâncias *spot* com a ferramenta MASA-OpenMP está disponível no repositório *FASTA Compare* ⁴ do GitHub.

A metodologia descrita na Seção 9.4 foi aplicada. Primeiro, foi executada a abordagem LSTM para as instâncias na Tabela 9.2, usando dados históricos de dezembro de 2020 a janeiro de 2021 (23 dias) para treinar a rede LSTM. Em seguida, foram obtidas as tendências de preço *spot* para uma semana à frente. Essas previsões foram usadas para escolher quatro tipos de instância, considerando variação de preços a tendência geral de queda como critérios. Os tipos de instância m5.2xlarge, c5n.9xlarge, i3.2xlarge e r5.2xlarge foram escolhidos.

Depois de analisar várias configurações, foi escolhida a configuração A2H32E200L0.005T2 para a instância m5.2xlarge, no período de teste de 13/01 a 21/01/2021. Conforme mostrado na Figura 9.15, para este cenário, a previsão LSTM indica um aumento no preço, com uma tendência de queda subsequente. A Figura 9.17 mostra a previsão de preços para a instância i3.2xlarge, no período de 16/01 a 24/01/2021, utilizando a configuração A2H16E100L0005T2. Nesse caso, os preços previstos são bastante estáveis, com baixa variação.

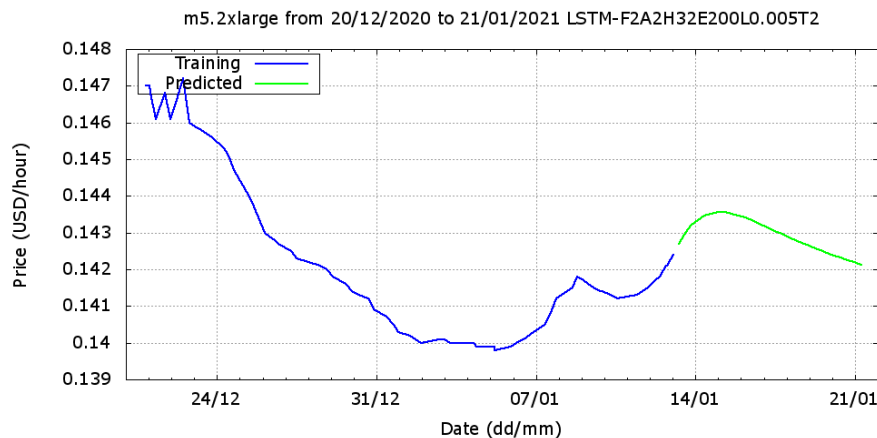


Figura 9.15: Tendência de preços do mecanismo LSTM para a instância m5.2xlarge.

Em seguida, foi executado o mecanismo de estimativa de preço e disponibilidade baseado em utilidade (Seção 8.1) para obter a sugestão de lance para as instâncias *spot* escolhidas. A abordagem com preço máximo baseado em utilidade (*utility*) foi comparada com duas outras abordagens, que são (a) preço atual (*current*): definiu-se o preço máximo como o preço *spot* atual, a fim de verificar se as variações do preço *spot* são tão pequenas que definir o preço atual como o preço máximo nos permitiria executar a instância a um preço muito baixo e sem interrupções por um período de tempo relevante;

⁴Disponível em <http://github.com/gjportella/fastacompare>

e (b) *default*: não foi definido o preço máximo *spot*, ou seja, sem restrição ao preço a ser pago pela instância.

Tabela 9.8: Resultados detalhados do estudo de caso.

| Instância (Data) | Abordagem Spot | Preço Máx. | Total USD | #Cmp. | Tempo Total | USD/#Cmp. | Tempo/#Cmp. |
|---------------------------|-----------------------|-------------------|------------------|--------------|--------------------|------------------|--------------------|
| m5.2xlarge 13/01/2021 | Utility | 0,152 | 1,43 | 43560 | 08:49:59 | 0,00003283 | 0,730 |
| | Current | 0,143 | 0,29 | 5420 | 01:06:42 | 0,00005351 | 0,738 |
| | Default | - | 1,29 | 37701 | 07:44:56 | 0,00003422 | 0,740 |
| c5n.9xlarge 15/01/2021 | Utility | 0,753 | 9,03 | 138804 | 11:41:01 | 0,00006506 | 0,303 |
| | Current | 0,752 | 3,76 | 59945 | 05:01:38 | 0,00006272 | 0,302 |
| | Default | - | 3,76 | 58993 | 04:56:06 | 0,00006374 | 0,301 |
| i3.2xlarge 16/01/2021 | Utility | 0,201 | 2,39 | 45678 | 11:36:29 | 0,00005232 | 0,915 |
| | Current | 0,200 | 2,39 | 46427 | 11:46:43 | 0,00005148 | 0,913 |
| | Default | - | 2,39 | 45461 | 11:36:21 | 0,00005257 | 0,919 |
| r5.2xlarge 17/01/2021 | Utility | 0,201 | 2,40 | 59098 | 11:47:17 | 0,00004061 | 0,718 |
| | Current | 0,200 | 2,40 | 58424 | 11:40:24 | 0,00004108 | 0,719 |
| | Default | - | 2,40 | 58458 | 11:40:26 | 0,00004106 | 0,719 |

Usando as três abordagens, ou seja, sugestão de preço máximo com base em utilidade, preço máximo atual e abordagem padrão sem preço máximo, foram realizadas 3 execuções, ao mesmo tempo, do mesmo tipo de instância. Em cada instância foi executada a ferramenta MASA-OpenMP para comparação de sequências, por um período máximo de 12h. Os resultados das 3 execuções são apresentados na Tabela 9.8. Para cada tipo de instância e abordagem, são apresentados nas colunas 3 a 7, o preço máximo *spot*, o custo total (ou seja, valor pago), o número de comparações executadas, o tempo decorrido (até a conclusão ou revogação da instância), o preço médio pago por comparação e o tempo médio por comparação.

Para a instância m5.2xlarge, o mecanismo LSTM previu um aumento no preço em 13/01/2021 (Figura 9.15) e, por esta razão, a definição do preço máximo da instância como o preço *spot* atual não foi uma boa escolha. Como consequência, instância foi revogada em menos de 2 horas de execução nessa abordagem. Ainda sobre esse caso, embora a execução tenha demorado 1 hora e 6 minutos, a Amazon EC2 cobrou por 2 horas, aumentando consideravelmente o custo por comparação. Para o mesmo tipo de instância, a abordagem que utilizou preço com base em utilidade foi revogada após execução por quase 9 horas.

O mecanismo LSTM previu poucas variações de preço para o tipo de instância c5n.9xlarge em 15/01/2021, conforme mostrado na Figura 9.16. Essa é a razão pela qual o preço com base em utilidade está muito próximo do preço que usa a abordagem de preço *spot* atual.

prazo utilizando redes LSTM, mostrou ser uma ferramenta importante para o usuário de computação em nuvem no mercado *spot* da Amazon EC2.

A avaliação por meio de cenários extensos em 2020 conseguiu fornecer informações úteis sobre sugestões de lances, bem como durabilidade das instâncias analisadas. Os resultados para o tipo de instância r4.2xlarge mostraram que as sugestões de lance médio variaram entre 35% e 38% do preço *on demand* para o primeiro e segundo trimestres, respectivamente. Com 95% de confiança, concluímos que 95% das sugestões de lance garantiram, pelo menos, 4,89 e 5,02 horas de disponibilidade da instância, e 90% das sugestões de lance, pelo menos, 5,59 e 5,73 horas. Tomando como base o conceito de durabilidade, 68,39% e 78,26% das sugestões de lance garantiram 12 horas ou mais de disponibilidade da instância, para o primeiro e segundo trimestres, respectivamente.

A proposta do mecanismo flexível baseado em redes LSTM para previsão de preço *spot* a longo prazo foi validada para diversos tipos de instâncias. Várias arquiteturas e configurações de rede foram avaliadas durante o primeiro semestre de 2020. Nosso mecanismo foi capaz de prever preços para todos os cenários com erros de treinamento e previsão muito baixos. Para o tipo de instância r5.2xlarge, durante o mês de junho de 2020, o mecanismo previu valores entre 0,18 e 0,19 USD/ ora, equivalente a 37% do preço *on demand* e próximo às variações *spot* ($MSE = 1,3e-7$).

Diante do exposto, foi aplicado o mecanismo combinado na execução de uma aplicação real de Bioinformática em instâncias *spot* do serviço Amazon EC2. A aplicação comparou cerca de 218.179 sequências de DNA de SARS-CoV-2, utilizando instâncias *spot* da Amazon EC2. Os resultados mostraram que nosso mecanismo LSTM é capaz de escolher instâncias apropriadas e que nossas sugestões de preços baseadas em utilidade são baixas, com boa disponibilidade. Mesmo em um cenário adverso, onde os preços *spot* aumentam constantemente, a instância que usou nossa sugestão de lance baseada em utilidade foi revogada após quase 9 horas de execução, possibilitando comparar um maior número de sequências do que a política *spot* AWS padrão (sem preço máximo).

A contribuição deste capítulo, que também é detalhada no trabalho [50], se insere no estado da arte em precificação em nuvem discutido no Capítulo 6, conforme é mostrado na última linha da Tabela 9.9.

Tabela 9.9: Comparação entre os trabalhos sobre precificação em computação em nuvem.

| Ref. (Ano) | Precificação | Objetivo | Grupo | Técnica | Parâmetros | Mercado |
|-------------|--------------------------|--|---------------------|---|----------------------------------|----------|
| [26] (2011) | <i>on demand</i> | escalonamento de tarefas em nuvem IaaS | OUT | <i>delay scheduling, progress share</i> | preço, memória RAM | 1 |
| [27] (2012) | <i>on demand</i> | seleção de instâncias virtuais de acordo com requisitos da aplicação | OUT | <i>ontology reasoning</i> | CPU, rede, preço, armazenamento | 1 |
| [30] (2013) | <i>spot</i> | modelagem do mercado <i>spot</i> | EST | análise de série temporal, simulação, distribuição de Pareto | preço, data | 2 |
| [29] (2013) | <i>spot</i> | modelagem do mercado <i>spot</i> | EST | análise de série temporal, análise estatística (MoG), modelagem de mercado | preço, data | 2 |
| [42] (2014) | extensível a <i>spot</i> | seleção de serviços com base em restrições de custo e qualidade | ECO | mecanismo de mercado baseado em leilões VCG | qualidade, preço, data | - |
| [28] (2015) | <i>on demand, spot</i> | leilão por instâncias <i>spot</i> | ECO | análise de série temporal, e modelagem de mercado, | preço, data | 2 |
| [31] (2015) | <i>spot</i> | estratégias de leilão por instâncias <i>spot</i> | EST | análise de série temporal, análise estatística (lognormal) | preço, data | 2 |
| [32] (2015) | <i>spot</i> | leilão por instâncias <i>spot</i> | ML | análise de série temporal, algoritmo de otimização <i>gradient descending</i> | preço, data | 2 |
| [33] (2015) | <i>on demand, spot</i> | escalonamento de instâncias com restrições | EST | análise de série temporal, médias móveis (simples, ponderada e exponencial) | preço, data, CPU, memória RAM | 2 |
| [34] (2016) | <i>on demand, spot</i> | modelagem do mercado <i>spot</i> | ECO | simulação e monitoramento de mercado | preço, data | 2 |
| [44] (2017) | <i>spot</i> | previsão de preços <i>spot</i> | ML | redes LSTM | preço, data | 2 |
| [35] (2017) | <i>spot</i> | estimativa de lance e disponibilidade | EST | série temporal, probabilidade e estatística | preço, data | 2 |
| [36] (2018) | <i>spot</i> | previsão de preços <i>spot</i> | ML | redes LSTM | preço, data | 3 |
| [25] (2018) | <i>spot</i> | previsão de preços <i>spot</i> | ML | redes LSTM | preço, data | 2 |
| [37] (2018) | <i>spot</i> | garantia de disponibilidade | EST | paralelismo, redundância e tolerância a falhas | preço, data | 2 |
| [38] (2018) | <i>spot</i> | identificação de tendências | ECO | indicadores financeiros e modelagem de riscos | preço, data | 2 |
| [39] (2018) | <i>spot</i> | previsão de preços <i>spot</i> | ECO | modelagem de série temporal | preço, data, <i>start/uptime</i> | 2 |
| [40] (2019) | <i>spot</i> | previsão de preços <i>spot</i> | EST | análise estatística e série temporal | preço, data | 2 |
| [41] (2019) | <i>spot</i> | previsão de preços <i>spot</i> | EST | análise estatística e série temporal | preço, data | 2 |
| [45] (2020) | extensível a <i>spot</i> | utilização de infraestrutura | EST & ML | análise estatística e redes LSTM | histórico de uso | - |
| [46] (2020) | <i>spot</i> | previsão de preços <i>spot</i> | ML | RRF | preço, data | 2 |
| [50] (2021) | <i>spot</i> | previsão e análise de tendência <i>spot</i> | EST & ML | utilidade (curto prazo) e LSTM (longo prazo) | preço, data | 3 |

Parte III
Conclusão

Capítulo 10

Conclusão e Trabalhos Futuros

10.1 Conclusão

Os modelos de precificação estática e dinâmica para computação em nuvem foram estudados e investigados nesta Tese, possibilitando a proposição de mecanismos para a contratação de instâncias permanentes e transientes. Tais mecanismos aplicaram técnicas de análise estatística, séries temporais, médias móveis, função de utilidade e redes neurais artificiais, objetivando a redução de custos e o aumento da disponibilidade das instâncias contratadas pelos usuários de computação em nuvem. Assim, foram realizadas diversas análises e experimentos para assegurar a eficácia dos mecanismos propostos, além da execução de uma aplicação real de bioinformática em ambiente transiente de computação em nuvem com precificação dinâmica.

Inicialmente, foram realizadas análises estatísticas do modelo de precificação *on demand* utilizado por dois provedores de nuvem pública, possibilitando a identificação dos parâmetros que compõem os preços praticados. Os resultados, discutidos no Capítulo 7 e publicados em [47], mostraram que a Equação 7.2, proposta nesta Tese, conseguiu modelar a precificação estática da Amazon EC2 com erro entre 1% e 14% (Tabela 7.3), por meio da utilização da técnica de regressão multilinear. Além disso, foi proposta a Equação 7.4, com resultados estatisticamente relevantes utilizando dados desse mesmo provedor.

Em seguida, partiu-se para o projeto de modelagem estatística da precificação *spot* da Amazon EC2, com o objetivo de determinar uma faixa razoável de lances, inferior ao preço *on demand*, proporcionando uma disponibilidade aceitável da instância contratada. Os resultados foram discutidos no Capítulo 7 e publicados em [126]. O limite superior do intervalo de confiança 95%, obtido a partir do cálculo de médias móveis com janelas deslizantes em séries temporais, se adaptou para os períodos de alta variação de preços *spot* em 2016, para os diversos tipos de instância analisados. No pior caso, a disponibilidade

média da instância g2.2xlarge alcançou 87,6%, para um valor médio de lance igual a 0,345 USD/hora (Tabela 7.6), equivalente a 53,08% do preço *on demand*. No melhor caso, a disponibilidade média da instância t1.micro alcançou 91,4%, para um valor médio de lance igual a 0,006 USD/hora (Tabela 7.9), equivalente a 30% do preço *on demand*.

Assim, percebeu-se então a oportunidade e a necessidade de formulação de um mecanismo voltado ao balanceamento de custo e disponibilidade de instâncias *spot* do serviço Amazon EC2. Para isso, foi aprofundado o conhecimento teórico sobre séries temporais, cálculo de médias móveis com janelas deslizantes, teoria de jogos e função de utilidade, conforme conteúdo apresentado no Capítulo 4. Em seguida, foi definido o mecanismo de estimativas de custo e disponibilidade baseadas em utilidade, conforme proposta e experimentos detalhados no Capítulo 8 e publicados em [49]. Para a instância g2.2xlarge, a taxa média de disponibilidade foi de 94,14%, com um valor médio de lance de 0,3435 USD/hora, o que equivale a 52,85% do preço *on demand*. Para a instância t1.micro, os resultados mostram uma taxa de disponibilidade média de 97,8%, com um valor de lance médio de 0,0064 USD/hora, o que equivale a 32,71% do preço *on demand*. Desse modo, o mecanismo proposto conseguiu evoluir as análises estatísticas realizadas anteriormente, proporcionando o balanceamento de custo e disponibilidade desejado.

Até aqui, as análises estatísticas e os mecanismo de utilidade se voltaram aos dados de histórico de preços de 2016, quando o modelo de precificação *spot* ainda se baseava em um modelo de mercado puro (vide mudança na precificação *spot*, discutida na Seção 6.1.19). Nesse período, as variações de preços eram muito intensas e não era raro encontrar valores de preço *spot* acima dos preços *on demand*. Ainda assim, os trabalhos desenvolvidos conseguiram prover estimativas de custos e disponibilidade bastante adequados.

Em seguida, foi proposto um mecanismo que combina a análise de utilidade, para sugestão de lances a curto prazo, com redes neurais LSTM, para análise de tendências de preço a longo prazo. Para isso, houve a necessidade de aprofundamento teórico sobre redes neurais artificiais, conforme conteúdo apresentado no Capítulo 5. Para isso, foram realizados experimentos com dados de histórico de preços de 2020, ou seja, após a mudança do modelo *spot* para mercado suavizado. A proposta do mecanismo combinado, os resultados dos experimentos e a realização do estudo de caso foram detalhados no Capítulo 9. O conteúdo desse capítulo foi submetido a um periódico de prestígio [50], sendo que uma primeira revisão já foi realizada.

Os experimentos com o mecanismo combinado mostraram que, para o tipo de instância r4.2xlarge, 90% das sugestões de lance *spot* garantiram pelo menos 5,73 horas de disponibilidade no segundo trimestre de 2020, com um valor de aproximadamente 38% do preço *on demand*. Os experimentos também foram capazes de prever tendências de preços *spot* para vários tipos de instâncias com erros muito baixos. Para o tipo de ins-

tância r5.2xlarge, o mecanismo previu um preço médio de 0,19 USD/hora, que equivale a cerca de 37% do preço *on demand*, com $MSE < 10^{-6}$ para a previsão em um período de 7 dias. O mecanismo combinado foi usado no estudo de caso para a comparação de milhares de sequências de DNA do SARS-CoV-2. Os resultados mostraram que a nossa abordagem foi capaz de fornecer boas opções de tipos de instância *spot*, com baixo custo e alta disponibilidade.

As análises detalhadas nas contribuições desta Tese podem ser aplicadas em serviços IaaS de outros provedores de computação em nuvem, como no caso do serviço *Elastic Compute Service* (ECS) do provedor Alibaba Cloud [130]. Neste provedor, instâncias preemptíveis são oferecidas como uma alternativa às instâncias *on demand*, para a redução de custos em determinados cenários. A precificação das instâncias se baseia num modelo de preço variável, em que o usuário estabelece lances para a contratação das instâncias, de forma muito semelhante ao modelo mercado real da precificação *spot* do serviço Amazon EC2, que vigorou até novembro de 2017 (Seção 6.1.19). As contribuições, de forma mais geral, também podem ser aplicadas em mercados que se baseiem em mecanismos de precificação estática ou dinâmica, para oferta de bens e serviços permanentes ou transientes, como por exemplo para negociação de ações no mercado financeiro.

10.2 Trabalhos Futuros

A presente Tese avançou o estado-da-arte ao fazer a análise do modelo de precificação *spot* com diversas técnicas (janelas deslizantes, função de utilidade, redes LSTMs), a curto e a longo prazo, visando lances de baixo valor e proporcionando alta disponibilidade dos recursos contratados. No entanto, existem ainda diversas questões em aberto, que merecem investigação em trabalhos futuros.

No Capítulo 9, determinou-se 10 configurações de *layout* e hiperparâmetros de rede LSTM (Tabela 9.3), que são adequadas para a previsão de tendências de preços *spot*, considerando os tipos de instâncias analisados, no período de tempo que compreende o primeiro semestre de 2020. A escolha entre essas 10 configurações de redes LSTM foi deixada a cargo do usuário e foi feita de maneira empírica. Como trabalho futuro, propõe-se que as previsões de cada configuração de rede LSTM sejam analisadas em detalhe, para se tentar determinar se existe relação entre a qualidade da previsão da rede e fatores como o tipo da instância, o tipo de variação de preço (frequência e amplitude) no período, o preço médio da instância, dentre outros. Caso uma relação entre os fatores e a configuração da rede seja determinada, é sugerido o projeto de um módulo que, de posse dos valores dos fatores relevantes, escolha a configuração de rede LSTM que proporcione os melhores resultados para o cenário específico.

Conforme discutido no Capítulo 6, o modelo de precificação *spot* está em evolução. No final de 2017, esse modelo foi bastante alterado e, atualmente, é usado o modelo de mercado suavizado. Assim, é sugerido, como trabalho futuro, o uso de Desenho de Mecanismos (Seção 4.6) para explorar alternativas em busca de uma evolução futura do modelo de precificação *spot*. O Desenho de Mecanismos é uma técnica muito promissora, que permite que diversos tipos de mercados, jogos e funções utilidade sejam explorados, visando a definição de alternativas ao modelo *spot* atual, que aumentem os benefícios para o provedor, para o cliente ou para ambos.

O mecanismo apresentado no Capítulo 8 leva em consideração a utilidade do ponto de vista do usuário de computação em nuvem. Caso fosse possível obter, além do histórico de precificação das instâncias *spot* da Amazon EC2, o histórico da demanda e utilização das instâncias, ou seja, o histórico de requisições por tipo de instância e zona de disponibilidade, no formato uma série temporal, seria possível modelar a função de utilidade do provedor. Assim, seria possível a modelagem de um jogo com informação incompleta (Seção 4.4), composto por dois jogadores: usuário e provedor. Tal modelagem possibilitaria ainda a busca pelo equilíbrio em jogos estáticos, além da simulação de execução de leilões 4.5.

O provedor AWS, bem como a maioria dos provedores de computação em nuvem pública, possui um único *pool* de instâncias, que podem ser utilizados ora no modelo *on demand* ora no modelo *spot*. Como trabalho futuro, propõe-se investigar como o padrão de uso das instâncias *on demand* afeta a disponibilidade das instâncias *spot*, na tentativa de compor uma modelagem unificada que leve em consideração ambos os modelos de precificação.

Finalmente, como evoluções pontuais e imediatas dos mecanismos apresentados, propõe-se a extensão do mecanismo de análise de utilidade (Capítulo 4), para a utilização de janelas dinâmicas. Pode-se, por exemplo, aplicar redes LSTM para aprendizado e proposição do tamanho ideal da janela, de forma semelhante à abordagem aplicada em [45]. Por último, também é proposta a construção de um módulo ou serviço que automatize a execução dos passos do mecanismo combinado (Seção 9.4 do Capítulo 9), de forma que receba características de infraestrutura desejáveis (e.g. quantidade de CPU e memória) e retorne a análise de tendências utilizando redes LSTM e as sugestões de lances e disponibilidade com base no mecanismo de utilidade, assim como a sugestão da instância mais adequada a ser contratada.

Referências

- [1] Leite, Alessandro F.: *A user-centered and autonomic multi-cloud architecture for high performance computing applications*. Tese de Doutorado, Computer Aided Engineering, Universite Paris Sud - Paris XI, 2014. xiii, 9, 16
- [2] Zhang, Qi, Lu Cheng e Boutaba Raouf: *Cloud computing: state-of-the-art and research challenges*. Journal of Internet Services and Applications, 1(1):7–18, 2010. xiii, 14
- [3] Services, Amazon Web: *Amazon ec2 spot blocks for defined-duration workloads*, 2017. <http://aws.amazon.com/blogs/aws/new-ec2-spot-blocks-for-defined-duration-workloads/>, acesso em 2017-09-23. xiii, 21, 23, 24
- [4] Norstad, John: *An introduction to utility theory*, 2011. xiii, 33, 34, 35
- [5] Brockwell, Peter J. e Richard A. Davis: *Introduction to Time Series and Forecasting*. Springer, 2016, ISBN 9783319298542. xiii, 35, 36, 37, 38
- [6] Standards, National Institute of e Technology: *Nist/sematech e-handbook of statistical methods*, 2019. <https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc4.htm>, acesso em 2019-11-20. xiii, 38, 39
- [7] Garratt, Rodd: *Vickery-clark-groves mechanism*. <http://faculty.econ.ucsb.edu/~garratt/Econ177/>, Lecture Notes, 2016. xiii, 45
- [8] Minsky, Marvin e Seymour A. Papert: *Perceptrons. An Introduction to Computational Geometry*. The MIT Press, 1969, ISBN 9780262630221. xiii, 46, 47
- [9] Aggarwal, Charu C.: *Neural Networks and Deep Learning*. Springer, Cham, 2018, ISBN 9783319944630. xiii, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 62, 64
- [10] Jaokar, Ajit: *A simplified explanation for understanding the mathematics of deep learning*, 2018. <https://www.datasciencecentral.com/profiles/blogs/a-simplified-explanation-for-understanding-the-mathematics-of>, acesso em 2018-12-17.
- [11] Ng, Andrew: *Machine learning by stanford university*, 2020. <https://www.coursera.org/learn/machine-learning/home/info>, acesso em 2020-11-19. xiii, 55, 56

- [12] Chang, Fi John, Li Chiu Chang e Hau Lung Huang: *Real-time recurrent learning neural network for stream-flow forecasting*. *Hydrological Processes*, 16(13):2577–2588, 2002. xiii, 64
- [13] Güldal, Veysel e Hakan Tongal: *Comparison of recurrent neural network, adaptive neuro-fuzzy inference system and stochastic models in egirdir lake level forecasting*. *Water Resources Management*, 24(1):105–128, 2010. xiii, 64, 65
- [14] Schafer, Anton Maximilian e Hans Georg Zimmermann: *Recurrent neural networks are universal approximators*. Em *Artificial Neural Networks (ICANN)*, páginas 632–640. Springer Berlin Heidelberg, 2006, ISBN 9783540386278. xiii, 65, 66
- [15] Greff, Klaus, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink e Jürgen Schmidhuber: *Lstm: A search space odyssey*. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017. xiii, 57, 58, 67
- [16] Leite, Alessandro F., Vander Alves, Genaina N. Rodrigues, Claude Tadonki, Christine Eisenbeis e Alba C.M.A. Melo: *Automating resource selection and configuration in inter-clouds through a software product line method*. Em *IEEE 8th International Conference on Cloud Computing*, páginas 726–733, New York, NY, USA, 2015. IEEE. xiii, xiv, 87, 88, 89, 90, 91
- [17] Services, Amazon Web: *Amazon ec2 on-demand pricing*, 2017. <http://aws.amazon.com/ec2/pricing/on-demand/>, acesso em 2017-09-23. xiv, xvi, 20, 21, 87, 88, 91, 92, 93
- [18] Platform, Google Cloud: *Google compute engine pricing*, 2017. <http://cloud.google.com/compute/pricing>, acesso em 2017-09-15. xvi, 25, 26
- [19] Azure, Microsoft: *Microsoft azure linux virtual machines pricing*, 2017. <http://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/>, acesso em 2017-09-27. xvi, 1, 28
- [20] Foster, Ian, Yong Zhao, I. Raicu e S. Lu: *Cloud computing and grid computing 360-degree compared*. Em *Grid Computing Environments Workshop*, Austin, TX, USA, 2008. IEEE. 1, 7, 8
- [21] Mell, Peter e Timothy Grance: *The nist definition of cloud computing*. Relatório Técnico SP800-145, NIST National Institute of Standards and Technology, 2011. 1, 2, 7, 8, 10, 11, 12, 13
- [22] Services, Amazon Web: *Cloud computing with aws*, 2017. <http://aws.amazon.com/what-is-aws/>, acesso em 2017-09-23. 1
- [23] Platform, Google Cloud: *Google cloud platform*, 2017. <http://cloud.google.com/>, acesso em 2017-09-15. 1, 25
- [24] Services, Amazon Web: *Amazon ec2 spot product details*, 2017. <http://aws.amazon.com/ec2/spot/details/>, acesso em 2017-09-23. 1, 19, 22, 23, 24, 71, 119

- [25] Baughman, Matt, Christian Haas, Rich Wolski, Ian Foster e Kyle Chard: *Predicting amazon spot prices with lstm networks*. Em *Proceedings of the 9th Workshop on Scientific Cloud Computing*, ScienceCloud'18, páginas 1:1–1:7. ACM, 2018, ISBN 978-1-4503-5863-7. <http://doi.acm.org/10.1145/3217880.3217881>. 2, 24, 77, 84, 116, 123, 137
- [26] Lee, Gunho, Byung Gon Chun e H. Katz: *Heterogeneity-aware resource allocation and scheduling in the cloud*. Em *Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing (HotCloud'11)*, páginas 4–4, Berkeley, CA, USA, 2011. USENIX Association. <http://dl.acm.org/citation.cfm?id=2170444.2170448>. 2, 69, 84, 137
- [27] Zhang, Miranda, Rajiv Ranjan, Surya Nepal, Michael Menzel e Armin Haller: *A declarative recommender system for cloud infrastructure services selection*. Em *Economics of Grids, Clouds, Systems, and Services (GECON)*, páginas 102–113. Springer Berlin Heidelberg, 2012. 2, 70, 84, 137
- [28] Huang, Botong, Nicholas W.D. Jarrett, Shivnath Babu, Sayan Mukherjee e Jun Yang: *Bidding strategies for spot instances in cloud computing markets*. *Proceedings of the VLDB Endowment*, 9(3):156–167, 2015. 2, 73, 84, 137
- [29] Javadi, Bahman, Ruppia K. Thulasiram e Rajkumar Buyya: *Characterizing spot price dynamics in public cloud environments*. *Future Generation Computer Systems*, 29(4):988–999, 2013. 2, 71, 72, 84, 137
- [30] Agmon Ben-Yehuda, Orna, Muli Ben-Yehuda, Assaf Schuster e Dan Tsafir: *Deconstructing amazon ec2 spot instance pricing*. *ACM Trans. Econ. Comput.*, 1(3):16:1–16:20, 2013, ISSN 2167-8375. 2, 70, 84, 96, 137
- [31] Karunakaran, Sowmya e Rangaraja P. Sundarraj: *Bidding strategies for spot instances in cloud computing markets*. *IEEE Internet Computing*, 19(3):32–40, 2015. 2, 73, 84, 137
- [32] Singh, Vivek Kumar e Kaushik Dutta: *Dynamic price prediction for amazon spot instances*. Em *48th Hawaii International Conference on System Sciences (HICSS)*, Kauai, HI, USA, 2015. IEEE. 2, 73, 84, 137
- [33] Lucas-Simarro, Jose Luis, Rafael Moreno-Vozmediano, Ruben S. Montero e Ignacio M. Llorente: *Cost optimization of virtual infrastructures in dynamic multi-cloud scenarios*. *Concurrency and Computation: Practice and Experience*, 27(9):2260–2277, 2015, ISSN 1532-0634. 2, 74, 84, 137
- [34] Ouyang, Xue, David Irwin e Prashant Shenoy: *Spotlight: An information service for the cloud*. Em *IEEE 36th International Conference on Distributed Computing Systems (ICDCS 2016)*, Nara, Japan, 2016. IEEE. 2, 74, 84, 137
- [35] Wolski, Rich, John Brevik, Ryan Chard e Kyle Chard: *Probabilistic guarantees of execution duration for amazon spot instances*. Em *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*,

- SC '17. Association for Computing Machinery, 2017, ISBN 9781450351140. 2, 76, 84, 120, 137
- [36] Al-Theiabat, Hana, Mahmoud Al-Ayyoub, Mohammad Alsmirat e Monther Aldwair: *A deep learning approach for amazon ec2 spot price prediction*. Em *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*, páginas 1–5, 2018. 2, 77, 84, 116, 137
- [37] Sharma, Prateek, Stephen Lee, Tian Guo, David Irwin e Prashant Shenoy: *Managing risk in a derivative iaas cloud*. *IEEE Transactions on Parallel and Distributed Systems*, 29(8):1750–1765, 2018. 2, 77, 84, 137
- [38] Shastri, Supreeth e David Irwin: *Cloud index tracking: Enabling predictable costs in cloud spot markets*. Em *Proceedings of the ACM Symposium on Cloud Computing (SoCC '18)*, página 451–463. Association for Computing Machinery, 2018, ISBN 9781450360111. 2, 78, 84, 137
- [39] Wang, Cheng, Qianlin Liang e Bhuvan Uргаonkar: *An empirical analysis of amazon ec2 spot instance features affecting cost-effective resource procurement*. Em *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering, ICPE '17*, páginas 63–74. ACM, 2017, ISBN 978-1-4503-4404-3. <http://doi.acm.org/10.1145/3030207.3030210>. 2, 78, 84, 137
- [40] Liu, Duan, Zhicheng Cai e Yifei Lu: *Spot price prediction based dynamic resource scheduling for web applications*. Em *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*, páginas 78–83, 2019. 2, 79, 84, 137
- [41] Mishra, Ashish Kumar, Abhishek Kesarwani e Dharmendra K. Yadav: *Short term price prediction for preemptible vm instances in cloud computing*. Em *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, páginas 1–9, 2019. 2, 80, 84, 137
- [42] Tanaka, Masahiro e Yohei Murakami: *Strategy-proof pricing for cloud service composition*. *IEEE Transactions on Cloud Computing*, 4(3):363–375, 2014. 2, 72, 84, 137
- [43] Xie, Ning, Xuejie Zhang e Jixian Zhang: *A truthful auction-based mechanism for virtual resource allocation and pricing in clouds*. Em *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, páginas 578–582, Chengdu, China, 2017. IEEE. 2
- [44] Agarwal, S., Ashish Mishra e D.K. Yadav: *Forecasting price of amazon spot instances using neural networks*. *International Journal of Applied Engineering Research*, 12:10276–10283, 2017, ISSN 09734562. 2, 76, 84, 137
- [45] Baig, Shuja ur Rehman, Waheed Iqbal, Josep Lluís Berral e David Carrera: *Adaptive sliding windows for improved estimation of data center resource utilization*. *Future Generation Computer Systems*, 104:212–224, 2020, ISSN 0167-739X. 2, 80, 84, 137, 142

- [46] Khandelwal, Veena, Anand Kishore Chaturvedi e Chandra Prakash Gupta: *Amazon ec2 spot price prediction using regression random forests*. IEEE Transactions on Cloud Computing, 8(1):59–72, 2020. 2, 81, 84, 137
- [47] Portella, Gustavo, Genaina N. Rodrigues e Alba C.M.A. Melo: *Análise de precificação de recursos utilizados em computação em nuvem*. Em *XVII Simpósio de Sistemas Computacionais de Alto Desempenho (WSCAD)*, Aracaju, SE, BRA, 2016. SBC. <http://www.lbd.dcc.ufmg.br/colecoes/wscad/2016/017.pdf>. 5, 139
- [48] Portella, Gustavo, Genaina N. Rodrigues, Eduardo Nakano e Alba C.M.A. Melo: *Statistical analysis of amazon ec2 cloud pricing models*. Concurrency and Computation: Practice and Experience, 31(18):e4451, 2019. <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4451>, e4451 cpe.4451. 5
- [49] Portella, Gustavo, Genaina N. Rodrigues, Eduardo Nakano e Alba C.M.A. Melo: *Utility-based strategy for balanced cost and availability at the cloud spot market*. Em *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, páginas 214–218, July 2019. 5, 104, 140
- [50] Portella, Gustavo, Genaina N. Rodrigues, Eduardo Nakano, Azzedine Boukerche e Alba C.M.A. Melo: *A statistical and neural network combined approach for the cloud spot market*. Em *Submitted to a prestigious journal*, 2021. 5, 116, 136, 137, 140
- [51] Vaquero, Luis M., Luis Roderó-Merino, Juan Caceres e Maik Lindner: *A break in the clouds: towards a cloud definition*. ACM SIGCOMM Computer Communication Review, 39(1):50–55, 2009. 7, 8
- [52] Buyya, Rajkumar, Chee Shin Yeo, Srikumar Venugopal, James Broberg e Ivona Brandic: *Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility*. Future Generation Computer Systems, 25(6):599–616, 2009. 7, 8
- [53] Foster, Ian e Carl Kesselman (editores): *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, ISBN 1-55860-475-8. 8, 9
- [54] Bandara, H. M. N. Dilum e Anura P. Jayasumana: *Collaborative applications over peer-to-peer systems—challenges and solutions*. Peer-to-Peer Networking and Applications, 6(3):257–276, 2013, ISSN 1936-6450. 8
- [55] Weiser, Mark: *Some computer science issues in ubiquitous computing*. Communications of the ACM, 36(7):75–84, 1993. 9
- [56] Dean, Jeffrey e Sanjay Ghemawat: *Mapreduce: simplified data processing on large clusters*. Communications of the ACM - 50th anniversary issue: 1958 - 2008, 51(1):107–113, 2008. 9
- [57] Foundation, Apache Software: *Apache hadoop*, 2017. <http://hadoop.apache.org/>, acesso em 2017-09-30. 9

- [58] Portella, Gustavo J. e Alba C. M. A. Melo: *Escalonamento de Tarefas Visando Balanceamento de Carga em Ambientes de Computação em Grade*. Em *XXXI Seminário Integrado de Software e Hardware - SEMISH*, Salvador, Bahia, Brazil, 2004. SBC. 9
- [59] Toolkit, Globus: *Globus toolkit 6.0 developer's guide*, 2017. <http://toolkit.globus.org/toolkit/docs/6.0/appendices/developer/>, acesso em 2017-10-23. 9
- [60] Androutsellis-Theotokis, Stephanos e Diomidis Spinellis: *A survey of peer-to-peer content distribution technologies*. *ACM Computing Surveys (CSUR)*, 36(4):335–371, 2004, ISSN 0360-0300. 9
- [61] Narahari, Y, C Raju, K Ravikumar e S Shah: *Dynamic pricing models for electronic business*. *Sadhana Academy Proceedings In Engineering Sciences*, 30:231–256, April 2005. 10, 19
- [62] Toosi, Adel Nadjaran, Rodrigo N. Calheiros e Rajkumar Buyya: *Interconnected cloud computing environments: Challenges, taxonomy, and survey*. *ACM Computing Surveys (CSUR)*, 47(1):1–47, 2014. 10
- [63] Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica e Matei Zaharia: *Above the clouds: A berkeley view of cloud computing*. Relatório Técnico UCB/EECS-2009-28, EECS Department, University of California, Berkeley, USA, 2009. 11
- [64] Barroso, Luiz André, Jimmy Clidaras e Urs Hölzle: *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition*. Morgan & Claypool Publishers, 2013, ISBN 9781627050104. 11
- [65] Duan, Yucong, Guohua Fu, Nianjun Zhou, Xiaobing Sun, Nanjangud Narendra e Bo Hu: *Everything as a service (xaas) on the cloud: Origins, current and future trends*. Em *2015 IEEE 8th International Conference on Cloud Computing*, páginas 621–628, 2015. 12
- [66] Fowler, Martin: *Serverless architectures*, 2018. <https://martinfowler.com/articles/serverless.html>, acesso em 2018-05-22. 12
- [67] Lee, Young e Bing Lian: *Cloud bursting scheduler for cost efficiency*. Em *IEEE 10th International Conference on Cloud Computing*, Honolulu, CA, USA, 2017. IEEE. 13
- [68] Sotomayor, B., R. Montero, S. Llorente e I. Foster: *Virtual infrastructure management in private and hybrid clouds*. *IEEE Internet Computing*, 13(5):14–22, 2009. 15
- [69] Varian, Hal R.: *Intermediate Microeconomics: A Modern Approach*. W. W. Norton & Company, 2014, ISBN 978-0393935332. 17

- [70] Plummer, Daryl: *Cloud elasticity could make you go broke - gartner blog network*, 2009. http://blogs.gartner.com/daryl_plummer/2009/03/11/cloud-elasticity-could-make-you-go-broke/, acesso em 2017-09-23. 18
- [71] Armbrust, Michael, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica e Matei Zaharia: *A view of cloud computing*. *Communications of the ACM*, 53(4):50–58, 2010. 18
- [72] Wang, Qiushi, Ming Tan, Xueyan Tang e Wentong Cai: *Minimizing cost in iaas clouds via scheduled instance reservation*. Em *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, GA, USA, 2017. IEEE. 19
- [73] Platform, Google Cloud: *Google preemptible vm instances*, 2017. <http://cloud.google.com/compute/docs/instances/preemptible>, acesso em 2017-09-15. 19, 27
- [74] Shastri, Supreeth, Amr Rizk e David Irwin: *Transient guarantees: Maximizing the value of idle cloud capacity*. Em *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, UT, USA, 2016. IEEE. 19, 20
- [75] Services, Amazon Web: *Amazon ec2 pricing*, 2017. <http://aws.amazon.com/ec2/pricing/>, acesso em 2017-09-23. 20, 22, 24, 108
- [76] Pahl, Claus, Antonio Brogi, Jacopo Soldani e Pooyan Jamshidi: *Cloud container technologies: A state-of-the-art review*. *IEEE Transactions on Cloud Computing*, 7(3):677–692, 2019. 20
- [77] Baldini, Ioana, Paul Castro, Kerry Chang, Perry Cheng, Stephen Fink, Vatche Ishakian, Nick Mitchell, Vinod Muthusamy, Rodric Rabbah, Aleksander Slominski e Philippe Suter: *Serverless Computing: Current Trends and Open Problems*, páginas 1–20. Springer Singapore, Singapore, 2017, ISBN 978-981-10-5026-8. 20
- [78] Chaisiri, Sivadon, Rakpong Kaewpuang, Bu Sung Lee e Dusit Niyato: *Cost minimization for provisioning virtual servers in amazon elastic compute cloud*. Em *2011 IEEE 19th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, Singapore, Singapore, 2013. IEEE. 20
- [79] Services, Amazon Web: *Amazon ebs pricing*, 2017. <http://aws.amazon.com/ebs/pricing/>, acesso em 2017-09-23. 21
- [80] Services, Amazon Web: *Amazon ec2 reserved instances pricing*, 2017. <http://aws.amazon.com/ec2/pricing/reserved-instances/pricing/>, acesso em 2017-09-23. 21, 22
- [81] Azure, Microsoft: *Microsoft azure reserved virtual machine instances*, 2017. <http://azure.microsoft.com/en-us/pricing/reserved-vm-instances/>, acesso em 2017-09-27. 29

- [82] Clark, Barry: *Political Economy: A Comparative Approach, 3rd Edition*. Praeger, 2016, ISBN 9781440843266. 33
- [83] Bernoulli, Daniel: *Exposition of a new theory on the measurement of risk*. *Econometrica*, 22(1):23–36, 1954, ISSN 00129682, 14680262. <http://www.jstor.org/stable/1909829>. 33
- [84] Merton, Robert C.: *Continuous-Time Finance*. Wiley-Blackwell, 1990, ISBN 0631185089. 33
- [85] Clements, Michael P. e David F. Hendry: *Forecasting Non-Stationary Economic Time Series*. The MIT Press, 2001, ISBN 9780262531894. 37
- [86] Jain, Raj: *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1991, ISBN 9780471503361. 39, 40, 87, 88
- [87] Gibbons, Robert S.: *Game theory for applied economists*. Princeton Univ. Press, Princeton, NJ, 1992, ISBN 0691043086. 40, 41, 42, 43
- [88] Myerson, Roger B.: *Game theory: Analysis of conflict*. Harvard University Press, 1997, ISBN 9780674341166. 40
- [89] Bugarin, Mauricio S. e Marilda Antonia de O. Sotomayor: *Jogos na forma estratégica com informação incompleta*. <http://www.bugarinmauricio.com/>, Notas de aula da disciplina de Teoria de Jogos. Programa de Doutorado em Economia da Universidade de Brasília., 2017. 40, 41, 42
- [90] Myerson, Roger B.: *Optimal auction design*. *Mathematics of Operations Research*, 6(1):58–73, 1981, ISSN 0364-765X. 43
- [91] Jehle, Geoffrey A. e Philip J. Reny: *Advanced Microeconomic Theory*. Prentice Hall, 2011, ISBN 9780273731917. 43, 44, 45
- [92] Vickrey, William: *Counterspeculation, auctions, and competitive sealed tenders*. *The Journal of Finance*, 16(1):8–37, 1961, ISSN 1540-6261. 44
- [93] Hurwicz, Leonid e Stanley Reiter: *Designing Economic Mechanisms*. Cambridge University Press, 2006, ISBN 9780511754258. 45
- [94] Easley, David e Jon Kleinberg: *Networks, Crowds, and Markets*. Cambridge University Press, 2011, ISBN 9780521195331. 45
- [95] Nisan, Noam e Amir Ronen: *Algorithmic mechanism design*". *Games and Economic Behavior*, 35(1):166–196, 2001, ISSN 0899-8256. 45
- [96] Groves, Theodore: *Incentives in teams*. *Econometrica*, 41(4):617–631, 1973, ISSN 00129682, 14680262. 45
- [97] Rosenblatt, Frank: *The perceptron: a probabilistic model for information storage and organization in the brain*. *Psychological review*, 65 (6):386–408, 1958. 46

- [98] Hecht-Nielsen, Robert: *Theory of the backpropagation neural network*, páginas 65–93. Academic Press, 1992, ISBN 978-0-12-741252-8. 47
- [99] Khan, Salman, Hossein Rahmani, Syed Afaq Ali Shah, Mohammed Bennamoun, Gerard Medioni e Sven Dickinson: *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool, 2018, ISBN 9781681730226. 47
- [100] He, Kaiming, Xiangyu Zhang, Shaoqing Ren e Jian Sun: *Deep residual learning for image recognition*. Em *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 770–778, 2016. 48, 57
- [101] Koza, John R., Forrest H. Bennett, David Andre e Martin A. Keane: *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*, páginas 151–170. Springer Netherlands, 1996, ISBN 978-94-009-0279-4. 48
- [102] Graves, Alex, Marcus Liwicki, Santiago Fernandez, Roman Bertolami, Horst Bunke e Jürgen Schmidhuber: *A novel connectionist system for unconstrained handwriting recognition*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009. 48, 63
- [103] Rumelhart, David E., Geoffrey E. Hinton e Ronald J. Williams: *Learning representations by back-propagating errors*. *Nature*, página 696–699, 1986. 49, 63
- [104] Goodfellow, Ian, Yoshua Bengio e Aaron Courville: *Deep Learning*. The MIT Press, 2016, ISBN 0262035618. 52, 53, 61, 62
- [105] Hardt, Moritz, Benjamin Recht e Yoram Singer: *Train faster, generalize better: Stability of stochastic gradient descent*. Em *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, volume 48, página 1225–1234. JMLR.org, 2016. <http://proceedings.mlr.press/v48/hardt16.pdf>. 54
- [106] Reimers, Nils e Iryna Gurevych: *Optimal hyperparameters for deep lstm-networks for sequence labeling tasks*. arXiv e-prints, 2017. 57
- [107] Bergstra, James e Yoshua Bengio: *Random search for hyper-parameter optimization*. *The Journal of Machine Learning Research*, 13:281–305, 2012. 57
- [108] Polyak, Boris T.: *Some methods of speeding up the convergence of iteration methods*. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964, ISSN 0041-5553. 59, 68
- [109] Nesterov, Yurii: *A method for solving the convex programming problem with convergence rate $o(1/k^2)$* . *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983. 60
- [110] Duchi, John, Elad Hazan e Yoram Singer: *Adaptive subgradient methods for online learning and stochastic optimization*. *J. Mach. Learn. Res.*, 12(null):2121–2159, 2011, ISSN 1532-4435. 60, 61

- [111] Ziemer, Rodger E. e William H. Tranter: *Principles of Communications: Systems, Modulation, and Noise*. Wiley, 2002, ISBN 9780471392538. 61
- [112] Hinton, Geoffrey: *Neural networks for machine learning, lecture 6a overview of mini-batch gradient descent*, 2014. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, acesso em 2020-11-20. 61
- [113] Kingma, Diederik e Jimmy Ba: *Adam: A method for stochastic optimization*. International Conference on Learning Representations (ICLR), 2015. <http://arxiv.org/abs/1412.6980>. 62
- [114] Dozat, Timothy: *Incorporating nesterov momentum into adam*. Relatório Técnico 054, Stanford University, 2016. http://cs229.stanford.edu/proj2015/054_report.pdf. 62, 63
- [115] Zimmermann, Hans, Ralph Grothmann, Anton Schafer e Christoph Tietz: *Identification and Forecasting of Large Dynamical Systems by Dynamical Consistent Neural Networks*, páginas 203–242. MIT Press, 2006, ISBN 9780262083485. 63
- [116] Han, Jun e Claudio Moraga: *The influence of the sigmoid function parameters on the speed of backpropagation learning*, páginas 195–201. Springer Berlin Heidelberg, 1995, ISBN 9783540492887. 66
- [117] Hochreiter, Sepp e Jürgen Schmidhuber: *Long short-term memory*. *Neural Computation*, 9(8):1735–1780, 1997. 66
- [118] Graves, Alex e Jürgen Schmidhuber: *Framewise phoneme classification with bidirectional lstm networks*. Em *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, páginas 2047–2052, 2005. 66
- [119] Gers, Felix A., Jürgen A. Schmidhuber e Fred A. Cummins: *Learning to forget: Continual prediction with lstm*. *Neural Comput.*, 12(10):2451–2471, 2000. 67
- [120] Sutskever, Ilya, James Martens, George Dahl e Geoffrey Hinton: *On the importance of initialization and momentum in deep learning*. Em *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13*, página III–1139–III–1147. JMLR.org, 2013. 68
- [121] Li, Zheng, He Zhang, Liam OBrien, Shu Jiang, You Zhou, Maria Kihl e Rajiv Ranjan: *Spot pricing in the cloud ecosystem: A comparative investigation*. *Journal of Systems and Software*, 114(Supplement C):1–19, 2016, ISSN 0164-1212. 75, 81, 82
- [122] Sharpe, William F.: *A simplified model for portfolio analysis*. *Management Science*, 9(2):277–293, 1963. 78
- [123] Wolski, Rich, Gareth George, Chandra Krintz e John Brevik: *Analyzing aws spot instance pricing*. Relatório Técnico 2018-02, University of California, Santa Barbara, 2018. 79, 82

- [124] Baughman, Matt, Simon Caton, Christian Haas, Ryan Chard, Rich Wolski, Ian Foster e Kyle Chard: *Deconstructing the 2017 changes to aws spot market pricing*. Em *Proceedings of the 10th Workshop on Scientific Cloud Computing*, ScienceCloud '19, página 19–26, New York, NY, USA, 2019. Association for Computing Machinery, ISBN 9781450367585. 79, 82
- [125] Lu, Ye, Xin Lei Chen, Bo Jie Wang, Teng Yue Wang, Pei Zhang e Yong Li: *Recycling price prediction of renewable resources*. Em *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, UbiComp/ISWC '19 Adjunct, página 571–576. Association for Computing Machinery, 2019, ISBN 9781450368698. <https://doi.org/10.1145/3341162.3349326>. 80, 82
- [126] Portella, Gustavo, Genaina N. Rodrigues, Eduardo Nakano e Alba C.M.A. Melo: *Statistical analysis of amazon ec2 cloud pricing models*. *Concurrency and Computation: Practice and Experience*, 30(7):1–15, 2017. 139
- [127] Bianchi, Filippo Maria, Enrico Maiorino, Michael C. Kampffmeyer, Antonello Rizzi e Robert Jenssen: *Recurrent Neural Networks for Short-Term Load Forecasting*. Springer International Publishing, 2017, ISBN 9783319703381. 116
- [128] Sandes, Edans, Guillermo Miranda, Xavier Martorell, Eduard Ayguade, George Teodoro e Alba C.M.A. Melo: *Masa: A multiplatform architecture for sequence aligners with block pruning*. *ACM Trans. Parallel Comput.*, 2(4), 2016. 131
- [129] Fiorentini, Simona, Serena Messali, Alberto Zani, Francesca Caccuri, Marta Giovanetti, Massimo Ciccozzi e Arnaldo Caruso: *First detection of sars-cov-2 spike protein n501 mutation in italy in august, 2020*. *The Lancet Infectious Diseases*, 2021, ISSN 1473-3099. 135
- [130] Cloud, Alibaba: *Elastic compute service*, 2021. <https://www.alibabacloud.com/help/product/25365.htm>, acesso em 2021-04-25. 141

Anexo I

Artigo Publicado no WSCAD 2016
(primeira página)

Análise de Precificação de Recursos Utilizados em Computação em Nuvem

Gustavo J. Portella¹, Genaina N. Rodrigues¹, Alba C. M. de Melo¹

¹Departamento de Ciência da Computação (CIC)
Universidade de Brasília (UnB) – Brasília, DF – Brazil

gustavo.portella@aluno.unb.br, {genaina,albamm}@cic.unb.br

Abstract. *Cloud providers usually offer a wide variety of resources to their users. In this scenario, selecting an inappropriate resource can lead to financial losses and/or long response times. Therefore, the use of an effective strategy for selecting cloud computing resources can bring important benefits to users. In this paper, we present experiments in order to determine the influence of the characteristics of processor and memory in the composition of the cost of different types of resources available on the Amazon EC2 and Google GCE providers. Such analysis can be incorporated in strategies which aim to decrease the financial cost and to attain a good performance for the cloud applications.*

Resumo. *Os provedores de nuvem geralmente oferecem uma grande variedade de recursos a seus usuários. Nesse cenário, a seleção de um recurso inapropriado pode levar a perdas financeiras e/ou tempos de resposta longos. Sendo assim, a utilização de uma estratégia eficiente de seleção de recursos em nuvem pode trazer importantes benefícios para os seus usuários. Neste artigo, são apresentados experimentos com o objetivo de se determinar a influência de características do processador e da memória na composição do custo de diferentes recursos disponíveis nos provedores Amazon EC2 e Google GCE. Tal análise pode ser incorporada em estratégias que visam ao mesmo tempo a diminuição do custo financeiro e a manutenção de bom desempenho para as aplicações em nuvem.*

1. Introdução


O surgimento da computação em nuvem proporcionou uma verdadeira mudança quanto ao uso de infraestrutura de recursos computacionais. Nesse novo modelo, um usuário tem a possibilidade de escolher os recursos computacionais de acordo com os requisitos de execução de sua aplicação e pagar pelo tempo de uso [Foster et al. 2008]. Essa possibilidade faz com que a escolha correta dos recursos mereça uma atenção especial, ainda mais pelo fato de que os custos decorrentes da utilização são aferidos não somente pelo tipo, mas também em função do tempo de utilização dos recursos.

Nesse contexto, os provedores de computação em nuvem oferecem serviços com modelos de precificação de recursos flexíveis e de acordo com as necessidades dos usuários. Um exemplo disso é o caso da *Amazon*, que oferece três diferentes modelos de precificação para comercialização de seu serviço *EC2 – Elastic Compute Cloud* [Amazon 2016], a depender do tipo de instância ou máquina virtual utilizada. A empresa *Google* também oferece um modelo flexível por meio do serviço *GCE –*

Anexo II

Artigo Publicado no CCPE 2017
(primeira página)

Statistical analysis of Amazon EC2 cloud pricing models

Gustavo Portella¹ | Genaina N. Rodrigues¹ | Eduardo Nakano² | Alba C.M.A. Melo¹ 

¹Department of Computer Science, Predio CIC/EST, Universidad de Brasília, Campus Darcy Ribeiro, Brasilia-DF 70910-900, Brazil

²Department of Statistics, Predio CIC/EST, Universidad de Brasília, Campus Darcy Ribeiro, Brasilia-DF 70910-900, Brazil

Correspondence

Alba C.M.A. Melo, Department of Computer Science, Predio CIC/EST, Universidad de Brasília, Campus Darcy Ribeiro, Brasilia-DF 70910-900, Brazil.
Email: alves@unb.br

Summary

In this paper, we conduct statistical analyses for two Amazon cloud pricing models: on demand and spot. On demand cloud instances are charged a fixed price and can only be terminated by the user, with very high availability. On the other hand, spot instances are charged a dynamic price determined by a market-driven model and can be revoked by the provider when the spot price becomes higher than the user-defined price, having possibly low availability. Our analysis for on-demand instances resulted in multiple linear regression equations that represent the influence of characteristics of the processor and RAM memory in the composition of the price of different types of instances available on the Amazon EC2 provider. In order to analyze the Amazon spot pricing, we used time-smoothed moving averages by 12-hour periods, aiming to provide a price-availability trade-off to the user. Our experiments with spot price histories from September to November 2016 show that the user's bid can be set at 30% of the on-demand price, with an availability above of 90%, depending on instance type.

KEYWORDS

cloud computing, cloud pricing policies

1 | INTRODUCTION

Cloud computing is a large scale distributed computing paradigm for provisioning software, platform, or computational infrastructure, enabling dynamic and elastic resource sharing. Cloud computing has many benefits such as (a) decreasing the financial cost of deploying applications; (b) increasing the capacity of private infrastructures in an elastic way; and (c) reducing power consumption due to resource sharing.

The access to cloud functionalities relies basically on three levels of abstraction: software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS). While IaaS offers access to the cloud infrastructure, PaaS builds upon the IaaS, providing tools that help cloud applications' design.¹ Finally, SaaS provides the applications themselves and manages in a transparent way the whole computing environment. In this paper, we deal with IaaS clouds.

Over the last decade, cloud computing has been seen by both the enterprises and academia as a new computing paradigm for achieving low cost solutions at reduced time. Indeed, public cloud providers such as Amazon Elastic Compute Cloud (EC2) have built an extraordinary infrastructure composed of several huge data centers spread over multiple continents, giving the users the illusion of unlimited resources. In order to have access to these infrastructures, users are offered a large amount of resource types, each one with different characteristics and distinct prices.

Amazon EC2 rents virtual machines (called instances) according to an approach that encompasses three pricing models: reserved, on demand, and spot.² In the reserved pricing scheme, users assume a long-term commitment (ie, one or three years), paying a discounted rate. On demand resources are allocated in a hourly basis and are charged a fixed price. The spot scheme emerged in 2009 because many EC2 cloud resources remained unsold, resulting in a considerable idle capacity. According to Amazon, the spot instances are regulated by a market-driven model, where cloud users bid on spare resources, which are often much cheaper than on-demand resources, and will be able to use them as long as the bid exceeds the current spot price. When this condition does not hold, a two-minute warning is sent to the user and, after this period of time, the instance is terminated. There is a different spot market for each cloud data center or small set of data centers. Recent analyses of the spot market showed that it appears to be no law behind the spot price variation, indicating that it does follow a market-driven model.²

In the IaaS cloud pricing models, there is clearly a trade-off between the price paid by the customers and the guarantees given by the cloud providers. Customers who choose spot instances certainly wish to pay a low price. However, these users also have to cope with the situation

Anexo III

**Artigo Publicado no IEEE Cloud
2019 (primeira página)**

Utility-based Strategy for Balanced Cost and Availability at the Cloud Spot Market

Gustavo J. Portella

*Dep. of Computer Science**University of Brasilia (UnB)**Brasilia, Brazil**gustavo.portella@aluno.unb.br*

Eduardo Nakano

*Dep. of Statistics**University of Brasilia (UnB)**Brasilia, Brazil**nakano@unb.br*

Genaina N. Rodrigues

*Dep. of Computer Science**University of Brasilia (UnB)**Brasilia, Brazil**genaina@cic.unb.br*

Alba C. M. A. Melo

*Dep. of Computer Science**University of Brasilia (UnB)**Brasilia, Brazil**alves@unb.br*

Abstract—In the Amazon EC2 cloud provider, the price of spot instances is much lower than on demand instances, however, at the cost of availability issues of the former. Recently, several strategies were proposed to analyze the Amazon EC2 spot pricing model employing techniques such as statistical and probabilistic modeling and neural networks. To the best of our knowledge, there is no work in the literature that can accurately capture the trade-off between price and availability in favor of user decision-making. In this work, we propose and evaluate a utility-based strategy that balances spot instance cost and availability to Amazon EC2 users. Our experiments show that the average availability reaches up to 98% for spot instances, using data gathered from Amazon (September to November 2016), and a bid value below 28% of the on demand price for the *m4.10xlarge* general purpose instance type.

Keywords—cloud computing; spot model; utility estimation

I. INTRODUCTION

Cloud computing enables convenient access to a shared set of configurable computing resources (e.g. computers, network, storage, and services) that can be quickly provisioned and made available with reduced management and administration effort [1]. In this context, a user has the possibility to choose the computational resources according to his/her application requirements and pay for the usage time [2]. Therefore, the correct choice of resources deserves special attention from the user.

Amazon Elastic Compute Cloud (Amazon EC2) provides *Infrastructure as a Service (IaaS)* resources or instances in three pricing models: *on demand*, *reserved* and *spot* [9]. In the first two models, the price is static per hour, and there is no price variation due to the amount of resources demanded by users. The *spot* pricing model is characterized as dynamic, since an auction-based market mechanism is used which makes the instance prices vary over time. The amount of instances offered in this pricing model is characterized by the current number of idle instances, i.e., instances which were not marketed in the *on demand* or *reserved* models. The prices are established according to the demand of users, who bid for instances. If the bid is above the current *spot* price and there are idle instances, the instance is provided to the user, otherwise, if the *spot* price exceeds the bid, the instance is revoked. Therefore, although the price paid for *spot* instances is in general much lower than the price of *on*

demand/reserved instances, *spot* instances may be revoked at any time, thus creating an availability issue.

Providing models/strategies that analyze *spot* instance histories is an intensive area of research [6] [7] [8] [11] [12]. However, to the best of our knowledge, there is no work in the literature that can accurately capture the trade-off between price and availability in favor of the decision-making for the Amazon EC2 user.

The main contribution of this work is a utility-based model that balances *spot* instance cost and availability to Amazon EC2 users. We claim that a useful *spot* model for user bids should combine cost and availability since both components are remarkably important to the user. Moreover, a strategy that provides a balance between these components is highly desirable.

To evaluate our model, we performed a pricing history analysis of four Amazon EC2 *spot* instance types. From September to November 2016, for the *m4.10xlarge* instance type, our model was able to obtain an average availability rate of 98.19% with an average bid equals to 28.82% with respect to the *on demand* price. The *c3.8xlarge* instance type presented a more complex behavior in the same period of time, with numerous price spikes. Even in this case, our model estimated an average availability rate of 95.89% for an average bid equivalent to 50.33% of the *on demand* price.

The remainder of this paper is organized as follows. In Section II, the Amazon *spot* pricing scheme is described. Related work is discussed in Section III. Our utility-based proposal is presented in Section IV. Section V presents and discusses experimental results. Finally, Section VI concludes the paper and suggests some important future work directions.

II. AMAZON SPOT PRICING SCHEME

In the IaaS model [1], the cloud provider offers a set of virtualized infrastructure resources. Users are responsible for selecting these resources (e.g. virtual machines) and managing them, while the provider is responsible for (a) selecting a physical resource and instantiating the virtual machine (VM) on it; (b) monitoring the physical resource; and (c) charge for use. This work focuses on the user strategy for instance selection.

Anexo IV

Artigo Submetido a Periódico
(primeira página)

A Statistical and Neural Network Combined Approach for the Cloud Spot Market

Gustavo J. Portella, Eduardo Nakano, Genaina N. Rodrigues,
Azzedine Boukerche, *Fellow, IEEE*, Alba C. M. A. Melo, *Senior Member, IEEE*

Abstract—The price of virtual machine instances in the Amazon EC2 spot model is often much lower than in the on-demand counterpart. However, this price reduction comes with a decrease in the availability guarantees. Several mechanisms have been proposed to analyze the spot model in the last years, employing different strategies. To the best of our knowledge, there is no work that accurately captures the trade-off between spot price and availability, for short term analysis, and does long term analysis for spot price tendencies, in favor of user decision making. In this work, we propose (a) a utility-based strategy, that balances cost and availability of spot instances and is targeted to short-term analysis, and (b) a recurrent neural network mechanism for long term spot price prediction. Our experiments show that, for r4.2xlarge, 90% of spot bid suggestions ensured at least 5.73 hours of availability in the second quarter of 2020, with a bid price of approximately 38% of the on-demand price. The neural network experiments were able to predict tendencies for spot prices for several instance types with very low error. The mechanism predicted an average value of 0.19 USD/hour for the r5.2xlarge instance type (Mean Squared Error $< 10^{-6}$) for a 7-day period of time, which is about 37% of the on-demand price. Finally, we used our combined mechanism to run an application that compares thousands of SARS-CoV-2 DNA sequences and show that our approach is able to provide good choices of instances, with low bids and very good availability.

Index Terms—cloud computing, utility estimation, neural network.

1 INTRODUCTION

CLOUD computing enables access to a wide range of distributed resources and the user pays for usage time of provisioned resources under a pricing model defined by the provider. Therefore, an appropriate user choice for resource type and pricing model deserves special attention, aiming to obtain high computational performance at low cost.

Amazon Elastic Compute Cloud (Amazon EC2) provides Infrastructure as a Service (IaaS) resources, or instances, in three pricing models: *on-demand*, *reserved* and *spot* [1]. In the first two models, the price is static, charged per hour, and there is no price variation. The spot pricing model is characterized as dynamic since it uses a market mechanism, which makes prices vary over time according to instance supply and user demand. In addition, spot instances may be revoked by Amazon before the end of the execution, leading to availability issues.

Several types of cloud applications may benefit from the spot model. The obvious candidates are applications composed of short-lived tasks, with no deadlines. In this case, if the instance is revoked, the user is notified and he/she can choose another spot instance to run the remaining tasks. Even long-running applications with defined deadlines may run in the spot model, assuming that checkpoints are taken.

In this case, the user may choose to start executing in the spot market and restart from the checkpoint, if the instance is revoked. If the deadline is approaching, the user may switch to the on demand model, which has availability guarantees.

Providing strategies to analyze the spot pricing model is an intensive area of research. Most works analyze spot prices, aiming to predict appropriate values to be set by the user when requesting spot instances. This is important, since appropriate spot bids (maximum price) may lead to significant financial savings, from the user side. However, setting the maximum price too low has a direct impact on the instance availability and may lead to several interruptions of the execution, augmenting considerably the execution time. For instance, an increase of 50% in the execution time of an application that takes 6 hours would correspond to a 9-hour execution. For some users, this 3-hour increase may be unacceptable. On the other hand, if the user wants to augment the availability and sets the maximum spot price too close to the on demand price, less interruptions may occur but the user may pay more than he/she wanted. Since spot price variations follow the market model [2], its analysis is very complex and needs sophisticated strategies.

To deal with this problem, several works in the literature use probability techniques and statistics [3], [4], [5], [6], [7], [8], and many other works use machine learning [9], [10], [11], [12]. To the best of our knowledge, there is no work in the literature that accurately captures the tradeoff between price and availability for short term analysis, and does long term analysis for spot price tendencies, both in favor of user decision making.

In this work, we propose an efficient utility-based scheme that balances instance availability and cost coupled

- Gustavo J. Portella, Genaina N. Rodrigues and Alba C. M. A. Melo are with the Department of Computer Science, University of Brasilia (UnB), Brasilia, Brazil. E-mail: gustavo.portella@aluno.unb.br, genaina@cic.unb.br and alves@unb.br.
- Eduardo Nakano is with the Department of Statistics, University of Brasilia (UnB), Brasilia, Brazil. E-mail: nakano@unb.br.
- Azzedine Boukerche is with the Faculty of Engineering, University of Ottawa, Ottawa, Canada. E-mail: boukerch@eecs.uottawa.ca.

Manuscript received October 6, 2020; revised February 4, 2021.