Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Redução de Custo Computacional em Classificações Baseadas em Transformadas Aprendidas

Emerson Lopes Machado

Tese apresentada como requisito parcial
para conclusão do Doutorado em Informática

Orientador
Prof. Ricardo Pezzuol Jacobi

Coorientador
Prof. Cristiano Jacques Miosso

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Doutorado em Informática

Coordenador: Prof. Alba Cristina Magalhães Alves de Melo

Banca examinadora composta por:

Prof. Ricardo Pezzuol Jacobi (Orientador) — CIC/UnB
Prof. Adson Ferreira da Rocha — FGA/UnB
Prof. Ricardo von Borries — EE/UTEP
Prof. Mylene Christine Queiroz de Farias — CIC/UnB
Prof. George Luiz Medeiros Teodoro — CIC/UnB

CIP — Catalogação Internacional na Publicação

Lopes Machado, Emerson.

Redução de Custo Computacional em Classificações Baseadas em Transformadas Aprendidas / Emerson Lopes Machado. Brasília : UnB, 2015.
69 p. : il. ; 29,5 cm.

Tese (Doutorado) — Universidade de Brasília, Brasília, 2015.

1. Classificação de imagens, 2. Aprendizado de dicionários,
3. Soft-thresholding, 4. Aprendizado de transformada, 5. Redução de custo computacional, 6. FPGA, 7. Powerize

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Redução de Custo Computacional em Classificações Baseadas em Transformadas Aprendidas

Emerson Lopes Machado

Tese apresentada como requisito parcial
para conclusão do Doutorado em Informática

Prof. Ricardo Pezzuol Jacobi (Orientador)
CIC/UnB

Prof. Adson Ferreira da Rocha
FGA/UnB

Prof. Ricardo von Borries
EE/UTEP

Prof. Mylene Christine Queiroz de Farias
CIC/UnB

Prof. Alba Cristina Magalhães Alves de Melo
Coordenador do Doutorado em Informática

Brasília, 8 de julho de 2015

Dedicatória

Dedico à minha querida esposa, por acreditar em mim e ficar ao meu lado nos momentos mais difíceis durante este trabalho de doutorado. Não há palavras que consigam demonstrar a gratidão que tenho por você. Dedico também ao meu filho João, esse lindo bebê que tem sede de explorar o mundo :)

Agradecimentos

Em especial, agradeço aos meus pais e à minha esposa, por acreditarem em mim e ficarem ao meu lado nos momentos mais difíceis desse doutorado. Agradeço também aos meus orientadores por me guiarem durante os caminhos sombrios deste doutorado. Sou grato também ao Professor George von Borries e a todas as pessoas que tive contato durante esses 5 anos de doutorado, em especial à Dr. Suzana Saruê, que me motivou a fazer um doutorado desde os meus tempos de mestrado.

I thank the Department of Electrical & Computer Engineering of the University of Texas at El Paso for allowing me access to the NSF-supported cluster (NSF CNS-0709438) used in all the simulations here described and also Mr. Nito Gumataotao for his assistance with it. I also thank Mr. Alhussein Fawzi for providing me the source code of LAST and for all his invaluable help on its details.

Sou grato também à CAPES pelas bolsa de doutorado pleno e de doutorado sanduíche, que me permitiram ter a tranquilidade necessária para eu focar nos meus estudos.

Enfim, sou grato de coração a tudo e a todos!

Resumo

Apresento nesta tese a análise teórica e as avaliações empíricas do conjunto de técnicas que proponho para a redução do custo computacional da classificação em tempo de teste de classificadores que são baseados em transformada e limiar suave aprendidos a partir dos dados de treinamento. Modificando o procedimento de otimização numérica utilizado na aprendizagem da transformada e do vetor de classificação, assim como aplicando um processamento em seus respectivos elementos após a aprendizagem, as técnicas que proponho permitem reduzir a quantidade de bits necessária para se realizar a classificação e trocar cada multiplicação em ponto flutuante por um simples deslocamento de bit em inteiro. Como estudo de caso, utilizei o algoritmo de classificação *Learning Algorithm for Soft-Thresholding* (LAST) e os mesmos conjuntos de dados utilizados no artigo que o apresenta. Os resultados do estudo de caso confirmam a possibilidade de se utilizar somente somas e deslocamentos em inteiro para a classificação em tempo de teste com uma perda de acurácia limitada. Essas operações de baixo custo computacional são importantes em implementações feitas em FPGA por permitir aumentar a velocidade de classificação e ao mesmo tempo diminuir o consumo de energia e o custo de fabricação. Além disso, as técnicas que apresento reduziram em quase 50% a quantidade de bits necessária para a extração de características na maioria dos experimentos que realizei.

Palavras-chave: Classificação de imagens, Aprendizado de dicionários, Soft-thresholding, Aprendizado de transformada, Redução de custo computacional, FPGA, Powerize

Abstract

We present a theoretical analysis and empirical evaluations of a novel set of techniques for computational cost reduction of classifiers that are based on learned transform and soft-threshold. By modifying optimization procedures for dictionary and classifier training, as well as the resulting dictionary elements, our techniques allow to reduce the bit precision and to replace each floating-point multiplication by a single integer bit shift. We also show how the optimization algorithms in some dictionary training methods can be modified to penalize higher-energy dictionaries. We applied our techniques with the classifier Learning Algorithm for Soft-Thresholding, testing on the datasets used in its original paper. Our results indicate it is feasible to use solely sums and bit shifts of integers to classify at test time with a limited reduction of the classification accuracy. These low power operations are a valuable trade off in FPGA implementations as they increase the classification throughput while decrease both energy consumption and manufacturing cost. Moreover, our techniques reduced almost 50% of the bit precision in almost all datasets we tested.

Keywords: Image classification, Dictionary learning, Soft-thresholding, Transform learning, Reduce computational cost, FPGA, Powerize

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Nesta pesquisa, proponho um conjunto de técnicas que permitem a redução do custo computacional em tempo de teste de classificadores baseados em tranformada aprendida [27] e soft-threshold [9]. Essas técnicas exploram as propriedades que derivei desses classificadores ao longo de minha pesquisa. Para a classificação em tempo de teste (do inglês *testing time classification*), as técnicas que apresento neste documento são divididas em quatro:

(i) Usar os sinais no mesmo formato em que são amostrados pelos conversores analógicos digitais (ADC, do inglês *analog-to-digital converter*), ou seja, representados por inteiros, em vez de reescalonados, como normalmente é feito em processos de normalização. Em *hardware*, especialmente em arranjo de portas programáveis por campo (FPGA, do inglês *field programmable gate array*), operações em inteiros são menos custosas do que operações em ponto flutuante. Eu provo que essa troca não afeta a acurácia de classificação.

(ii) Aproximar os valores da matriz de transformação e do vetor de classificação para suas respectivas potências de 2 mais próximas. Essa aproximação permite que cada multiplicação na classificação em tempo de teste seja substituída por um único deslocamento de bit.

(iii) Quantizar os sinais a serem classificados em tempo de teste de forma a diminuir suas respectivas faixas dinâmicas e com isso diminuir a quantidade de bits necessária para se realizar a classificação.

(iv) Diminuir a faixa dinâmica da matriz de transformação utilizando duas abordagens: primeiramente, adicionando uma penalização à sua norma; em seguida, zerando as entradas dessa matriz que possuem valores abaixo de um limiar, cujo valor é aprendido a partir dos dados de treinamento.

As três últimas técnicas introduzem uma redução na acurácia de classificação, sendo que, nas últimas duas, essa redução pode ser controlada.

Como estudo de caso para essas técnicas que proponho, usei um algoritmo de classificação recente chamado *Learning Algorithm for Soft-Thresholding classifier* (LAST), que aprende ao mesmo tempo a matriz que esparsifica os sinais e o vetor hiperplano de classificação. Descreve esse algoritmo na Seção 2.5. Nos testes que realizei, utilizei bases de dados frequentemente utilizadas na literatura para comparar desempenho de classificação, as quais descrevo na Seção 4.1. Os resultados que obtive indicam que as técnicas que proponho reduzem o custo computacional com perda limitada de acurácia da classificação. Além disso, a acurácia da classificação em uma das bases de dados aumentou.

De acordo com o levantamento bibliográfico que realizei, as técnicas que apresento neste documento são inéditas na literatura. Essas técnicas dispensam a necessidade de microprocessadores especializados em processamento digital de sinal (DSP, do inglês *digital signal processor*) para as operações de multiplicação matriz-vetor em arquiteturas baseadas em FPGAs. Isso reduz o custo total de fabricação do sistema além de reduzir o consumo de energia. Portanto, essas técnicas são apropriadas para o uso desde em classificadores embarcados, nos quais o consumo de energia é crítico e o poder computacional é restrito, até classificações de grande porte, como servidores de busca por imagem do Google, onde o uso de energia chega a ultrapassar o de pequenas cidades [28].

Neste trabalho, todas as simulações que realizei foram para a classificação de imagens utilizando o algoritmo LAST. Não obstante, as técnicas que proponho são suficientemente gerais para serem aplicadas em outros problemas e com a utilização de outros algoritmos de classificação que usam multiplicações do sinal pela matriz esparsificante para a extração de características, tais como o algoritmo máquinas de aprendizado extremo (ELMs, do inglês *extreme learning machines*) [16] e o algoritmo redes neurais profundas (DNNs, do inglês *deep neural networks*) [29].

1.1 Contextualização

Classificação de imagens em tempo real é um desafio importante e difícil em processamento de imagens e visão computacional, ainda mais quando existe a necessidade de se embarcar. Uma plataforma dedicada pode trazer as vantagens de reduzir o custo de fabricação, o tamanho do produto e o consumo de energia, além de poder acelerar a classificação de forma significativa. Algumas aplicações embarcadas recentes incluem o uso de unidades de processamento de gráficos (GPUs, do inglês *graphics processing unit*), como em [2], no qual os autores portaram o algoritmo de detecção automática de alvo e de classificação (ATDCA, do inglês *automatic target detection and classification algorithm*) para a classificação de imagens hiperespectrais. Outras aplicações incluem o uso de DSPs, como em [26], no qual um sistema de classificação de imagens foi embarcado para a detecção automática de pequenos defeitos na confecção têxtil. Além disso, FPGAs têm sido também utilizados para embarcar algoritmos do estado da arte da classificação de imagens, como em [17], no qual os autores embarcaram uma rede neural convolucional (CNN, do inglês *convolutional neural network*) [19] para detecção de objetos, resultando em um desempenho que permitiu classificar vídeo em tempo real. Entretanto, um ponto negativo dessas abordagens é a necessidade do uso de métodos especializados para a extração e/ou seleção de características, geralmente baseadas em métodos genéricos como transformada discreta de cossenos (DCT, do inglês *discrete cosine transform*), transformada discreta de Fourier (DFT, do inglês

discrete Fourier transform), banco de filtros, *Wavelets*, ou mesmo exigindo conhecimento de um especialista na fase de pré-processamento [12].

No problema de classificação, a extração de características é um passo importante, especialmente em problemas nos quais o conjunto de treinamento tem um alto espaço dimensional, o qual demanda um alto processamento e grande quantidade de memória [8]. Uma tendência recente na extração de características para a classificação de imagens consiste na representação esparsa da imagem em um dicionário redundante. Quando o dicionário é aprendido especificamente para o conjunto de treinamento, a classificação de características esparsas podem conseguir resultados comparáveis aos do estado da arte [22]. Entretanto, essa abordagem tem com ponto negativo a alta demanda computacional em tempo de teste, tornando-se impraticável para aplicações embarcadas, que geralmente dispõem de escasso poder computacional e restrições de consumo de energia.

Uma forma recente de lidar com essa limitação é aprender uma transformada esparsificante a partir dos próprios dados de treinamento [13, 27, 31]. Em tempo de teste, essa abordagem reduz a representação esparsa de sinais a uma simples operação de multiplicação de matriz por vetor seguida de uma operação de limiar suave (do inglês *soft-threshold*), na qual os valores abaixo de um limiar são anulados. Essas operações podem ser realizadas de forma eficiente em *hardware*, devido à sua natureza paralela. Entretanto, essas multiplicações envolvem operações em ponto flutuante, as quais têm um alto custo computacional. Esse custo é especialmente alto em FPGAs, pois necessitam de uma área muito maior quando programados para efetuar operações em ponto flutuante. Comparado com uma GPU, um FPGA necessita de uma densidade 12 vezes maior quando operações em ponto flutuante são utilizadas [5]. Uma solução comum para esse problema é o uso de operações em ponto fixo, gerando um compromisso entre o custo de fabricação e uma diminuição da precisão numérica causada por erro de quantização. Esse erro de quantização é geralmente contornado com o aumento da precisão do acumulador da multiplicação, com um custo maior do DSP e um aumento do consumo de energia [33].

A técnicas propostas nesta tese buscam resolver essas questões de desempenho computacional sem perda ou com perda limitada de acurácia de classificação. Em um trabalho relacionado [32], é apresentada uma proposta de processador capaz de classificar sinais de eletroencefalograma (EEG) para detecção de episódios epilépticos, utilizando técnicas de *compressive sensing* para classificação no domínio comprimido. A proposta deles reduz em 21 vezes o tamanho dos sinais de EEG e consequentemente, reduz em 21 vezes o número de operações de multiplicação seguida de acumulação (MACs, do inglês *multiply–accumulate operations*). Cabe lembrar que os autores não contabilizam o custo computacional da projeção aleatória necessária para amostrar os sinais de forma comprimida, pois ela é constituída por apenas somas e subtrações em ponto flutuante e, portanto, de baixo custo computacional [32].

Inspirado nessa afirmativa, proponho nesta tese que todas as operações MAC necessárias para a classificação em tempo de teste sejam substituídas por operações de baixo custo computacional, nas quais cada multiplicação em ponto flutuante é substituída por uma simples operação de deslocamento de bit em inteiro, que é uma das operações de menor custo em *hardware*.

1.2 Objetivos, Perguntas de Pesquisa e Contribuições

Este documento apresenta contribuições para a área de classificação de imagens e visão computacional, com foco em processamento em tempo real e sistemas embarcados. O objetivo principal é desenvolver técnicas que reduzam o custo computacional de classificadores, com foco nos que são baseados em multiplicação matriz-vetor para a extração de características. Primeiramente, desenvolvo um estudo das propriedades desses classificadores e, com base nesse estudo preliminar, proponho técnicas para tornar esses classificadores mais eficientes do ponto de vista computacional.

A primeira proposta é que a parte que demanda maior custo computacional, a extração de características, seja feita utilizando operações em inteiro em vez de operações em

ponto fixo/flutuante. Essa troca é benéfica e reduz o custo, visto que operações em ponto flutuante são mais onerosas do que operações em inteiro. Apresento a prova matemática de que essa troca não afeta a acurácia de classificação. Em seguida, proponho a aproximação das entradas de ambos dicionário e hiperplano de classificação para potências de 2. Para a aproximação para potências de 2, provo que a máxima distância entre qualquer valor escalar real x e sua potência de 2 mais próxima está limitada superiormente por $x/3$. Também proponho a quantização das entradas dos vetores sinais. Mostro que, em determinados casos, é possível reduzir a representação dos vetores sinais a 1-bit sem que haja queda substancial da acurácia de classificação.

Como forma de se reduzir a faixa dinâmica das entradas do dicionário, proponho uma modificação na função objetivo de algoritmos de aprendizagem de dicionários que são baseados otimização numérica. A ideia geral é favorecer vetores com baixa energia e, assim, obter dicionários com entradas dentro de uma faixa reduzida de valores, de forma que ocupem menos bits para sua representação e diminua a precisão de bits das operações de extração de características. Primeiramente, proponho uma abordagem particular que se aplica a qualquer algoritmo de aprendizagem baseado no método do gradiente (do inglês, gradient descent method). Em seguida, estendo para o caso mais geral do método de Newton, que é um dos métodos mais utilizados para otimização numérica [3, 24].

Por fim, avalio essas técnicas que proponho com o algoritmo LAST e as bases de dados que são utilizadas no artigo que o apresenta [13]. De forma a avaliar essas técnicas, as análises e os experimentos que foram conduzidos neste trabalho foram guiadas pelas seguintes perguntas de pesquisa:

i. Como se comporta a acurácia de um dicionário e hiperplano de classificação formados por valores inteiros? Quais as condições suficientes e necessárias para que essa seja comparável à obtida com outros classificadores da literatura que possuem dicionário e hiperplano com entradas em ponto flutuante?

ii. Quais subconjuntos de valores discretos que os elementos do dicionário e hiperplano podem assumir de forma a reduzir o custo computacional? Qual é a redução que essas

escolhas impõem no custo computacional? Qual o impacto que essas escolhas fazem na acurácia do modelo gerado?

iii. Como a redução na precisão das entradas do dicionário e das entradas dos vetores de sinais que serão classificados em tempo de teste afeta a acurácia?

iv. Como o aprendizado do dicionário pode ser modificado de forma que as suas entradas fiquem dentro de uma faixa reduzida de bits? Qual impacto que essa modificação impõe na acurácia do modelo gerado?

Capítulo 2

Fundamentação Teórica

Este capítulo aborda de forma sucinta a teoria que embasa a representação esparsa de sinais com dicionários super completos em tarefas de classificação. Na Seção 2.1, descrevo em linhas gerais o processo de aprendizado de classificadores de sinais, discorrendo sobre a importância da representação correta de sinais para tarefas de classificação e conceituando sinais do tipo imagem, que é o tipo de sinal que utilizei como estudo de caso. Em seguida, apresento na Seção 2.2 o estado da arte de algoritmos de classificação de sinais que aprendem a representação esparsa a partir do próprio conjunto de treinamento. Em seguida, na Seção 2.3, abordo com mais detalhes a representação esparsa de sinais, estendendo-a na Seção 2.4 com a representação esparsa em dicionários de síntese e de análise. Por último, apresento com mais detalhes o algoritmo LAST, que utilizei como estudo de caso para a validação das técnicas que proponho para a redução de custo computacional.

2.1 Conceito de Sinais e de Classificação de Sinais

No domínio de aprendizagem de máquina e estatística, a classificação é um problema de categorizar novas observações utilizando um modelo gerado a priori para um conjunto de observações com categorias já conhecidas. Em aprendizagem de máquina, usualmente são utilizados os termos instância para observação e classe para categoria.

Cada instância contém um conjunto de propriedades chamadas de características, ou mesmo vetor de características, que podem assumir valores discretos ou contínuos. As características discretas podem assumir valores categóricos, tais como o tipo sanguíneo que pode ser 'A', 'B', 'AB' ou 'O', e ordinais, ou seja, que possuem relação de ordem, tais como 'pequeno', 'médio' ou 'grande'. Já as características contínuas podem assumir quaisquer valores do domínio real. Um conjunto de instâncias é usualmente chamado de conjunto de dados (do inglês, *data set*, ou mesmo *dataset*).

Um algoritmo, ou a sua implementação que executa uma classificação, é chamado de classificador. O desempenho de classificadores em diversas tarefas geralmente depende das características dos dados a serem classificados. Isso implica a inexistência de um único classificador que é o melhor em todos os problemas. Essa prerrogativa é conhecida como teorema 'No Free Lunch' (tradução livre, 'nada é de graça') [8]. Portanto, encontrar um classificador apropriado a um conjunto de dados consiste em um processo contínuo de execução de diversos testes empíricos.

Neste documento, considero para estudo de caso a classificação de imagens. Antes de introduzir o conceito formal de classificação de sinais, apresento a seguir o conceito formal de sinais e imagens, que é o tipo de sinal que utilizei nesta pesquisa.

2.1.1 Conceito de Sinais e de Imagens

Na área de processamento de sinais, um sinal é qualquer quantidade que representa uma informação que varia de acordo com o espaço e/ou tempo, ou seja, é uma função sobre o estado de algum fenômeno físico, tal como luz, som e energia elétrica. Sinais são adquiridos com sensores próprios para cada tipo de sinal, também chamados de transdutores, que transformam os sinais em sua forma original para sinais elétricos. Essa transformação é feita com sensores específicos para o tipo de sinal de interesse, como um microfone para a captura de áudio, um sensor fotoelétrico para a captura de imagens, e um galvanômetro para a captura de sinais elétricos.

Nesta pesquisa considerei o caso de imagens e, portanto, focarei nesse tipo de sinal. Imagens obtidas com tecnologia óptica são formadas a partir de dispositivos que podem conter milhões de transdutores fotossensíveis distribuídos em uma matriz de forma quadrada ou retangular. Os sinais elétricos resultantes variam no tempo de forma contínua de acordo com a incidência de luz sobre o sensor em determinado instante.

Para formar uma imagem estática, os sinais contínuos capturados por cada transdutor fotossensível são amostrados em intervalos específicos utilizando ADCs. Além de discretizar no tempo, um ADC também quantiza a amplitude contínua do sinal elétrico. Com isso, a imagem que incidiu no sensor em um determinado instante é transformada em uma matriz com cada elemento, variando entre 0 e $2^n - 1$, onde n é a profundidade em bits do quantizador existente no ADC. Nesse contexto, cada elemento adquirido é denominado *píxel* (do inglês *picture element*). Para a aquisição de cores, é comum o uso de três ou mais transdutores fotossensíveis por píxel, cada um contendo um filtro específico para cada cor.

2.1.2 Classificação de Sinais

Formalmente, a classificação de sinais é definida da seguinte forma. Seja um conjunto de sinais $(\mathbf{X}, \mathbf{y})$, onde $\mathbf{X}$ contém n sinais $\mathbf{x} = [x_1, x_2, \ldots, x_m] \in \mathbb{R}^m$, associados a uma das k classes possíveis $y \in \{1, 2, \ldots, k\}$. Seja $y_i = f(\mathbf{x}_i)$, com $i = 1, \ldots, n$, a função desconhecida que define a relação entre os sinais e as suas respectivas classes. O problema de classificação consiste em encontrar uma função $\hat{f}(\mathbf{x})$ que melhor aproxime $y_i = f(\mathbf{x}_i)$, com $i = 1, \ldots, n$ de forma que o erro médio quadrático (MSE, do inglês *mean squared error*)

$$\arg\min_{\hat{f}(\mathbf{x})} \sum (y_i - \hat{f}(\mathbf{x}_i))^2 \ \forall i = 1, \ldots, n, \tag{2.1}$$

seja mínimo tanto para o conjunto de treinamento $\mathbf{X}$ quanto para demais sinais do mesmo tipo que não foram apresentados ao algoritmo de treinamento. Como existe um compromisso entre minimizar o erro para os sinais do conjunto de treinamento e minimizar o erro para os demais sinais [8], é importante que o modelo aprendido não seja

superajustado aos sinais de treinamento para que ele possa generalizar para instâncias desconhecidas. A função $\hat{f}(\mathbf{x})$ que melhor aproxima $f(\mathbf{x})$ e que não seja sobreajustada a $\mathbf{X}$ pode ser encontrada com métodos de otimização matemática tais como métodos de otimização numérica [3, 24].

Um problema recorrente em aprendizado de máquina é a maldição da dimensionalidade, onde os sinais de treinamento possuem alta dimensionalidade. Para se ter uma ideia de como a dimensionalidade afeta o aprendizado de um classificador, considere a tarefa de se aprender um classificador para um conjunto de dados binários $(\mathbf{X}, \mathbf{y})$ contendo 100 dimensões. Caso o conjunto de dados contenha 1.000.000 de instâncias, que é uma quantidade razoável de dados, essa tarefa consiste em encontrar uma função $\hat{f}(\mathbf{x})$ que consiga prever a classe de $2^{100} - 10^6 \approx 10^{30} - 10^6 = 10^{24}$ instâncias desconhecidas de forma correta. Para se ter uma ideia de como isso é um problema em classificação de imagens, uma imagem monocromática do conjunto de dados MNIST, apresentado na Seção 4.2, possui resolução de 12×12 píxeis possui e, portanto, 144 dimensões caso cada píxel seja utilizado como característica. Como cada píxel 8 bits de profundidade, e o conjunto de treinamento possui 60.000 instâncias, aprender um classificador para esse conjunto de dados consiste em encontrar uma função $\hat{f}(\mathbf{x})$ que consiga prever a classe de $256^{144} - 60.000 \approx 10^{346} - 10^5 = 10^{341}$ instâncias desconhecidas.

Duas formas de se mitigar o problema da dimensionalidade são obter mais dados para o treinamento e encontrar uma representação com menor dimensionalidade para os sinais. Obter mais dados para o treinamento nem sempre é possível e portanto a redução de dimensionalidade é geralmente a mais utilizada. Existem diversas técnicas de redução de dimensionalidade tais como análise de componentes principais (PCA, do inglês *principal component analysis*) [10] e projeções aleatórias em subdimensões [6].

Um ponto a se considerar sobre sinais naturais é que eles não estão distribuídos de forma uniforme no espaço em que residem [8]. Sinais com alta dimensionalidade tendem a se aglomerar em sub-regiões do espaço multidimensional e essa característica pode ser explorada de forma a facilitar a busca do classificador mais adequado para esses sinais.

Um exemplo prático é classificação de dígitos manuscritos. Mesmo que os dados sejam imagens contendo diversos píxeis, como o conjunto de dados MNIST que possui $m = 144$ dimensões, cada dígito ocupa apenas uma sub-região do espaço m-dimensional. Dessa forma, a representação dos sinais pode ser aprendida utilizando essa informação a priori, de forma que o classificador possa aprender padrões relevantes para a tarefa de classificação.

2.2 Estado da Arte em Classificação de Sinais

Na classificação de sinais, a forma em que um sinal é representado pode realçar ou suprimir diferentes fatores que explicam a variação dos sinais [1]. Assim, a representação de sinais, especialmente em problemas em que os sinais possuem alta dimensionalidade é um dos passos mais importantes do aprendizado de classificadores, chamado de extração de características. Esse passo é tão importante que existem *workshops* regulares em conferências, tais como NIPS (*Neural Information Processing Systems*) e ICML (*International Conference on Machine Learning*), e uma conferência dedicada a esse tópico, a ICLR (*International Conference on Learning Representations*) [1].

Essa alta relevância da fase de extração de características na construção de classificadores é devido à incapacidade dos algoritmos clássicos de aprendizagem de máquina em adquirir informações discriminativas a partir do dados [1]. Portanto, a comunidade de aprendizado de máquina tem concentrado esforços na construção de algoritmos de classificação de sinais que buscam compensar essa deficiência, incorporando no seu aprendizado a busca pela representação mais adequada para os sinais de treinamento. As características a priori tipicamente aprendidas em classificadores recentes são suavidade (do inglês *smoothness*), coerência espacial e temporal, variedade (do inglês *manifold*) e esparsidade. Dentre elas, a esparsidade tem se mostrado eficiente não somente em tarefas de classificação [22] como também em outras aplicações de processamento de sinais, tais como redução de ruído em imagens [11] e compressão de imagens [35].

Seguindo essa tendência de representar sinais de forma esparsa, a construção de

dicionários específicos ao domínio do problema para representação esparsa tem sido tema frequente na literatura [13, 14, 22]. Quando um dicionário é aprendido especificamente para o conjunto de treinamento, a acurácia da classificação de características esparsas é comparável à de algoritmos do estado da arte [22]. Esse tipo de dicionário é conhecido na literatura como dicionário de síntese, que abordo com mais detalhes na Seção 2.4.1.

Entretanto, o uso dicionários de síntese para representação esparsa possui alto custo computacional em tempo de teste, pois é necessária a busca da combinação dos elementos do dicionário que melhor aproxima o sinal de teste. Dessa forma, dicionários para representação esparsa baseados em transformadas, chamados de dicionários de análise, são preferidos para aplicações que requerem menor custo computacional em tempo de teste. De forma geral, dicionários de análise são compostos por transformada seguida da operação limiar (do inglês *threshold*), que é uma operação não-linear onde valores negativos são mapeados para zero. Esse tipo de dicionário possui a vantagem de ser uma simples operação de multiplicação matriz-vetor seguida de comparador lógico, que pode ser computada de forma paralela em hardware. Descrevo com mais detalhes esse tipo de dicionário na Seção 2.4.2.

2.3 Representação Esparsa de Sinais

Essa seção é baseada no Capítulo 3 do livro *Compressive Sensing* [30]. Sinais podem ser vistos como elementos de certos espaços matemáticos. Usualmente, o espaço de Hilbert é utilizado, pois ele possui propriedades importantes para o processamento de sinais, como produto interno e completo. Em outras palavras ele é um espaço vetorial abstrato, no qual comprimento, distância e ângulo podem ser medidos. Por completo, entende-se que, se uma sequência de vetores pertencentes a esse espaço converge para um limite, é garantido que esse limite esteja também dentro desse mesmo espaço.

Dentre as propriedades do espaço de Hilbert, a de maior interesse para o processamento de sinais é a do par de pontos mais próximo. Em linhas gerais, essa propriedade dita que

para todo elemento $\mathbf{x}$ no espaço de Hilbert existe apenas um elemento $\hat{\mathbf{x}}$ dentro de um conjunto não-vazio, convexo e completo que é o mais próximo de $\mathbf{x}$. Isso garante que, no espaço de Hilbert, a melhor aproximação de um sinal dentro de um conjunto não-vazio, convexo e completo é única.

Seja uma função $R : \mathcal{H} \mapsto S$ que mapeia um espaço de Hilbert $\mathcal{H}$ em um espaço de sequências infinitas. Um sinal $\mathbf{x} \in \mathcal{H}$ pode ser representado como um sequência $R(\mathbf{x})$ da forma

$$R(\mathbf{x}) = (s_1, s_2, s_3, \dots) \in S, \tag{2.2}$$

na qual $s_i = (\alpha_i, \gamma_i)$, γ_i é uma forma de onda e α_i é o coeficiente associado a essa forma de onda. Essas formas de ondas constituem um conjunto de funções $\mathbf{D} = (\gamma)$, chamado de dicionário, que contém os átomos usados para representar um sinal no espaço $\mathcal{H}$. Quando R é inversível, o sinal $\mathbf{x}$ pode ser recuperado de forma precisa a partir da sua representação α utilizando

$$\mathbf{x} = \sum \alpha\gamma, \tag{2.3}$$

e quando a representação não é exata, a reconstrução de $\mathbf{x}$ é feita com técnicas de aproximação, que são apresentadas na Seção 2.4.

Uma representação adequada requer o uso de pelo menos um dicionário completo, onde a quantidade de átomos do dicionário é maior ou igual à dimensão do sinal. No caso em que o tamanho do dicionário é igual à dimensão do sinal e seus átomos geram o espaço $\mathcal{H}$, ele é chamado de base.

Um dicionário pode conter redundâncias e assim possuir mais átomos que a dimensão do sinal representado. Nesse caso, a representação do sinal não é mais única como no caso de uma representação em uma base. O conjunto formado por seus átomos é chamado de *Frame*, que é uma generalização do conceito de base para dicionários com elementos que geram o espaço sem a restrição de independência dos átomos. A vantagem de se ter um dicionário redundante vem da possibilidade de se representar um sinal com uma quantidade menor de coeficientes não nulos do que seria possível com um dicionário não-redundante.

Bases e *frames* permitem que um sinal seja representado de forma esparsa e a busca dessa representação pode ser descrita da seguinte forma. Seja $\mathbf{x} \in \mathbb{R}^m$ um vetor sinal e $\mathbf{D} \in \mathbb{R}^{m \times d}$ um dicionário onde existe uma representação esparsa de $\mathbf{x}$. A forma canônica do problema de busca de uma representação esparsa é dada por

$$\min_{\alpha} \|\alpha\|_0 \text{ sujeito a } \mathbf{x} = \mathbf{D}\alpha, \tag{2.4}$$

onde $\alpha \in \mathbb{R}^d$ é o vetor de coeficientes esparsos e $\|\cdot\|_0$ é uma medida de esparsidade que conta o número de coeficientes não nulos. Apesar desse problema ser NP-completo, existem soluções por aproximação que permitem reduzi-lo a um problema polinomial usando diferentes técnicas de otimização numérica. Esse tipo de dicionário é revisto na Seção 2.4.1.

Quando o dicionário $\mathbf{D}$ é inversível, a representação esparsa de $\mathbf{x}$ é $\alpha = \mathbf{D}^{-1}\mathbf{x}$. Essa propriedade é válida também para dicionários onde seus átomos formam uma base ortonormal, pois $\mathbf{D}^{-1} = \mathbf{D}^\top$ e assim a representação esparsa de $\mathbf{x}$ é simplesmente $\alpha = \mathbf{D}^\top\mathbf{x}$. Essa propriedade permite reduzir o problema de otimização (2.4) a uma simples multiplicação matriz-vetor, que possui menor custo computacional. Esse tipo de dicionário é revisto na Seção 2.4.2.

2.4 Tipos de Dicionários

A escolha do dicionário é de suma importância para uma representação eficiente do sinal de forma esparsa. É ela que ditará o nível de compressão do sinal e o custo computacional envolvida na operação.

Em geral, dicionários podem ser utilizados de três formas: escolhendo um dicionário dentre os pré-fabricados, tais como DCT e DFT, geralmente referenciados como transformadas; criando uma seleção ajustada de um dicionário, no qual uma base ou um *Frame* é criado controlando-se certos parâmetros, como em pacotes de Wavelets, onde as subdivisões tempo-frequência são ajustadas ao conjunto de sinais de interesse; ou aprendendo o

dicionário a partir dos próprios sinais de interesse, utilizando-se de técnicas de aprendizado de máquina.

A vantagem de se utilizar dicionários pré-construídos ou mesmo adaptados é que eles podem possuir uma análise teórica mais detalhada. Outra vantagem é que eles podem possuir algoritmos rápidos de transformação, como a transformada rápida de Fourier (FFT, do inglês *fast Fourier transform*) que reduz a complexidade computacional da DFT de $O(n^2)$ para $O(n \log n)$. Entretanto, existem tipos de sinais que não possuem representação esparsa nesses dicionários. Nesses casos, é interessante que o dicionário seja aprendido especificamente para os sinais de interesse.

2.4.1 Dicionários de Síntese

A representação esparsa de sinais em dicionários aprendidos tem-se mostrado eficiente em diversas aplicações, indicando que sinais naturais tendem a possuir representações esparsas. Apesar de alguns elementos de dicionários aprendidos serem semelhantes a componentes de Wavelets ou de filtros de Gabor, eles são ajustados aos sinais de exemplo, proporcionando melhores resultados na prática. Diferentemente da decomposição baseada na análise de componentes principais e suas variantes, esses modelos não impõem que o dicionário seja ortogonal e nem mesmo que seus vetores sejam linearmente independentes.

Seja um conjunto de sinais $\mathbf{X} = \{\mathbf{x}_i\}$, $1 \leq i \leq n$ gerada por um modelo fixo mas desconhecido, definido como

$$\mathcal{M}_{(\mathbf{D},k_0,\alpha,\epsilon)}, \tag{2.5}$$

onde $\mathbf{D}$ é o dicionário, k_0 é quantidade de coeficientes não nulos de α, que é o vetor de coeficientes da representação nesse modelo, e ϵ é o desvio do sinal ao modelo. Assumindo que ϵ é fornecido, o aprendizado do dicionário $\mathbf{D}$ é reduzido ao problema de otimização

$$\min_{\mathbf{D},\{\alpha_i\}_{i=1}^n} \sum_{i=1}^{n} \|\alpha_i\|_0 \text{ sujeito a } \|\mathbf{x}_i - \mathbf{D}\alpha_i\|_2 \leq \epsilon. \tag{2.6}$$

Esse problema opera de forma a encontrar o dicionário $\mathbf{D}$ e as representações $\{\alpha_i\}_{i=1}^n$ ao mesmo tempo. Ele tem como restrição que a representação pode ter desvio no máximo igual a ε e como penalização que a representação tem que ser esparsa. Encontrar uma solução para esse problema significa encontrar um modelo candidato a (2.5).

As regras de penalização e restrições podem ser invertidas em (2.6) se for escolhido restringir a esparsidade e obter o melhor ajuste. O problema fica da forma

$$\min_{\mathbf{D},\{\alpha_i\}_{i=1}^n} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{D}\alpha_i\|_2 \text{ sujeito a } \|\alpha_i\|_0 \leq k_0. \tag{2.7}$$

2.4.2 Dicionários de Análise

Outra forma de se representar um sinal de forma esparsa é com a utilização de uma transformada, onde o dicionário é uma transformação linear que mapeia um sinal a sua representação esparsa. Como exemplo, os sinais formados pela superposição de senoides têm representação densa no domínio do tempo e representação esparsa no domínio da frequência. Para esses tipos de sinais, a transformada de Fourier é a transformada que representa os sinais de forma esparsa. Portanto,

$$\mathbf{z} = \mathbf{D}^\top \mathbf{x} \tag{2.8}$$

é a representação esparsa de $\mathbf{x}$ obtida pela transformada $\mathbf{D}$.

De forma geral, a transformada $\mathbf{D}$, também chamada de dicionário de análise, pode ser aprendida de forma específica para o conjunto de sinais de interesse. Ela pode ser mais que completa (do inglês *overcomplete*) como em [31], pode ser quadrada e inversível como em [27], ou mesmo pode ser uma matriz sem restrições na quantidade e forma dos átomos que a constitui como o algoritmo LAST [13].

Quando um sinal é corrompido por um ruído aditivo Gaussiano (AWGN, do inglês *additive white gaussian noise*), o vetor de coeficientes resultado da transformação esparsa não é esparso. Uma forma comum de esparsificar esse vetor de coeficientes é zerar os

valores que são abaixo de um certo limiar, utilizando para isso um operador limiar. As duas técnicas mais simples de operação limiar são limiar duro (do inglês *hard threshold*) e limiar suave (do inglês *soft threshold*). O limiar duro simplesmente zera todos os valores abaixo do limiar duro, deixando um espaço vazio entre o valor zero e os demais valores. O limiar suave é uma forma de se lidar com esse espaço vazio, no qual além de zerar os valores abaixo do limiar, os demais valores são subtraídos do limiar, de forma a deixá-los mais próximos de zero [9].

Seja $\mathbf{z} = [z_i]_{i=1}^{n}$ a representação esparsa de um sinal corrompido por AWGN. Portanto,

$$z_i = s_i + e\,\epsilon_i \quad i = 1, ..., n \tag{2.9}$$

onde s_i são os coeficientes da representação esparsa do sinal não corrompido por ruído, $e > 0$ é o nível do ruído e ϵ_i é o ruído independente do sinal e identicamente distribuído com $\mathcal{N}(0, 1)$. Como os coeficientes s_i são esparsos, existe um limiar λ que permite separar a maior parte de s_i do ruído ϵ_i tal que o operador limiar

$$h_\lambda(\mathbf{z}) = \text{sgn}(\mathbf{z}) \max(0, |\mathbf{z}| - \lambda) \tag{2.10}$$

esparsifica $\mathbf{z}$, onde

$$\text{sgn}(x) = \begin{cases} 1 & \text{se } x > 0; \\ 0 & \text{se } x = 0; \\ -1 & \text{se } x < 0. \end{cases}$$

Para tarefas de classificação, a melhor forma de se estimar λ é calculá-lo diretamente a partir dos sinais de treinamento.

2.5 Learning Algorithm for Soft-Thresholding Classifier (LAST)

O algoritmo de classificação LAST [13] é baseado no aprendizado de uma transformada e de um limiar suave para a representação esparsa de sinais. Diferentemente do operador de limiar suave (2.10), LAST utiliza um operador que zera todos os valores negativos. Assim,

$$h_\alpha(\mathbf{z}) = \max(0, z - \alpha), \tag{2.11}$$

onde α é o limiar, também chamado de parâmetro de esparsidade. Quando $\alpha = 0$, esse operador não-linear pode ser visto como a função de ativação retificadora (do inglês, *rectifier activation function*) $g(x) = \max(0, x)$, que tem mostrado bons resultados em arquiteturas de aprendizado profundas (do inglês, *deep learning*) [15, 21, 23, 37].

A vantagem do LAST é a sua capacidade de aprender ao mesmo tempo o dicionário de esparsificação e os parâmetros do classificador. Para os sinais do conjunto de treinamento $(\mathbf{X}, \mathbf{y})$, onde $\mathbf{X} = [\mathbf{x}_i]_{i=1}^n$ é o conjunto de sinais com cada sinal $\mathbf{x}_i \in \mathbf{R}^m$ e $\mathbf{y} = [y_i]_{i=1}^n$ são as respectiva classes de $\mathbf{x}_i$ com cada classe $y_i \in \{-1, 1\}$, o dicionário esparsificante $\mathbf{D} \in \mathbb{R}^{m \times d}$, que contém d átomos, e o hiperplano classificador $\mathbf{w} \in \mathbb{R}^d$ são estimados usando a otimização supervisionada

$$\arg\min_{\mathbf{D},\mathbf{w}} \sum_{i=1}^{m} L(y_i \mathbf{w}^\top h_\alpha(\mathbf{D}^\top \mathbf{x_i})) + \frac{v}{2}\|\mathbf{w}\|_2^2, \tag{2.12}$$

onde L é a função de perda *hinge* $L(x) = \max(0, 1 - x)$ e v é o parâmetro de regularização que previne o classificador $\mathbf{w}$ de se sobreajustar aos dados do conjunto de treinamento.

Em tempo de teste, a saída de classificador para um sinal $\mathbf{x}$ normalizado (com norma ℓ_2 igual a 1) é

$$\mathit{classe} = \begin{cases} +1 & \text{se } \mathbf{w}^\top \max(0, \mathbf{D}^\top \mathbf{x} - \alpha) > 0, \\ -1 & \text{caso contrário,} \end{cases} \tag{2.13}$$

onde *classe* é a classe retornada pelo classificador. A Figura 2.1 mostra o procedimento de classificação em tempo de teste de um sinal de entrada $\mathbf{x}$ normalizado (com norma ℓ_2 igual a 1).

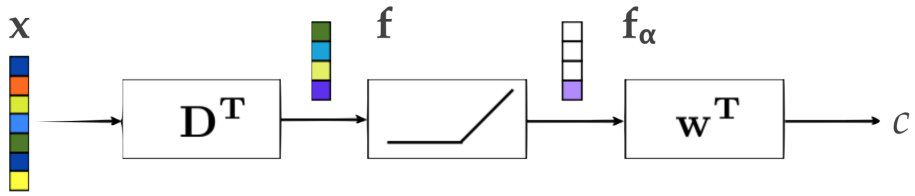


Figura 2.1: Classificação em tempo de teste do LAST. O sinal de teste $\mathbf{x}$ normalizado (com norma ℓ_2 igual a 1) é primeiramente transformado de forma linear por meio da matriz $\mathbf{D}$, resultando no vetor de características $\mathbf{f} = \mathbf{D}^\top \mathbf{x}$. Em seguida, o vetor de características $\mathbf{f}$ é transformado de forma não-linear por meio de limiarização suave, resultando no vetor de características esparsas $\mathbf{f}_\alpha = \max(0, \mathbf{f} - \alpha)$. Por fim, esse vetor de características esparsas $\mathbf{f}_\alpha$ é classificado pelo vetor hiperplano $\mathbf{w}$, resultando no valor escalar $c = \mathbf{w}^\top \mathbf{f}_\alpha$. A classe predita pelo classificador será 1 se $c > 0$ e -1 caso contrário. Figura adaptada de [13].

Direciono o leitor ao artigo [13] mais detalhes sobre o LAST.

2.6 Análise de Custo Computacional de Técnicas de Classificação

Como mencionado no Capítulo 1, uma possível métrica para comparação de custo computacional de técnicas de classificação em *hardware* é a quantidade de operações de multiplicação seguida de acumulação (MAC, do inglês *multiply-accumulate operations*). Esse métrica é utilizada em [32] para quantificar o ganho de desempenho computacional alcançado com a abordagem que os autores propõem comparado à abordagem original de classificação de sinais de eletroencefalograma para a detecção de episódios epilépticos.

Comparada à utilização de técnicas especificas para a extração de características, o uso de dicionários geralmente resulta em maior quantidade de operações MAC, especialmente se o dicionário não for otimizado para redução de custo computacional. Isso se deve ao possível alto número de átomos do dicionário e falta de estrutura do dicionário que permita o desenvolvimento de algoritmos rápidos de transformação, como a transformada rápida de Fourier (FFT), que é um algoritmo rápido para a DFT. O algoritmo da FFT

explora a estrutura da DFT para reduzir o número de operações MAC de forma a baixar a complexidade computacional de $O(n^2)$ para $O(n \log n)$, onde n é a dimensão do sinal.

Entretanto, não faz parte desta pesquisa a comparação do custo computacional em tempo de teste de diferentes algoritmos de classificação. O foco desta pesquisa é somente a proposição e avaliação de técnicas que permitem reduzir o custo computacional em tempo de teste de classificadores baseados em representação esparsa em dicionários de análise. Portanto, a métrica que utilizei para quantificar a redução desse custo computacional é baseada na quantidade de bits necessária para a classificação em tempo de teste.

Cabe lembrar que cada multiplicação é realizada em *hardware* com diversos deslocamento de bits seguidos de somas. Portanto, a técnica que proponho de aproximação de uma multiplicação por um simples deslocamento de bit, apresentada na Seção 3.2, por si só já reduz o custo computacional do classificador em tempo de teste. Entretanto, não há como quantificar essa redução de custo computacional sem haver uma implementação em *hardware* dessa técnica. O mesmo se aplica à técnica que proponho que possibilita a troca de operações em ponto flutuante por operações em inteiro, apresentada na Seção 3.3.

Capítulo 3

Técnicas Propostas para a Redução do Custo Computacional da Classificação Baseada em Transformadas Aprendidas

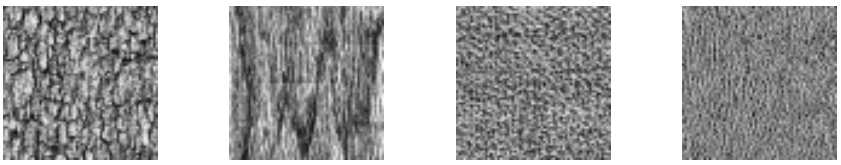
Neste capítulo, proponho técnicas para redução de custo computacional de operações realizadas durante a classificação em tempo de teste de classificadores baseados em transformadas aprendidas e limiar suave. Desenvolvi essas técnicas a partir dos achados empíricos que obtive durante estudo preliminiar da classificação em tempo de teste do algoritmo LAST. Nesse estudo preliminar utilizei duas bases de dados construídas a partir de texturas extraídas do conjunto de dados Brodatz [34], que também foram utilizadas em [13], onde o LAST foi apresentado.

Este capítulo está organizado da seguinte forma. Na Seção 3.2, proponho uma técnica que permite reduzir cada multiplicação a um simples deslocamento de bits. Na Seção 3.3, proponho que a classificação em tempo de teste seja feita com os sinais originais em vez de suas versões normalizadas e provo que essa mudança não afeta a classificação. Na Seção 3.4, proponho que a quantidade de bits da representação de sinais seja diminuída para a classificação em tempo de teste e mostro que a perda de acurácia do classificador é limitada. Na Seção 3.5, proponho modificações nos algoritmos de aprendizagem de

dicionários baseados em otimização numérica para a diminuição da quantidade de bits necessária para armazenar o dicionário de forma a diminuir a quantidade de bits necessária para efetuar a classificação em tempo de teste. Em seguida, na Seção 3.6, proponho reduzir ainda mais a quantidade de bits necessária para armazenar o dicionário aprendido, aplicando a operação limiar duro em seus elementos. Mostro também a relação entre o valor escolhido para o limiar e a acurácia de classificação. Finalmente, na Seção 4.2, descrevo as bases de dados e a metodologia que utilizei para validar as técnicas propostas.

3.1 Bases de Dados Utilizadas

As duas bases de dados utilizadas neste estudo preliminar são formadas por retalhos (do inglês, *patches*) aleatórios de quatro texturas extraídas do conjunto de imagens Brodatz [34], mostradas na Figura 3.1.



(a) *bark* (b) *woodgrain* (c) *pigskin* (d) *pressedcl*

Figura 3.1: Texturas utilizadas no estudo preliminar que executei sobre a classificação em tempo de teste de classificadores baseados em transformada aprendida. Essas texturas foram retiradas do conjunto de imagens Brodatz [34].

Assim como em [13], a primeira tarefa consistiu em construir um classificador para discriminar entre imagens de *bark* e *woodgrain*. Já a segunda tarefa foi discriminar entre imagens de *pigskin* e *pressedcl*. A construção dessas duas bases de dados foi feita com os seguinte passos:

(i) Primeiro, separei cada imagem em duas partes disjuntas, uma para treinamento e outra para teste.

(ii) Em seguida, construí o conjunto de treinamento com 500 retalhos de 12×12 píxeis retirados de forma aleatória das imagens de treinamento. Os retalhos foram dispostos

no conjunto de treinamento de forma vetorizada, no qual todas as colunas de cada retalho são colocadas em sequência em um vetor com 144 dimensões.

(iii) De forma análoga ao passo anterior, construí o conjunto de testes a partir das imagens de teste.

(iv) Por último, criei cópias do conjunto de treinamento e teste para serem os conjuntos normalizados, onde cada um dos vetores de sinais foram normalizados para terem norma ℓ_2 igual a 1.

3.2 Aproximação de Dicionário e Classificador para Potências de 2 Mais Próximas

Para facilitar o entendimento do texto a seguir, defino o operador não-linear $\mathrm{pwz}(\cdot)$ para descrever a função que mapeia os valores reais de entrada às suas respectivas potências de 2 mais próximas.

Teorema 1. *A distância euclidiana entre qualquer valor escalar real x e o inteiro dado por $x_{power} = \mathrm{pwz}(x)$ é limitada superiormente por $|1/3|$.*

Demonstração. Seja $2^n \leq x \leq 2^{n+1}$, $n \in \mathbb{Z}$ e $\mathrm{d}(x)$ a distância euclidiana entre x e $x_{power} = pwz(x)$. A distância euclidiana $\mathrm{d}(x)$ é máxima quando x é igual ao ponto médio m entre ambas as potências de 2 mais próximas, ou seja, $m = \frac{1}{2}\left(2^{n+1} + 2^n\right) = \frac{2^n}{2}\left(2 + 1\right) = 2^{n-1}\,3$.

Portanto, a distância euclidiana $\mathrm{d}(x)$ quando $x = m$ é $\mathrm{d}(m) = m - 2^n = 2^{n-1}\,3 - 2^n = 2^{n-1}\left(3 - 2\right) = 2^{n-1} = \frac{m}{3}$. Logo, a distância euclidiana máxima relativa entre x e x_{power} é $\mathrm{d}(m)/m = 1/3$. □

Mostro agora como a acurácia de classificação no conjunto de testes se comporta quando pequenas variações são introduzidas nas entrada do dicionário $\mathbf{D}$ e do vetor classificador $\mathbf{w}$ treinados com o LAST. Usando as bases de dados descritas no início deste capítulo, treinei 10 pares diferentes de $\mathbf{D}$ e $\mathbf{w}$ com 50 átomos, e criei 50 versões de cada par. Cada

um desses pares $\mathbf{D}_i$ e $\mathbf{w}_i$, $i = 1, 2, \ldots, 50$, foi formado pela multiplicação dos elementos de $\mathbf{D}$ e $\mathbf{w}$ por valores aleatórios escolhidos de uma distribuição uniforme no intervalo fechado $(1 - d_i, 1 + d_i)$, onde $d_i \in \{0.02, 0.04, 0.06, \ldots, 1\}$. Em seguida, avaliei esses modelos no conjunto de testes. Os resultados na Figura 3.2 indicam que existe claramente um compromisso entre a acurácia de classificação e o quanto que as entradas de ambos $\mathbf{D}_i$ e $\mathbf{w}_i$ estão distantes de seus respectivos valores originais, $\mathbf{D}$ e $\mathbf{w}$, sendo que essas distâncias são limitadas por d_i.

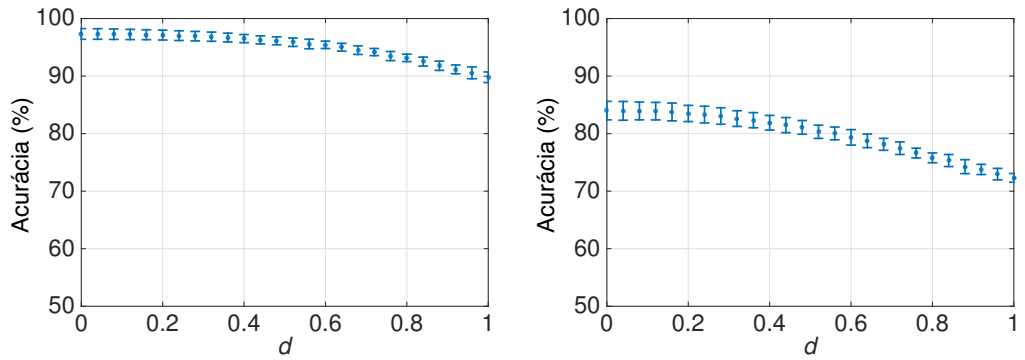


Figura 3.2: Acurácia de classificação das versões $\mathbf{D}_i$ e $\mathbf{w}_i$ de $\mathbf{D}$ e $\mathbf{w}$, com suas entradas deslocadas de seus valores originais por um valor aleatório, limitado superiormente por d. Cada resultado apresentado corresponde à média de 10 pares diferentes de $\mathbf{D}_i$ $\mathbf{w}_i$ com 50 átomos. Os resultados mostrados são referentes às bases (a) *bark* versus *woodgrain* e (b) *pigskin* versus *pressedcl*, descritas na Seção 4.2.

Hipótese 1. *Os elementos de ambos* $\mathbf{D}$ *e* $\mathbf{w}$ *podem ser aproximados para suas respectivas potências de 2 mais próximas com perda limitada de acurácia de classificação.*

Cabe lembrar que o Teorema 1 garante um limite superior de 1/3 para a distância euclidiana relativa entre qualquer valor escalar real e sua potência de 2 mais próxima. Isso sugere que a acurácia de classificação usando $\mathbf{D}_{power} = \text{pwz}(\mathbf{D})$ e $\mathbf{w}_{power} = \text{pwz}(\mathbf{w})$ seja não inferior à acurácia de classificação de $\mathbf{D}_i$ e $\mathbf{w}_i$, quando $d_i = 1/3$, mostrado na Figura 3.2. De forma a testar essa hipótese, realizei outra simulação usando as duas bases de dados descritas no início deste capítulo. Para essa simulação, treinei 10 pares de $\mathbf{D}$ e $\mathbf{w}$ com diferentes conjuntos treinamento e os avaliei com suas respectivas versões $\mathbf{D}_{power} = \text{pwz}(\mathbf{D})$ e $\mathbf{w}_{power} = \text{pwz}(\mathbf{w})$ nos conjuntos de treinamento. Apresento na Tabela 3.1 os resultados

dessa simulação. Nessa tabela, observe que a aproximação dos elementos do dicionário $\mathbf{D}$ e do classificador $\mathbf{w}$ para suas respectivas potências de 2 mais próximas produz limitada redução de acurácia nas duas bases de dados testadas.

Tabela 3.1: Comparação entre a média da acurácia de classificação de 10 pares $\mathbf{D}$ e $\mathbf{w}$ diferentes de modelos originais treinados com LAST e da média da acurácia de suas respectivas versões $\mathbf{D}_{power} = \text{pwz}(\mathbf{D})$ e $\mathbf{w}_{power} = \text{pwz}(\mathbf{w})$. As bases de dados são descritas na Seção 4.2.

	bark vs. *woodgrain*	*pigskin* vs. *pressedcl*
Original	97.33 (0.93)	84.00 (1.61)
Técnica 1	97.00 (1.06)	82.65 (1.26)

Portanto, a Técnica 1 fica da seguinte forma.

Técnica 1. *Aproximar as entradas do dicionário e do classificador para suas respectivas potências de 2 mais próximas.*

Essa técnica permite a troca de cada multiplicação por apenas um simples deslocamento de bit, que é uma das mais simples operações em *hardware*. Como discutido anteriormente, o uso dessa técnica produz um decréscimo limitado na acurácia de classificação nas bases de dados testados, sugerindo que essa técnica pode ser realizada em outras bases de dados mantendo uma limitada redução na acurácia de classificação.

3.3 Utilização de Sinais em Inteiro

Teorema 2. *Sejam $\mathbf{X}_{int}$ um conjunto de treinamento com sinais em inteiro e $\mathbf{X}$ sua versão normalizada com norma $\ell_2 = 1$. Sejam também $\mathbf{D}$ um dicionário esparsificante e $\mathbf{w}$ um classificador linear ambos treinados com o conjunto de treinamento $\mathbf{X}$. A acurácia de classificação em tempo de teste de ambos conjuntos de dados normalizados $\mathbf{X}$ e originais $\mathbf{X}_{int}$ são idênticas quando o limiar suave α para cada $\mathbf{x}_{int} \in \mathbf{X}_{int}$ é fixado em $\alpha = \|\mathbf{x}_{int}\|_2$.*

Demonstração. Sejam $\mathbf{x} \in \mathbf{X}$ e $\mathbf{x}_{int} \in \mathbf{X}_{int}$. Dessa forma, as características $\mathbf{f}$ extraídas de $\mathbf{x}$ são

$$\begin{aligned}\mathbf{f} &= \max\left(0, \mathbf{D}^\top \frac{\mathbf{x}_{int}}{\|\mathbf{x}_{int}\|_2} - 1\right) \\ &= \max\left(\frac{0}{\|\mathbf{x}_{int}\|_2}, \mathbf{D}^\top \frac{\mathbf{x}_{int}}{\|\mathbf{x}_{int}\|_2} - \frac{\|\mathbf{x}_{int}\|_2}{\|\mathbf{x}_{int}\|_2}\right) \\ &= \frac{1}{\|\mathbf{x}_{int}\|_2} \max\left(0, \mathbf{D}^\top \mathbf{x}_{int} - \|\mathbf{x}_{int}\|_2\right).\end{aligned}$$

Finalmente, a classificação de $\mathbf{x}$ é

$$c = \begin{cases} +1 & \text{se } \mathbf{w}^\top \mathbf{f}_\alpha = \mathbf{w}^\top \frac{1}{\|\mathbf{x}_{int}\|_2} \max(0, \mathbf{D}^\top \mathbf{x}_{int} - \|\mathbf{x}_{int}\|_2) > 0, \\ -1 & \text{caso contrário.} \end{cases}$$

Como todo vetor real diferente do vetor nulo possui norma ℓ_2 maior que 0, então

$$c = \begin{cases} +1 & \text{se } \mathbf{w}^\top \max(0, \mathbf{D}^\top \mathbf{x}_{int} - \|\mathbf{x}_{int}\|_2) > 0, \\ -1 & \text{caso contrário.} \end{cases}$$

Finalmente, como $\mathbf{x} = \mathbf{x}_{int}/\|\mathbf{x}_{int}\|_2$, as expressões

$$c = \mathbf{w}^\top \max(0, \mathbf{D}^\top \mathbf{x} - \alpha) > 0, \text{ com } \alpha = 1 \text{ e}$$
$$c = \mathbf{w}^\top \max(0, \mathbf{D}^\top \mathbf{x}_{int} - \alpha) > 0, \text{ com } \alpha = \|\mathbf{x}_{int}\|_2$$

são equivalentes. □

Portanto, a Técnica 2 fica da seguinte forma.

Técnica 2. *Usar sinais em sua forma original (em inteiro) em vez de normalizados (em ponto flutuante).*

Essa técnica permite o uso de operações em inteiro que são de baixo custo computacional quando comparadas a operações em ponto flutuante [4]. Como demonstrado no Teorema 2,

as classificações de sinais normalizados e de sinais em inteiro são as mesmas quando o parâmetro de esparsidade (limiar suave) é ajustado para $\alpha = \|\mathbf{x}_{int}\|_2$.

3.4 Quantização dos Sinais de Teste

Evidência Empírica 1. *A diminuição do nível de quantização dos sinais representados em inteiro do conjunto* $\mathbf{X}_{int}$ *até um certo limite causa diminuição da faixa dinâmica de* $\mathbf{X}_{int}$ *ao custo de uma diminuição limitada da acurácia da classificação.*

Levantei também a hipótese de que os sinais originais representados em inteiro estejam desnecessariamente super quantizados e que eles podem ser quantizados com um número menor de níveis sem que haja redução substancial da acurácia de classificação. De forma a testar essa hipótese, construí e realizei outra simulação com as bases de dados descritas no início deste capítulo. Nessa simulação, tomei a média dos resultados de 10 pares de $\mathbf{D}$ e $\mathbf{w}$ treinados e avaliados com diferentes conjuntos de treinamento e testes, respectivamente. Para cada par $\mathbf{D}$ e $\mathbf{w}$, quantizei o conjunto de testes $\mathbf{X}_{test}$ variando de 1 a 15 níveis de quantização. Os resultados estão na Figura 3.3. É interessante notar nessa figura que os sinais dos conjuntos de testes de ambas bases podem ser quantizados a níveis tão baixos quanto 2 e 3 com uma pequena diminuição da acurácia de classificação, chegando a necessitar de apenas 2 bits para representar o conjunto de testes.

Portanto, a Técnica 3 fica da seguinte forma.

Técnica 3. *Diminuir a faixa dinâmica do conjunto de testes* $\mathbf{X}_{int}$ *quantizando seus sinais para níveis menores.*

Essa técnica permite diminuir a quantidade de bits necessária para a representação esparsa dos sinais de testes com uma possível redução de acurácia de classificação. O nível de quantização é aprendido a partir dos dados de treinamento, de forma que o número de bits seja reduzido e a perda de acurácia seja limitada.

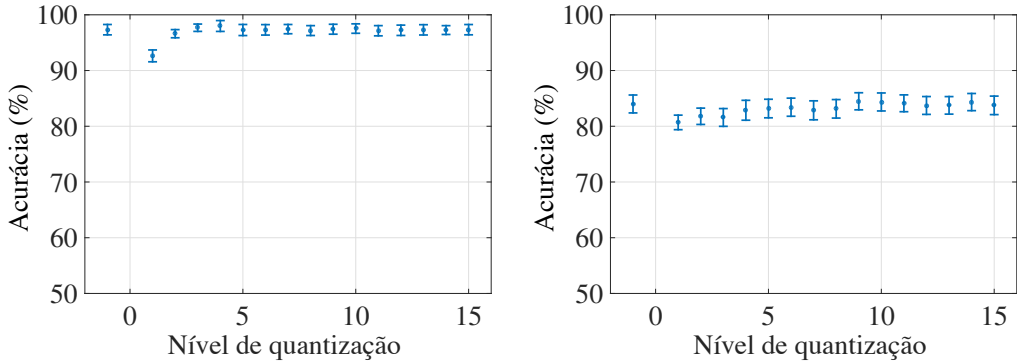


Figura 3.3: Acurácia de classificação em conjunto de testes com diferentes niveis de quantização para as bases de dados (a) *bark* versus *woodgrain* e (b) *pigskin* versus *pressedcl*. Esses resultados são as médias da classificação no conjunto de testes avaliados com 10 diferentes pares de $\mathbf{D}$ e $\mathbf{w}$, com 50 átomos, treinados com diferentes conjuntos de treinamento. Os resultados originais são mostrados nos gráficos na posição -1 do eixo de nível de quantização. As bases de dados utilizadas são descritas no início deste capítulo.

3.5 Redução da Faixa Dinâmica do Dicionário Durante Seu Treinamento

A técnica que proponho para a diminuição do faixa dinâmica do dicionário $\mathbf{D}$ consiste na adição de uma penalização à função objetivo do algoritmo de treinamento utilizado para aprendê-lo. Proponho duas estratégias descritas a seguir.

A primeira estratégia que proponho consiste na adição da penalização norma ℓ_2 de $\mathbf{D}$, que visa penalizar as entradas do dicionário que possuem alta energia. No caso do algoritmo LAST que utilizei como estudo de caso nesta pesquisa, cuja função objetivo original está descrita em 2.12, o novo problema de otimização para o treinamento incluindo a penalização fica da forma

$$\arg\min_{\mathbf{D},\mathbf{w}} \sum_{i=1}^{m} L(y_i \mathbf{w}^\top h_\alpha(\mathbf{D}^\top \mathbf{x_i})) + \frac{v}{2}\|\mathbf{w}\|_2^2 + \frac{\kappa}{2}\|\mathbf{D}\|_F^2, \tag{3.1}$$

onde κ controla o peso dessa penalização e a $\|\cdot\|_F$ é a norma de Frobenius.

Portanto, a Técnica 4 fica da seguinte forma.

Técnica 4. *Diminuir a faixa dinâmica das entradas do dicionário penalizando sua norma ℓ_2 durante o treinamento.*

Na Seção 3.5.1, mostro como essa penalização pode ser incluída em algoritmos gerais de otimização numérica com restrição. Mais especificamente, mostro como essa penalização pode ser incluída em métodos do gradiente (do inglês, *gradient descent methods*) e, de forma mais geral, em métodos de Newton.

A segunda estratégia que proponho é adicionar um termo que penaliza tanto valores com alta energia quanto com baixa energia. O objetivo é que os valores fiquem dentro de uma reduzida faixa de valores, diminuindo, assim, a faixa dinâmica do dicionário. No caso do algoritmo LAST, o novo problema de otimização fica da forma

$$\arg\min_{\mathbf{D},\mathbf{w}} \sum_{i=1}^{m} L(y_i \mathbf{w}^\top h_\alpha(\mathbf{D}^\top \mathbf{x_i})) + \frac{v}{2}\|\mathbf{w}\|_2^2 + \kappa \left[\max(\mathbf{D}) - \min(\mathbf{D})\right], \tag{3.2}$$

onde $\min(\cdot)$ e $\max(\cdot)$ são funções que retornam, respectivamente, a menor e a maior entrada de uma matriz e κ controla o peso dessa penalização. Entretanto, as funções $\min(\cdot)$ e $\max(\cdot)$ possuem derivadas complicadas e, portanto, proponho a aproximação

$$\arg\min_{\mathbf{D},\mathbf{w}} \sum_{i=1}^{m} L(y_i \mathbf{w}^\top h_\alpha(\mathbf{D}^\top \mathbf{x_i})) + \frac{v}{2}\|\mathbf{w}\|_2^2 + \kappa \frac{1}{p}\left[\|\mathbf{D}\|_p^p - (\|\overline{\mathbf{D}}\|_p)^{-p}\right], \tag{3.3}$$

onde $\overline{\mathbf{D}}$ é a matriz que contém o inverso das entradas de $\mathbf{D}$, κ controla o peso dessa penalização e p controla o nível de aproximação das funções $\min(\cdot)$ e $\max(\cdot)$.

Portanto, a Técnica 5 fica da seguinte forma.

Técnica 5. *Diminuir a faixa dinâmica das entradas do dicionário, penalizando, durante o treinamento, a diferença entre o valor máximo e o valor mínimo do dicionário, ambos aproximados utilizando ℓ_p.*

Na Seção 3.5.2, mostro como essa penalização pode ser incluída em algoritmos gerais de otimização numérica com restrição. Mais especificamente, mostro como essa penalização

pode ser incluída em métodos do gradiente (do inglês, *gradient descent methods*) e, de forma mais geral, em métodos de Newton.

3.5.1 Inclusão de Norma ℓ_2 do Dicionário como Termo de Penalização em Algoritmos de Treino de Dicionário Baseado em Otimização Numérica com Restrição

Nessa seção, mostro como incluir um termo na função objetivo que penaliza potenciais dicionários que possuem elementos com alta energia, de forma a favorecer dicionários com baixa energia. Favorecendo baixa energia, é possível obter dicionários com valores compreendidos em uma faixa de valores mais estreita. Em seguida, mostro como incluir essa penalização em métodos do gradiente, que é um dos métodos mais usados para otimização [3].

Diversos métodos de treinamento de dicionário e classificador são baseados em otimização numérica com restrição, tal como [13, 27]

$$\begin{aligned} \min_{\mathbf{d},\mathbf{w}} \quad & f(\mathbf{d},\mathbf{w}) \\ \text{sujeito a} \quad & g(\mathbf{d},\mathbf{w}) = \mathbf{0}, \end{aligned} \tag{3.4}$$

onde:

(i) $\mathbf{d}$ é um vetor $n_1 \times 1$ que contém os termos do dicionário e $\mathbf{w}$ é um vetor $n_2 \times 1$ de parâmetros do classificador, onde n_1 é o número de elementos de $\mathbf{D}$ e n_2 é o número de elementos de $\mathbf{w}$;

(ii) $f : \mathbb{R}^{n_1+n_2} \to \mathbb{R}$ é a função de custo baseada no conjunto de treinamento;

(iii) $\mathbf{0}$ é vetor nulo;

(iv) e $g : \mathbb{R}^r \to \mathbb{R}$ é uma função que representa r restrições descritas como equações.

Alguns métodos também incluem restrições descritas como inequações.

De forma a penalizar a energia total associada às entradas do dicionário, é possível trocar qualquer problema da forma (3.4) por

$$\begin{aligned} \min_{\mathbf{d},\mathbf{w}} \quad & f(\mathbf{d},\mathbf{w}) + \kappa \frac{1}{2}\|\mathbf{d}\|_2^2 \\ \text{sujeito a} \quad & g(\mathbf{d},\mathbf{w}) = \mathbf{0}, \end{aligned} \tag{3.5}$$

onde $\kappa > 0$ é o peso da penalização.

Métodos iterativos são comumente utilizados para solucionar problemas de otimização com restrição [3, 24] tais como (3.5). Eles iniciam com um valor $\gamma^0 = [\mathbf{d}^0 \;\; \mathbf{w}^0]^T$ para $\gamma = [\mathbf{d} \;\; \mathbf{w}]^T$, o qual é iterado para gerar uma suposta sequência convergente $\gamma^{(i)}$ satisfazendo

$$\gamma^{(i+1)} = \gamma^{(i)} + \xi \Delta\gamma^{(i)}, \;\; \forall \; i \geq 0, \tag{3.6}$$

onde ξ é o tamanho do passo e $\Delta\gamma^{(i)} = [\Delta\mathbf{d}^{(i)} \;\; \Delta\mathbf{w}^{(i)}]$ é o passo calculado com o método iterativo em questão.

Considero nesta pesquisa o método do gradiente, em que o cálculo do $\Delta\gamma^{(i)}$ requer avaliar o gradiente de uma função dual associada com a função objetivo e as suas restrições [3]. De forma específica, o Lagrangiano $L(\mathbf{d},\mathbf{w})$ é um exemplo de função dual, e, portanto, possui máximo local que é o mínimo da função objetivo em um ponto que satisfaz as restrições. Para os problemas (3.4) e (3.5), as funções Lagrangianas são dadas respectivamente por

$$L(\mathbf{d},\mathbf{w},\lambda) = f(\mathbf{d},\mathbf{w}) + \lambda^T g(\mathbf{d},\mathbf{w}) \text{ e} \tag{3.7}$$

$$\hat{L}(\mathbf{d},\mathbf{w},\lambda) = f(\mathbf{d},\mathbf{w}) + \lambda^T g(\mathbf{d},\mathbf{w}) + \kappa \frac{1}{2}\|\mathbf{d}\|_2^2, \tag{3.8}$$

onde λ é o vetor de m multiplicadores de Lagrange.

O primeiro objetivo na solução do problema modificado (3.5) é calcular o gradiente de $\hat{L}(\mathbf{d},\mathbf{w},\lambda)$ em termos do gradiente de $L(\mathbf{d},\mathbf{w},\lambda)$, de forma a mostrar como um problema que soluciona (3.4) pode ser modificado para solucionar (3.5).

Incluindo o Termo de Penalização em Métodos do Gradiente

Nos métodos do gradiente, o passo $\Delta\gamma^{(i)}$ depende diretamente do gradiente da função dual $\hat{L}(\gamma)$, avaliada em $\gamma^{(i)} = [\mathbf{d}^{(i)} \ \ \mathbf{w}^{(i)}]^T$ [3]. Dessa forma, o próximo passo consiste em estabelecer a relação entre $\hat{L}$ e L e com isso determinar a modificação necessária nesses métodos para incluir a penalização que proponho.

Comparando (3.7) e (3.8) e definindo $\nabla_{\mathbf{v}} g$ como o gradiente de qualquer função g com respeito ao vetor $\mathbf{v}$, note que

$$\nabla_{\mathbf{d}} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\mathbf{d}} L(\mathbf{d}, \mathbf{w}, \lambda) + \kappa \nabla_{\mathbf{d}} \frac{1}{2} \|\mathbf{d}\|_2^2,$$
$$\nabla_{\mathbf{w}} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\mathbf{w}} L(\mathbf{d}, \mathbf{w}, \lambda) \text{ e}$$
$$\nabla_{\lambda} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\lambda} L(\mathbf{d}, \mathbf{w}, \lambda).$$

Como $\nabla_{\mathbf{d}} \frac{1}{2} \|\mathbf{d}\|_2^2 = \mathbf{d}$, segue que $\nabla_{\mathbf{d}} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\mathbf{d}} L(\mathbf{d}, \mathbf{w}, \lambda) + \kappa \mathbf{d}$.

Em suma, o gradiente do Lagrangiano modificado $\hat{L}$ pode ser calculado a partir do Lagrangiano original com as expressões

$$\nabla_{\mathbf{d}} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\mathbf{d}} L(\mathbf{d}, \mathbf{w}, \lambda) + \kappa \mathbf{d}, \tag{3.9}$$
$$\nabla_{\mathbf{w}} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\mathbf{w}} L(\mathbf{d}, \mathbf{w}, \lambda) \text{ e} \tag{3.10}$$
$$\nabla_{\lambda} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\lambda} L(\mathbf{d}, \mathbf{w}, \lambda). \tag{3.11}$$

As equações (3.9), (3.10) e (3.11) mostram como fica a modificação do gradiente estimado em qualquer método do gradiente, tal como no algoritmo LAST [13], de forma a penalizar a largura da faixa das entradas do dicionário e, assim, forçar a solução a ter uma faixa mais estrita. Note que somente o gradiente com respeito ao dicionário é modificado.

Incluindo o Termo de Penalização em Métodos de Newton

No caso de métodos de Newton, o passo $\Delta\gamma^{(i)}$ em (3.6) é [3]

$$\Delta\gamma^{(i)} = -\left[\hat{H}(\mathbf{d}^{(i)}, \mathbf{w}^{(i)}, \lambda^{(i)})\right]^{-1} \nabla\hat{L}(\mathbf{d}^{(i)}, \mathbf{w}^{(i)}, \lambda^{(i)}),$$

onde $\hat{H}(\mathbf{d}^{(i)}, \mathbf{w}^{(i)}, \lambda^{(i)})$ é a matriz Hessiana de $\hat{L}(\mathbf{d}^{(i)}, \mathbf{w}^{(i)}, \lambda^{(i)})$.

A relação entre $\hat{L}$ e L é a mesma que a apresentada na Seção 3.5.1, portanto as mesmas modificações em relação ao gradiente devem ser feitas em métodos de Newton. A seguir, estabeleço a relação entre a Hessiana $\hat{H}$ de $\hat{L}$ e a Hessiana H de L.

Observe que a Hessiana é o Jacobiano do gradiente, e, portanto, a partir de (3.9), (3.10), and (3.11), respectivamente, tem-se que

$$\hat{H}_{\mathbf{d}}(\mathbf{d}, \mathbf{w}, \lambda) = H_{\mathbf{d}}(\mathbf{d}, \mathbf{w}, \lambda) + \kappa I_{n1\times n1}, \tag{3.12}$$

$$\hat{H}_{\mathbf{w}}(\mathbf{d}, \mathbf{w}, \lambda) = H_{\mathbf{w}}(\mathbf{d}, \mathbf{w}, \lambda), \tag{3.13}$$

e

$$\hat{H}_{\lambda}(\mathbf{d}, \mathbf{w}, \lambda) = H_{\lambda}(\mathbf{d}, \mathbf{w}, \lambda), \tag{3.14}$$

onde $\hat{H}_{\mathbf{v}}$ é a Hessiana de $\hat{L}$ com respeito a $\mathbf{v}$, $H_{\mathbf{v}}$ é a Hessiana de L com respeito a $\mathbf{v}$ e $I(n1 \times n1)$ é a matriz identidade com tamanho $n_1 \times n_1$.

Notar que (3.9), (3.10) e (3.11) mostram como o gradiente calculado a cada passo de métodos de Newton deve ser modificado de forma a penalizar a faixa das entradas do dicionário, onde (3.12), (3.13) e (3.14) mostram as modificações que devem ser feitas nas matrizes Hessianas.

3.5.2 Inclusão de Penalização de Faixa de Valores do Dicionário em Algoritmos de Treino de Dicionário Baseado em Otimização Numérica com Restrição

Diversos métodos de treinamento de dicionários e classificadores são baseados em otimização númerica com restrição, tais como [13, 36]

$$\begin{aligned} &\min_{\mathbf{d},\mathbf{w}} f(\mathbf{d},\mathbf{w}) \\ &\text{sujeito a } \; g(\mathbf{d},\mathbf{w}) = \mathbf{0}, \end{aligned} \tag{3.15}$$

onde:

(i) $\mathbf{d}$ é um vetor $n_1 \times 1$ que contém os termos do dicionário e $\mathbf{w}$ é um vetor $n_2 \times 1$ de parâmetros do classificador, onde n_1 é o número de elementos de $\mathbf{D}$ e n_2 é o número de elementos de $\mathbf{w}$;

(ii) $f : \mathbb{R}^{n_1+n_2} \to \mathbb{R}$ é a função de custo baseada no conjunto de treinamento;

(iii) $\mathbf{0}$ é vetor nulo;

(iv) e $g : \mathbb{R}^r \to \mathbb{R}$ é uma função que representa r restrições descritas como equações.

Alguns métodos também incluem restrições descritas como inequações.

Nessa seção, mostro como (3.15) pode ser modificado para incluir um termo na função objetivo que penaliza potenciais dicionários que possuem larga faixa de valores. Como o número total de bits necessário para representar cada entrada do dicionário depdende nessa faixa de valores, essa penalização pode reduzir o número de bits para representar o dicionário.

Primeiramente, note que dado um vetor $\mathbf{y}$ com entrada não nulas, seus valores mínimo e máximo absoluto são respectivamente

$$\max(|\mathbf{y}|) = \|\mathbf{y}\|_{\text{Inf}}$$

e

$$\min(|\mathbf{y}|) = (\|\overline{\mathbf{y}}\|_{\text{Inf}})^{-1},$$

onde $|\mathbf{y}|$ é o vetor que contém os valores absolutos de $\mathbf{y}$, $\overline{\mathbf{y}}$ é o vetor que contém os recíprocos das entradas de $\mathbf{y}$, e $\|\mathbf{v}\|_{\text{Inf}}$ é a norma infinita de qualquer vetor $\mathbf{v}$.

Portanto, de forma a penalizar a faixa total das entradas dos dicionários, pode-se trocar qualquer problema da forma (3.15) por

$$\begin{aligned} &\min_{\mathbf{d},\mathbf{w}} f(\mathbf{d},\mathbf{w}) + \kappa \left[\|\mathbf{d}\|_{\text{Inf}} - (\|\overline{\mathbf{d}}\|_{\text{Inf}})^{-1}\right] \\ &\text{sujeito a } \; g(\mathbf{d},\mathbf{w}) = \mathbf{0}, \end{aligned} \tag{3.16}$$

onde $\kappa > 0$ é o peso da penalização e $\mathbf{d}$ é o vetor que contém os termos do dicinário.

Além disso, de forma a obter um problema de otimização mais tratável, sugiro a troca da norma infinita pela norma ℓ_p definida por

$$\|\mathbf{v}\|_p = \left(\sum_{i=1}^{n_1} |v_i|^p\right)^{\frac{1}{p}},$$

para qualquer vetor n_1-dimensional $\mathbf{v}$, pois $\|\mathbf{v}\|_p$ aproxima $\|\mathbf{v}\|_{\text{Inf}}$ quando p aumenta. Assim, (3.16) pode ser substituido por

$$\begin{aligned} &\min_{\mathbf{d},\mathbf{w}} f(\mathbf{d},\mathbf{w}) + \kappa \cdot \frac{1}{p} \left[\|\mathbf{d}\|_p^p - (\|\overline{\mathbf{d}}\|_p)^{-p}\right] \\ &\text{sujeito a } \; g(\mathbf{d},\mathbf{w}) = \mathbf{0}, \end{aligned} \tag{3.17}$$

para p suficientemente alto. Observe que, para facilitar a derivação, uso a potência p da norma ℓ_p. Essa troca resulta em um problema de otimização equivalente que possui derivação mais conveniente de ser calculada.

Métodos iterativos são comumente utilizados para resolver problemas de otimização numérica com restrição [3, 24], tais como (3.17). Esses métodos iniciam a partir de um

valor inicial $\gamma^0 = [\mathbf{d}^0 \ \ \mathbf{w}^0]^T$ para $\gamma = [\mathbf{d} \ \ \mathbf{w}]^T$, que é subsequentemente iterado para gerar uma sequência supostamente convergente $\gamma^{(i)}$ satisfazendo

$$\gamma^{(i+1)} = \gamma^{(i)} + \xi \Delta\gamma^{(i)}, \ \ \forall \ n \geq 0, \tag{3.18}$$

onde ξ é o tamanho do passo e $\Delta\gamma^{(i)} = [\Delta\mathbf{d}^{(i)}; \ \Delta\mathbf{w}^{(i)}]$ é o passo computado baseado no método iterativo utilizado.

Considero dois tipo de métodos: método do gradiente e método de Newton. Em ambos, o cálculo de $\Delta\gamma^{(i)}$ requer avaliar o gradiente da função dual associada à função objetiva e as restrições [3]. De forma mais específica, o Lagrangiano $L(\mathbf{d}, \mathbf{w})$ é um exemplo de função dual, e portanto possui um máximo local que é um mínimo local da função objetivo em um ponto que satisfaz as restrições. Para os problemas (3.15) e (3.17), as funções Lagrangianas são respectivamente

$$L(\mathbf{d}, \mathbf{w}, \lambda) = f(\mathbf{d}, \mathbf{w}) + \lambda^T g(\mathbf{d}, \mathbf{w}) \tag{3.19}$$

e

$$\hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = f(\mathbf{d}, \mathbf{w}) + \kappa \cdot \frac{1}{p} \left[\|\mathbf{d}\|_p^p - (\|\overline{\mathbf{d}}\|_p)^{-p} \right] + \lambda^T g(\mathbf{d}, \mathbf{w}), \tag{3.20}$$

onde λ é o vetor que contém os m multiplicadores de Lagrange.

O primeiro passo para encontrar a solução do problema modificado (3.17) é calcular $\hat{L}(\mathbf{d}, \mathbf{w}, \lambda)$ em termos de $L(\mathbf{d}, \mathbf{w}, \lambda)$, de forma a mostrar que um problema que soluciona (3.15) pode ser modificado de forma a solucionar (3.17).

Incluindo o Termo de Penalização em Métodos do Gradiente

Em métodos do gradiente, o passo $\Delta\gamma^{(i)}$ depende diretamente no gradiente da função dual $\hat{L}(\gamma)$ avaliada em $\gamma^{(i)} = [\mathbf{d}^{(i)} \ \ \mathbf{w}^{(i)}]^T$ [3]. Estabeleço agora a relação

Nos métodos do gradiente, o passo $\Delta\gamma^{(i)}$ depende diretamente do gradiente da função dual $\hat{L}(\gamma)$, avaliada em $\gamma^{(i)} = [\mathbf{d}^{(i)} \ \ \mathbf{w}^{(i)}]^T$ [3]. Dessa forma, o próximo passo consiste em estabelecer a relação entre $\hat{L}$ e L e com isso determinar a modificação necessária nesses métodos para incluir a penalização que proponho.

Comparando (3.19) e (3.20) e definindo $\nabla_{\mathbf{v}} g$ como o Lagrangiano de qualquer função g com respeito ao vetor $\mathbf{v}$, note que

$$\nabla_{\mathbf{d}} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\mathbf{d}} L(\mathbf{d}, \mathbf{w}, \lambda) + \kappa \nabla_{\mathbf{d}} \left[\frac{1}{p} \|\mathbf{d}\|_p^p - \frac{1}{p} (\|\overline{\mathbf{d}}\|_p)^{-p} \right],$$

$$\nabla_{\mathbf{w}} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\mathbf{w}} L(\mathbf{d}, \mathbf{w}, \lambda)$$

e

$$\nabla_{\lambda} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\lambda} L(\mathbf{d}, \mathbf{w}, \lambda).$$

Agora, para calcular o termo $\nabla_{\mathbf{d}} \left[\frac{1}{p} \|\mathbf{d}\|_p^p - \frac{1}{p} (\|\overline{\mathbf{d}}\|_p)^{-p} \right]$, defino o termo

$$h(\mathbf{d}) = \kappa \frac{1}{p} \|\mathbf{d}\|_p^p - \kappa \left[\frac{1}{p} \|\overline{\mathbf{d}}\|_p^p \right]^{-1},$$

de forma que

$$h(\mathbf{d}) = h_1(\mathbf{d}) + h_2(\mathbf{d}), \tag{3.21}$$

com

$$h_1(\mathbf{d}) = \frac{1}{p} \|\mathbf{d}\|_p^p \tag{3.22}$$

e

$$h_2(\mathbf{d}) = \left[\frac{1}{p}\|\overline{\mathbf{d}}\|_p^p\right]^{-1}. \tag{3.23}$$

Desenvolvendo (3.23) e calculando suas derivadas parciais restritas a p par,

$$h_2(\mathbf{d}) = \frac{1}{\frac{1}{p}\left(\frac{1}{d_1^p} + \frac{1}{d_2^p} + \frac{1}{d_3^p} + \ldots + \frac{1}{d_{n_1}^p}\right)};$$

$$\frac{\partial h_2(\mathbf{d})}{\partial d_k} = \frac{1}{\frac{1}{p}\left(\frac{1}{d_1^p} + \frac{1}{d_2^p} + \frac{1}{d_3^p} + \ldots + \frac{1}{d_{n1}^p}\right)^2} \cdot \frac{1}{p} \cdot d_k^{p-1};$$

$$\frac{\partial h_2(\mathbf{d})}{\partial d_k} = \frac{d_k^{p-1}}{\|\overline{\mathbf{d}}\|_p^{2p}}, \tag{3.24}$$

lembrando que $\overline{\mathbf{d}}$ é o vetor cuja entradas são os respectivos recíprocos das entradas de $\mathbf{d}$.

A partir de (3.22) e (3.24), os gradientes de h_1 e h_2 são respectivamente

$$\nabla h_1(\mathbf{d}) = \mathbf{d}^{p-1}$$

e

$$\nabla h_2(\gamma) = \frac{1}{\|\overline{\gamma}\|_p^{2p}} \cdot \gamma^{p-1},$$

onde

$$\mathbf{d}^{p-1} = [d_1^{p-1} \;\; d_2^{p-1} \;\; \ldots \;\; d_{n1}^{p-1}]^T.$$

Portanto,

$$\nabla h(\mathbf{d}) = \kappa \mathbf{d}^{p-1} - \kappa \frac{1}{\|\overline{\mathbf{d}}\|_p^{2p}} \cdot \mathbf{d}^{p-1}.$$

Em suma, a partir do Lagrangiano original usado em um determinado problema de otimização, o gradiente do Lagrangiano modificado $\hat{L}$ pode ser computado com o uso das expressões

$$\nabla_{\mathbf{d}} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\mathbf{d}} L(\mathbf{d}, \mathbf{w}, \lambda) + \kappa \mathbf{d}^{p-1} - \kappa \frac{1}{\|\overline{\mathbf{d}}\|_p^{2p}} \cdot \mathbf{d}^{p-1}, \tag{3.25}$$

$$\nabla_{\mathbf{w}} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\mathbf{w}} L(\mathbf{d}, \mathbf{w}, \lambda) \tag{3.26}$$

e

$$\nabla_{\lambda} \hat{L}(\mathbf{d}, \mathbf{w}, \lambda) = \nabla_{\lambda} L(\mathbf{d}, \mathbf{w}, \lambda). \tag{3.27}$$

As equações (3.25), (3.26) e (3.27) mostram como os gradientes estimados podem ser modificados em qualquer método do gradiente (tal como o LAST [13]) de forma a penalizar a faixa de valores das entradas do dicionário e, assim, forçar a encontrar uma solução que possua uma faixa de valores mais estreita. Na próxima seção, mostro as modificações para o caso de métodos de Newton.

Incluindo o Termo de Penalização em Métodos de Newton

No caso de métodos de Newton, o passo $\Delta\gamma^{(i)}$ in (3.18) é dado por [3]

$$\Delta \mathbf{x}^{(i)} = - \left[\hat{H}(\mathbf{d}^{(i)}, \mathbf{w}^{(i)}, \lambda^{(i)}) \right]^{-1} \nabla \hat{L}(\mathbf{d}^{(i)}, \mathbf{w}^{(i)}, \lambda^{(i)}),$$

onde $\hat{H}(\mathbf{d}^{(i)}, \mathbf{w}^{(i)}, \lambda^{(i)})$ é a matriz Hessiana de $\hat{L}(\mathbf{d}^{(i)}, \mathbf{w}^{(i)}, \lambda^{(i)})$.

A relação entre $\hat{L}$ e L é a mesma que a apresentada na Seção 3.5.2, portanto as mesmas modificações em relação ao gradiente devem ser feitas em métodos de Newton. A seguir, estabeleço a relação entre a Hessiana $\hat{H}$ de $\hat{L}$ e a Hessiana H de L.

Observe que a Hessiana é o Jacobiano do gradiente, e, portanto, a partir de (3.25), (3.26), and (3.27), respectivamente, tem-se que

$$\hat{H}_{\mathbf{d}}(\mathbf{d}, \mathbf{w}, \lambda) = H_{\mathbf{d}}(\mathbf{d}, \mathbf{w}, \lambda) + \kappa I[(p-1)\mathbf{d}^{p-2}] - \kappa J(\mathbf{d}), \tag{3.28}$$

$$\hat{H}_{\mathbf{w}}(\mathbf{d}, \mathbf{w}, \lambda) = H_{\mathbf{w}}(\mathbf{d}, \mathbf{w}, \lambda), \tag{3.29}$$

e

$$\hat{H}_{\lambda}(\mathbf{d}, \mathbf{w}, \lambda) = H_{\lambda}(\mathbf{d}, \mathbf{w}, \lambda), \tag{3.30}$$

onde $\hat{H}_{\mathbf{v}}$ é a Hessiana de $\hat{L}$ com respeito a $\mathbf{v}$, $H_{\mathbf{v}}$ é a Hessiana de L com respeito a $\mathbf{v}$, $I(\mathbf{v})$ é a matriz diagonal cuja diagonal principal é o vetor $\mathbf{v}$ e $J(\mathbf{d})$ é a matriz definida por

$$J_{(i,j)}(\mathbf{d}) = \begin{cases} \dfrac{(p-1)d_i^{p-2}}{\|\overline{\mathbf{d}}\|_p^{2p}} + \dfrac{2pd_i^{-2}}{\|\overline{\mathbf{d}}\|_p^{3p}} & \text{se } i = j,\ 1 \leq i \leq n_1 \\ \dfrac{2pd_i^{p-1}d_j^{-p-1}}{\|\overline{\mathbf{d}}\|_p^{3p}} & \text{caso contrário.} \end{cases}$$

Notar que (3.25), (3.26) e (3.27) mostram como o gradiente calculado a cada passo de métodos de Newton deve ser modificado de forma a penalizar a faixa das entradas do dicionário, onde (3.28), (3.29) e (3.30) mostram as modificações que devem ser feitas nas matrizes Hessianas.

3.6 Quantizar o Dicionário por Meio de Limiarização

Evidência Empírica 2. *Quantizar o dicionário* $\mathbf{D}$ *até um certo nível irá diminuir o número mínimo de bits necessários para armazená-lo e, consequentemente, irá reduzir o número de bits necessários para se computar a representação esparsa* $\mathbf{D}^\top\mathbf{X}$ *ao custo de uma limitada diminuição da acurácia de classificação.*

Levantei a hipótese de que quantizar $\mathbf{D}$ iria diminuir sua faixa dinâmica sem detrimento substancial da sua acurácia de classificação. Minha premissa é que esses valores próximos de zero contribuem pouco para o valor final da característica extraída e, portanto, podem ser anuladas sem que isso afete de forma substancial a acurácia de classificação. Além disso, muitos valores podem se encontrar próximos de zero, tendo em vista as técnicas propostas e descritas nas seções anteriores.

Para testar essa hipótese, realizei outra simulação utilizando as bases de dados descritas no início deste capítulo. Para tanto, primeiro treinei 10 pares $\mathbf{D}_i$ e $\mathbf{w}_i$ e em seguida criei 14 versões $\mathbf{D}_{i,l}$ de cada um dos 10 dicionários $\mathbf{D}_i$ transformados com o operador limiar duro, com l escolhido do conjunto contendo 14 valores linearmente espaçado $l = [0, 4]$. Em seguida, avaliei cada $\mathbf{D}_{i,l}$ com seu respectivo $\mathbf{w}_i$ no conjunto de testes e tirei a média da acurácia de classificação obtida para cada l utilizado.

Como mostrado na Figura 3.4(a)(c), o primeiro limiar diferente de zero já reduz para menos da metade o número de bits para armazenar $\mathbf{D}$ sem que haja perda substancial da acurácia de classificação. O mesmo ocorre na outra base de dados testada, como pode-se ver na Figure 3.4(b)(d), onde a acurácia média praticamente se mantém até o terceiro ponto de corte, reduzindo para menos da metade a quantidade de bits necessária para se armazenar $\mathbf{D}$.

Portanto, a Técnica 6 fica da seguinte forma.

Técnica 6. *Quantizar o dicionário aplicando nas suas entradas um limiar duro, sendo que o limiar é treinado para produzir a maior acurácia de classificação no conjunto de treinamento.*

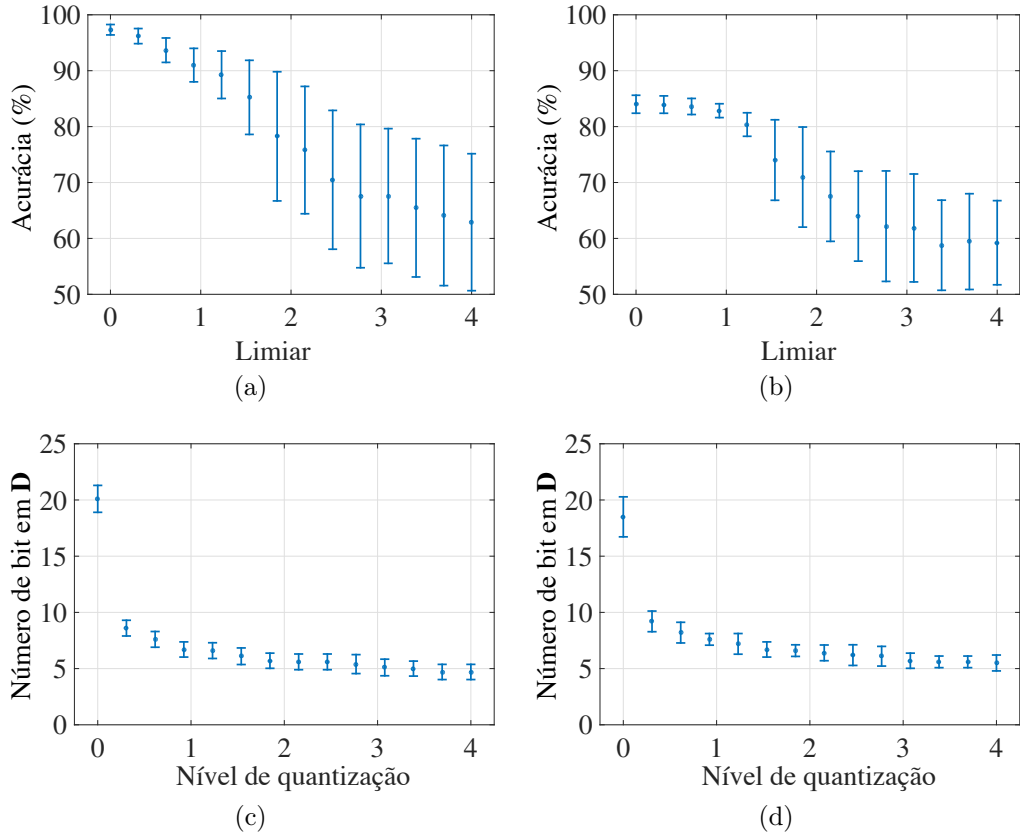


Figura 3.4: Acurácia de classificação de modelos $\mathbf{D}$ modificados com o operador limiar duro com limiar variando entre 0 e 14 para as bases de dados (a)(c) *bark* versus *woodgrain* e (b)(d) *pigskin* versus *pressedcl*. Cada resultado mostrado é a média de 10 pares diferentes de $\mathbf{D}$ e $\mathbf{w}$, com 50 átomos treinados com diferentes conjuntos de treinamento. Os resultados originais são mostrados com nível de quantização igual a zero. A descrição das bases de dados utilizadas estão no início deste capítulo.

Para o cálculo do limiar que produz a maior acurácia de classificação, é importante testar exaustivamente todos os valores únicos do absoluto de $\mathbf{D}$. Essa tarefa pode ter alto custo computacional dependendo do tamanho do dicionário e de sua quantidade de elementos distintos. Caso essa técnica seja utilizada após ser aplicada a Técnica 1, a quantidade de testes será reduzida para no máximo $\log_2(\max(|\text{pwz}(\mathbf{D})|))$, onde $|\cdot|$ é a função absoluto.

Capítulo 4

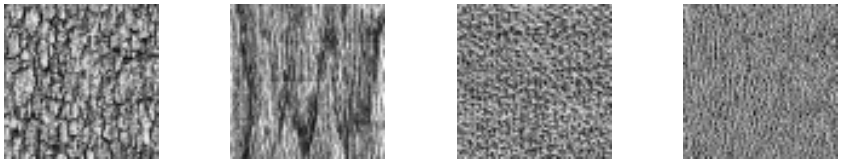
Avaliações Empíricas das Técnicas Propostas

4.1 Bases de Dados Utilizadas

Nesta pesquisa, utilizei as mesmas bases de dados usadas no artigo que descreve o algoritmo LAST [13]. Todas elas são imagens, contendo bases com texturas, fotos diversas e dígitos escritos a mão. Para avaliar as técnicas que proponho, utilizei as bases de dados originais, sem haver pré-processamento nas imagens. Entretanto, não encontrei disponível a base de dados USPS no seu formato original, representada em inteiro e, portanto, não a utilizei nas minhas simulações. Portanto, utilizei apenas cinco das seis bases de dados, sendo três binárias e duas multiclasse com 10 classes. Para cada base de dados, criei os conjuntos de treinamento e teste e suas versões pré-processadas conforme descrito em [13] para o treinamento com o algoritmo LAST.

As duas primeiras bases de dados binárias foram descritas na Seção 3.1. Copio abaixo o texto referente a essas bases para facilitar a leitura. As duas primeiras bases de dados utilizadas são formadas por retalhos (do inglês, *patches*) aleatórios de quatro texturas extraídas do conjunto de imagens Brodatz [34], mostradas na Figura 4.1.

Assim como em [13], a primeira tarefa consistiu em construir um classificador para



(a) *bark* (b) *woodgrain* (c) *pigskin* (d) *pressedcl*

Figura 4.1: Texturas utilizadas no estudo preliminar que executei sobre a classificação em tempo de teste de classificadores baseados em transformada aprendida. Essas texturas foram retiradas do conjunto de imagens Brodatz [34].

discriminar entre imagens de *bark* e *woodgrain*. Já a segunda tarefa foi discriminar entre imagens de *pigskin* e *pressedcl*. A construção dessas duas bases de dados foi feita com os seguinte passos:

(i) Primeiro, separei cada imagem em duas partes disjuntas, uma para treinamento e outra para teste.

(ii) Em seguida, construí o conjunto de treinamento com 500 retalhos de 12×12 píxeis retirados de forma aleatória das imagens de treinamento. Os retalhos foram dispostos no conjunto de treinamento de forma vetorizada, no qual todas as colunas de cada retalho são colocadas em sequência em um vetor com 144 dimensões.

(iii) De forma análoga ao passo anterior, construí o conjunto de testes a partir das imagens de teste.

(iv) Por último, criei cópias do conjunto de treinamento e teste para serem os conjuntos normalizados, onde cada um dos vetores de sinais foram normalizados para terem norma ℓ_2 igual a 1.

Assim como em [13], construí a terceira base de dados binária utilizando um subconjunto da base de dados CIFAR-10 [18]. Essa base de dados contém 10 classes de 60 000 imagens em RGB, sendo que o conjunto de treinamento contém 50 000 imagens e o conjunto de testes contém 10 000. Cada imagem possui 32×32 píxeis e 3 canais de cor e é armazenada em um vetor com $32 \times 32 \times 3 = 3\,072$ dimensões. A terceira base de dados binária que

utilizei é o subconjunto formado pelas imagens das classes *deer* e *horse*. A Figura 4.2 contém exemplos dessas classes juntamente com exemplos das demais classes.

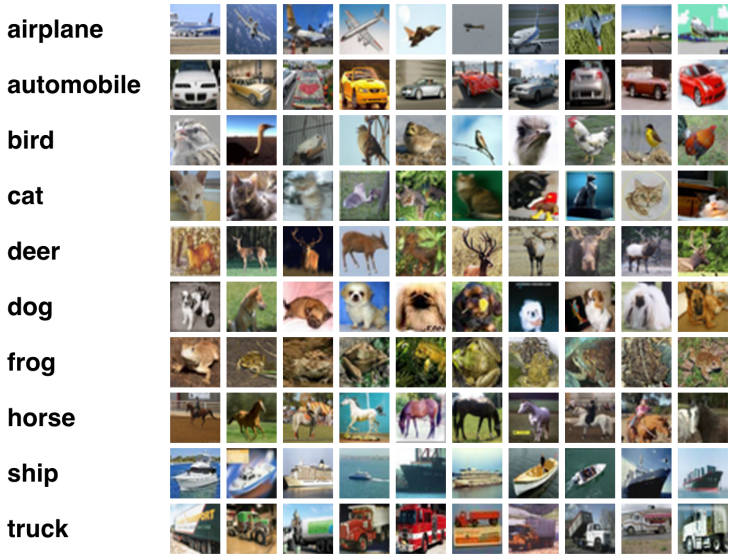


Figura 4.2: Exemplos das dez classes que constituem o conjunto de imagens CIFAR-10 [18]. Extraído de http://www.cs.toronto.edu/~kriz/cifar.html

A primeira base de dados multiclasse que utilizei é a MNIST [20], que contém 70.000 imagens de dígitos escritos a mão de tamanho 28×28 píxeis distribuídos em 60.000 imagens para treinamento e 10.000 imagens para teste. Assim como em [13], as imagens pré-processadas para têm média igual a zero e norma ℓ_2 igual a 1.

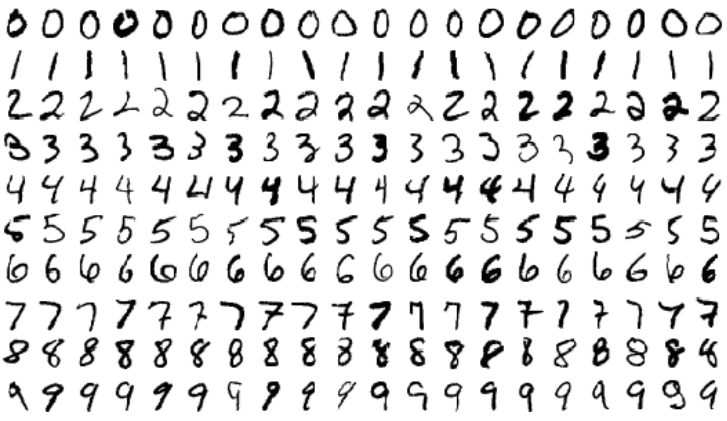


Figura 4.3: Exemplos das dez classes que constituem o conjunto de imagens MNIST [20]. Extraído de http://myselph.de/mnistExamples.png

A segunda base de dados multiclasse que utilizei é a própria base de dados CIFAR-10 com todas as 10 classes.

4.2 Descrição das Simulações

Depois de construídas as bases de dados, reservei 80% do conjunto de treinamento para a geração de $\mathbf{D}$ e $\mathbf{w}$ e utilizei os demais 20% para estimar os melhores valores de parâmetros utilizados na Técnica 3 e na Técnica 4. Observe que utilizei essa mesma abordagem de reservar 80% também para o treinamento do modelo original, $\mathbf{D}_{original}$ e $\mathbf{w}_{original}$, e os demais 20% eu descartei. Isso foi necessário para que ambos os modelos originais e propostos fossem treinados com o mesmo conjunto de treinamento e avaliados no mesmo conjunto de testes. Para cada base de dados que utilizei, criei os seguintes passos para avaliar as técnicas que proponho:

(i) Treinar $\mathbf{D}_{original}$ e $\mathbf{w}_{original}$

Treinei $\mathbf{D}_{original}$ e $\mathbf{w}_{original}$ com os mesmos valores de parâmetros utilizados no algoritmo LAST. Direciono o leitor ao artigo [13] para entendimento de todos os parâmetros utilizados no LAST. Esses modelos foram utilizados para estimar a acurácia e número de bits necessária para a representação esparsa obtidas com o método LAST original.

(ii) Treinar $\mathbf{D}_{proposto}$ e $\mathbf{w}_{proposto}$

Para a construção dos modelos com a Técnica 4, utilizei os valores $\kappa = \{4, 8, 10, \ldots, 20\} \times 10^{-3}$ para controlar o peso da penalização introduzida no treinamento de $\mathbf{D}_{proposto}$. Após o treinamento de $\mathbf{D}_{proposto}$, aplico a Técnica 1 em ambos $\mathbf{D}_{proposto}$ e $\mathbf{w}_{proposto}$ para aproximar os seus elementos para suas respectivas potências de 2 mais próximas. Em seguida, aplico a Técnica 4, que consiste em aplicar um limiar duro às entrada de $\mathbf{D}_{proposto}$. O conjunto de valores que testo para o limiar `zt` é formado pelos valores únicos do absoluto de $\mathbf{D}_{proposto}$.

(iii) Criar $\mathbf{X}_{teste\_proposto}$

Em seguida, utilizo a Técnica 2 e a Técnica 3 para criar $\mathbf{X}_{teste\_proposto}$ que é a versão inteira e quantizada do conjunto de teste com o parâmetro `quanta` =

$\{, 1, 2, \ldots, 10\} \cup \{31, 127\} \cup b$, onde b é o número de níveis da quantização original dos sinais.

(iv) Avaliar modelos

Por último, avalio cada modelo que criei nos passos (i) e (ii) no conjunto de teste $\mathbf{X}_{teste\_proposto}$, criado no passo (iii) e seleciono o melhor modelo utilizando o método que criei para seleção de modelos descrito a seguir na Seção 4.3. Nessa seleção de modelos, o parâmetro γ controla o compromisso entre a acurácia do classificador e a quantidade de bits necessária para a classificação. Utilizei $\gamma = 0.001$.

A regra tradicional de se escolher 2/3 do conjunto de dados para treinamento e 1/3 para testes é uma forma segura de se estimar a acurácia de classificação quando a acurácia de classificação do modelo avaliado no conjunto de dados completo é maior que 85% [7]. Como meu objetivo no passo (i) é somente reservar parte do conjunto de treinamento para a seleção dos melhores parâmetros e não para se estimarr a verdadeira acurácia de classificação, optei pela proporção mais conservadora de 80% para treinar os modelos. A vantagem dessa escolha é o aumento da chance de se conseguir treinar o modelo com sinais de baixa representatividade no conjunto de treinamento.

4.3 Seleção dos Modelos

Quando aplicadas a um modelo composto por $\mathbf{D}$ e $\mathbf{w}$, a Técnica 3 e a Técnica 4 geram diversos novos modelos de acordo com o número de combinações formadas pelos seus respectivos parâmetros, descritos na Seção 4.2. Para a seleção do melhor modelo, criei um método que escolhe o modelo que menos diverge do modelo original, no quesito acurácia de classificação, e que, ao mesmo tempo, necessita da menor quantidade de bits para o cálculo da representação esparsa dos sinais de teste. A variável γ controla o compromisso entre a acurácia e a quantidade de bits necessária para a classificação. Esse método de seleção de modelo é composto dos seguintes passos:

(i) Seja o conjunto $\mathbf{X}_{param}$ formado para se estimar a melhor combinação dos parâmetros κ, `zt` e `quanta` e seja $\mathcal{M}$ o conjunto de todos os modelos modificados com esses parâmetros. Seja também $\mathcal{A} \leftarrow \mathcal{M}(\mathbf{X}_{param})$ o conjunto das acurácias de classificação do conjunto $\mathbf{X}_{param}$ usando os modelos $\mathcal{M}$ e seja `best_acc` a maior acurácia de $\mathcal{A}$.

(ii) A partir de $\mathcal{M}$, crio o subconjunto $\mathcal{M}_\gamma$ que contém os modelos de $\mathcal{M}$ com acurácias $\mathcal{A}_\gamma \leftarrow [\,\mathcal{A} \mid \text{acurácia}(\mathcal{A}) \geq (1-\gamma)\,\texttt{best\_acc}\,]$, onde o parâmetro γ é escolhido a priori.

(iii) A partir de $\mathcal{M}_\gamma$, crio o subconjunto $\mathcal{M}_{bits}$ que contém os modelos de $\mathcal{M}_\gamma$ com acurácias $\mathcal{A}_{bits} \leftarrow [\,\mathcal{A}_\gamma \mid \text{número de bits}(\mathcal{A}_\gamma) = \texttt{lnb}\,]$, onde `lnb` é o menor número de bits necessário para se calcular $\mathbf{D}^\top\mathbf{X}_{param}$ dentre todos os modelos de $\mathcal{M}_\gamma$.

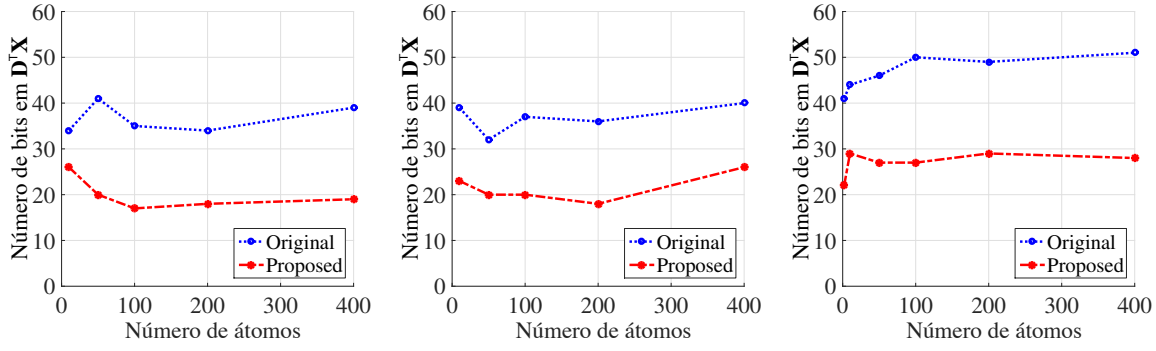
(iv) Por fim, o modelo escolhido é o $\mathbf{D} \in \mathcal{M}_{bits}$ que gera a representação $\mathbf{D}^\top\mathbf{X}_{param}$ mais esparsa.

Um ponto importante de se notar no algoritmo de seleção de modelo é que o modelo selecionado no último passo é aquele que produz a representação mais esparsa de todos os sinais de $\mathbf{X}_{param}$. Essa escolha é motivada pelo fato de que modelos que geram representações mais esparsas tendem a generalizar melhor para novos sinais [1].

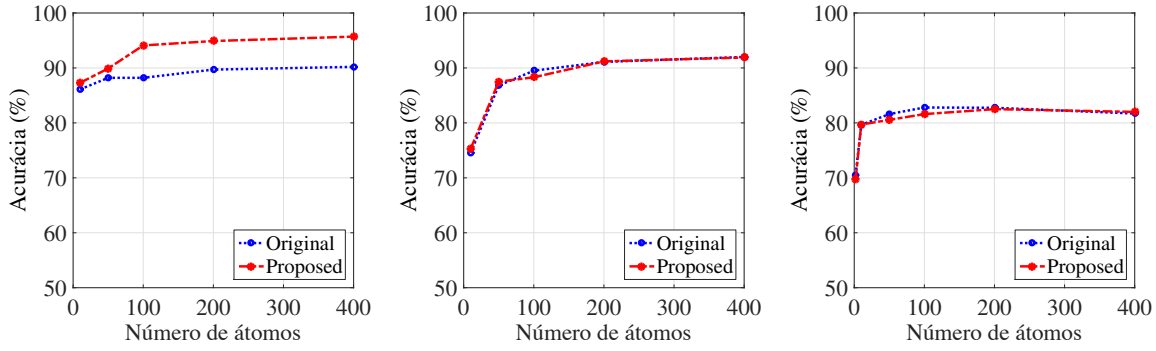
4.4 Resultados e Discussões

Nesta seção, apresento os resultados obtidos com as simulações descritas no Capítulo 3. Para maior clareza, utilizo o termo *original* para referenciar os resultados obtidos com os modelos originais $\mathbf{D}$ e $\mathbf{w}$ treinados com LAST e o termo *proposto* para referenciar os resultados obtidos com $\mathbf{D}$ e $\mathbf{w}$ modificados com as técnicas que proponho.

Os resultados obtidos nos conjuntos de testes das três bases de dados binárias são mostrados na Figura 4.4, na qual os três gráficos superiores mostram a quantidade de bits necessária para se realizar a classificação e os três gráficos inferiores mostram a acurácia de classificação.



(a) *bark* versus *woodgrain* (b) *pigskin* versus *pressedcl* (c) CIFAR-10 *deer* versus *horse*



(d) *bark* versus *woodgrain* (e) *pigskin* versus *pressedcl* (f) CIFAR-10 *deer* versus *horse*

Figura 4.4: Comparação dos resultados das simulações com o modelos originais treinados com o LAST e os modelos modificados com as técnicas que proponho. Para a classificação em tempo de teste de cada base de dados binária, os gráficos apresentam o compromisso existente entre o número de bits necessário (parte de cima) e a acurácia de classificação (parte de baixo). Observe que as técnicas que proponho reduzem para quase metade a quantidade de bits necessária com uma perda limitada de acurácia de classificação. As bases de dados utilizadas estão descritas na Seção 4.2.

Como apresentado nas Figura 4.4(d), 4.4(e) e 4.4(f) as técnicas que proponho produzem uma limitada diminuição na acurácia de classificação. Cabe lembrar que as técnicas que proponho reduzem o custo computacional de classificação em *hardware*, visto que operações em ponto flutuante são trocadas por operações em inteiro e cada multiplicação é substituída por um simples deslocamento de bit.

Além da redução promovida pela troca de operações custosas por operações mais eficientes em *hardware*, as técnicas que proponho também reduzem a quantidade de bits necessária para efetuar a computação da multiplicação $\mathbf{D}^{\top}\mathbf{X}$, que é a parte de maior custo computacional da representação esparsa. Como apresentado na Figura 4.4(a), 4.4(b) e 4.4(c), as técnicas que proponho reduzem quase pela metade o número de bits necessário

para a multiplicação $\mathbf{D}^\top\mathbf{X}$.

Perceba que as acurácias originais na Figura 4.4(d) e 4.4(e) são inferiores às apresentadas em [13], mesmo para a configuração original do LAST. A razão dessa diferença se deve ao fato de eu ter utilizado um conjunto de treinamento disjunto do conjunto de testes para não superestimar a acurácia de classificação do modelo. Além disso, treinei $\mathbf{D}$ e $\mathbf{w}$ com apenas 80% do conjunto de treinamento, conforme descrito na Seção 4.2.

Apresento na Tabela 4.1 os resultados das simulações feitas com as bases de dados MNIST e CIFAR-10. Assim como no caso da classificação binária, as técnicas que proponho permitem o uso de operações eficientes em *hardware* ao custo de uma limitada perda de acurácia de classificação.

Tabela 4.1: Comparação dos resultados originais e propostos no quesito taxa de erro de classificação e número de bits necessário para o cálculo da multiplicação $\mathbf{D}^\top\mathbf{X}$, que é a operação de maior custo computacional da classificação em tempo de teste. As bases de dados MNIST e CIFAR-10 estão descritas na Seção 4.2.

	MNIST		CIFAR-10	
	Erro %	Qtd de bits $\mathbf{D}^\top\mathbf{X}$	Erro %	Qtd de bits $\mathbf{D}^\top\mathbf{X}$
Original	2.48	61	60.00	55
Proposto	2.52	29	53.08	29

As taxas de erros que obtive para essas bases de dados multiclasse são maiores que as apresentadas em [13]. Provavelmente, isso foi causado pela natureza aleatória do LAST para bases de dados grandes, em que cada iteração de descida do gradiente o modelo é otimizado para um subconjunto dos dados de treinamento, chamada de *mini-batch*. Esse subconjunto é construído de forma aleatória a cada iteração a partir dos sinais do conjunto de treinamento. Além disso, conforme descrito na Seção 4.2, usei apenas 80% do conjunto de treinamento para gerar os modelos com o LAST e isso pode ter afetado negativamente suas capacidades de generalização para novos sinais.

Conforme apresentado na Tabela 4.1, as técnicas que proponho podem resultar em um pequeno aumento do erro de classificação da base de dados MNIST. Entretanto, assim como no caso das bases binárias, essas técnicas reduzem para quase metade o número de

bits necessário para a multiplicação $\mathbf{D}^\top\mathbf{X}$, que é a parte de maior custo computacional da representação esparsa.Cabe lembrar, novamente, que isso é benéfico para operações efetuadas em *hardware*.

Em relação aos resultados que obtive para a base de dados CIFAR-10, as técnicas que proponho produziram um modelo com taxa de erro menor que o produzido com o modelo original usando quase metade da quantidade de bits necessária para a classificação em tempo de teste. Provavelmente, essa diferença na taxa de erro também foi causada pela natureza aleatória do LAST, conforme discutido anteriormente.

Capítulo 5

Conclusão

Apresentei neste documento um conjunto de técnicas para a redução do custo computacional da classificação em tempo de teste para algoritmos de classificação baseados em transformadas e limiares suaves aprendidos a partir dos dados de treinamento. Desenvolvi essas técnicas a partir de um estudo preliminar que realizei sobre a operação mais onerosa da classificação em tempo de teste, que é a extração de características. Basicamente, essa operação é um produto matriz-vetor seguido de um limiar suave. Mostrei em seguida que essas técnicas permitem utilizar operações de baixo custo computacional e também reduzir o número de bits necessário para a extração de características com uma limitada perda de acurácia.

Neste último capítulo, apresento as conclusões sobre o estudo dessas técnicas com base na análise dos resultados das experimentações que realizei. Também sumarizo as contribuições desta pesquisa e proponho algumas sugestões de trabalhos futuro relacionados com as minhas descobertas.

De forma geral, as técnicas que propus neste documento podem ser divididas em quatro grupos:

(i) Usar os sinais no mesmo formato em que são amostrados pelos conversores analógicos digitais (ADC), ou seja, representados por inteiros, em vez de reescalonados, como normalmente é feito em processos de normalização. Em *hardware*, especialmente

em arranjo de portas programáveis por campo (FPGA, do inglês *field programmable gate array*), operações em inteiros são menos custosas do que operações em ponto flutuante. Eu provo que essa troca não afeta a acurácia de classificação.

(ii) Aproximar os valores da matriz de transformação e do vetor de classificação para suas respectivas potências de 2 mais próximas. Essa aproximação permite que cada multiplicação na classificação em tempo de teste seja substituída por um único deslocamento de bit.

(iii) Quantizar os sinais a serem classificados em tempo de teste de forma a diminuir suas respectivas faixas dinâmicas e com isso diminuir a quantidade de bits necessária para se realizar a classificação.

(iv) Diminuir a faixa dinâmica da matriz de transformação utilizando duas abordagens: primeiramente, adicionando uma penalização à sua norma; em seguida, zerando as entradas dessa matriz que possuem valores abaixo de um limiar, cujo valor é aprendido a partir dos dados de treinamento.

Os resultados apresentados no Capítulo 4.4 mostram a viabilidade de se realizar a classificação de imagens em tempo de teste com operações em inteiros no lugar de operações em ponto flutuante, além de substituir cada operação de multiplicação por um simples deslocamento de bit. Essas substituições possibilitam reduzir o custo computacional da classificação em tempo de teste realizadas em FPGAs, com importância em aplicações embarcadas, cujo consumo de energia é crítico. Além disso, os modelos gerados com essas técnicas reduziram pela metade a quantidade de bits necessária para efetuar a multiplicação matriz-vetor $\mathbf{D}^\top\mathbf{X}$, que é a operação de maior custo computacional na classificação em tempo de teste.

Também é importante notar que desenvolvi essas técnicas somente para a redução do custo computacional da classificação, com uma perda de acurácia de classificação dentro de limites aceitáveis. Entretanto, as acurácias de classificação dos modelos modificados com essas técnicas na base de dados *bark* versus *woodgrain* superaram as acurácias de

classificação do modelo original, conforme mostrado na Figura 4.4(a). Após inspecionar esses resultados, percebi que os modelos originais obtiveram acurácia de 100% no conjunto de treinamento, indicando que possivelmente eles se sobreajustaram a esse conjunto (para os casos com mais de 100 átomos). A técnica que aproxima as entradas de $\mathbf{D}$ e $\mathbf{w}$ para suas respectivas potências de dois mais próximas introduz um erro de quantização, e é possível que esse erro tenha causado um aumentado na variância desses modelos de tal forma a aumentar suas capacidades de generalização para novos sinais [25]. De qualquer forma, esse fato necessita de uma investigação mais profunda, o qual deixo como trabalho futuro. Também como trabalho futuro, é importante investigar com maior profundidade como essa técnica de aproximação para potências de dois afeta a acurácia do modelo.

Referências

[1] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: a review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, August 2013. 12, 49

[2] Sergio Bernabe, Sebastián Lopez, Antonio Plaza, and Roberto Sarmiento. GPU Implementation of an Automatic Target Detection and Classification Algorithm for Hyperspectral Image Analysis. *IEEE Geoscience and Remote Sensing Letters*, 10:221–225, March 2013. 3

[3] Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004. 6, 11, 31, 32, 33, 34, 36, 37, 38, 40

[4] Tom M Bruintjes. Design of a fused multiply-add floating-point and integer datapath. 2011. 27

[5] Ben Cope, Peter YK Cheung, Wayne Luk, and Lee Howes. Performance comparison of graphics processors to reconfigurable logic: a case study. *Computers, IEEE Transactions on*, 59(4):433–448, 2010. 4

[6] Sanjoy Dasgupta. Experiments with Random Projection. *arXiv.org*, cs.LG, January 2013. 11

[7] Kevin K Dobbin and Richard M Simon. Optimally splitting cases for training and testing high dimensional classifiers. *BMC medical genomics*, 4:31, 2011. 48

[8] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012. 4, 9, 10, 11

[9] David L Donoho and Iain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika Trust*, 81:425–455, 1994. 1, 18

[10] George H Dunteman. *Principal components analysis*. Sage, 1989. 11

[11] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing, IEEE Transactions on*, 15(12):3736–3745, 2006. 12

[12] Alexander Fabisch, Yohannes Kassahun, Hendrik Wöhrle, and Frank Kirchner. Learning in compressed space. *Neural networks : the official journal of the International Neural Network Society*, 42C:83–93, February 2013. 4

[13] Alhussein Fawzi, Mike Davies, and Pascal Frossard. Dictionary Learning for Fast Classification Based on Soft-thresholding. *International Journal of Computer Vision*, pages 1–16, November 2014. 4, 6, 13, 17, 19, 20, 22, 23, 31, 33, 35, 40, 44, 45, 46, 47, 51

[14] A Ganesh, S S Sastry, and Y Ma. Robust Face Recognition via Sparse Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. 13

[15] X Glorot, A Bordes, and Y Bengio. Deep Sparse Rectifier Neural Networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011. 19

[16] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1):489–501, 2006. 3

[17] Deepak Khosla, Yang Chen, David J Huber, Darrel J Van Buer, Kyungnam Kim, and Shinko Y Cheng. Real-time, low power neuromorphic hardware for autonomous object recognition. In *Proceedings of the SPIE*, pages 871313–871313. HRL Labs., LLC. (United States), 2013. 3

[18] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Computer Science Department, University of Toronto, 2009. 45, 46

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in neural information processing systems*, pages 1097–1105, 2012. 3

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE. Institute of Electrical and Electronics Engineers*, 86(11):2278–2324, 1998. 46

[21] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, 2013. 19

[22] J Mairal, F Bach, and J Ponce. Task-Driven Dictionary Learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):791–804, April 2012. 4, 12, 13

[23] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010. 19

[24] Jorge Nocedal and Stephen J Wright. *Numerical optimization*, volume 2. Springer, 2006. 6, 11, 32, 36

[25] Bernhard Pfahringer. Compression-Based Discretization of Continuous Attributes. In *Proc. 12th International Conference on Machine Learning*, pages 456–463. unknown, 1995. 55

[26] J L Raheja, B Ajay, and Ankit Chaudhary. Real Time Fabric Defect Detection System on an Embedded DSP Platform. *arXiv.org*, cs.CV:–, September 2014. 3

[27] Saiprasad Ravishankar and Yoram Bresler. Learning Sparsifying Transforms. *IEEE Transactions on Signal Processing*, 61(5):1072–1086, January 2013. 1, 4, 17, 31

[28] DAILY MAIL REPORTER. Google uses MORE power than Salt Lake City as vast data farms suck up electricity, September 2011. 2

[29] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks : the official journal of the International Neural Network Society*, 61:85–117, January 2015. 3

[30] Adriana Schulz, Eduardo Antônio Barros Da Silva, and Luiz Velho. *Compressive sensing*. IMPA, 2009. 13

[31] Sumit Shekhar, Vishal M Patel, and Rama Chellappa. Analysis sparse coding models for image-based classification. In *IEEE International Conference on Image Processing. Proceedings*, 2014. 4, 17

[32] Mohammed Shoaib, Niraj K Jha, and Naveen Verma. Signal Processing With Direct Computations on Compressively Sensed Data. *Schölkopf, B*, (99), February 2014. 5, 20

[33] Steven W Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Pub. San Diego, December 1997. 4

[34] K Valkealahti and E Oja. Reduced multidimensional co-occurrence histograms in texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):90–94, 1998. 22, 23, 44, 45

[35] Rosa M Ventura, Pierre Vandergheynst, and Pascal Frossard. Low-rate and flexible image coding with redundant representations. *Image Processing, IEEE Transactions on*, 15(3):726–739, 2006. 12

[36] Zhen James Xiang, Hao Xu, and Peter J Ramadge. Learning sparse representations of high dimensional data on large scale dictionaries. *Advances in neural information processing systems*, 24:900–908, 2011. 35

[37] Matthew D Zeiler, M Ranzato, Rajat Monga, M Mao, K Yang, Quoc Viet Le, Patrick Nguyen, A Senior, Vincent Vanhoucke, Jeffrey Dean, and others. On rectified linear units for speech processing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3517–3521. IEEE, 2013. 19