COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

PROJETO DE REDE NEURAL PARA COMPRESSÃO DE DADOS BASEADA EM KERNEL PCA: ANÁLISE DE TAXA-DISTORÇÃO E COMPLEXIDADE

Vitor Rosa Meireles Elias

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Rio de Janeiro
Setembro de 2015

PROJETO DE REDE NEURAL PARA COMPRESSÃO DE DADOS BASEADA EM KERNEL PCA: ANÁLISE DE TAXA-DISTORÇÃO E COMPLEXIDADE

Vitor Rosa Meireles Elias

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. José Gabriel Rodríguez Carneiro Gomes, Ph.D.

Prof. Eduardo Antônio Barros da Silva, Ph.D.

Prof. Lisandro Lovisolo, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
SETEMBRO DE 2015

Elias, Vitor Rosa Meireles

Projeto de Rede Neural para Compressão de Dados Baseada em Kernel PCA: Análise de Taxa-Distorção e Complexidade/Vitor Rosa Meireles Elias. – Rio de Janeiro: UFRJ:/COPPE, 2015.

XIII, 54 p.: il.; 29,7cm.

Orientador: José Gabriel Rodríguez Carneiro Gomes

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2015.

Referências Bibliográficas: p. 48 – 50

1. Compressão de Dados. 2. Redes Neurais. 3. Kernel PCA. 4. Quantização Vetorial. I. Gomes, José Gabriel Rodríguez Carneiro. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenheria Elétrica. III. Título.

Aos meus pais.

Agradecimentos

Agradeço à minha família por todo apoio dado em minha formação. Agradeço aos meus pais, Sandra e Roberto, por estarem ao meu lado em todas as decisões tomadas nestes anos de mestrado e pelo esforço em garantir que eu tivesse todas as condições necessárias para concluir este projeto.

Agradeço aos meus amigos de faculdade, que estiveram ao meu lado por todos os últimos anos. Em especial, agradeço a Vitor Borges, Vitor Tavares, Mauricio Pereira, Jonathan Gois e Nilson Carvalho. Agradeço aos meus amigos de fora da faculdade, em especial Vitor Santos, Pedro Marcus e Jefferson Oliveira, que estão sempre trazendo novas formas de tornar a vida mais agradável.

Aos colegas do Laboratório de Processamento Analógico e Digital de Sinais, pela ajuda fornecida durante este projeto.

Agradeço aos professores do Programa de Engenharia Elétrica, todos fundamentais para minha formação. Especialmente, agradeço ao meu orientador José Gabriel Rodríguez Carneiro Gomes, por toda paciência e dedicação em tirar minhas dúvidas e por tudo que me ensinou.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

PROJETO DE REDE NEURAL PARA COMPRESSÃO DE DADOS BASEADA EM KERNEL PCA: ANÁLISE DE TAXA-DISTORÇÃO E COMPLEXIDADE

Vitor Rosa Meireles Elias

Setembro/2015

Orientador: José Gabriel Rodríguez Carneiro Gomes
Programa: Engenharia Elétrica

Este trabalho apresenta um estudo das propriedades de um método de quantização vetorial (VQ) baseado em Análise de Componentes Principais Kernel (KPCA), com foco na complexidade e viabilidade de implementação deste método em processamento de imagens. A teoria de suporte para este método é descrita e, então, o método é comparado a métodos de quantização tradicionais, como quantização escalar e quantização vetorial com restrição de entropia. As principais características comparadas são as curvas entropia versus distorção, ilustrando o desempenho taxa-distorção dos quantizadores, e a complexidade associada ao processo de quantização, como função do custo computacional requerido em implementação digital. Finalmente, este trabalho introduz uma abordagem com restrição de complexidade para o projeto de quantizadores.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

NEURAL NETWORK DESIGN FOR DATA COMPRESSION BASED ON KERNEL PCA: RATE-DISTORTION AND COMPLEXITY ANALYSIS

Vitor Rosa Meireles Elias

September/2015

Advisor: José Gabriel Rodríguez Carneiro Gomes
Department: Electrical Engineering

This work presents a study of the properties of a vector quantization (VQ) method based on Kernel Principal Component Analysis (KPCA), focused on the complexity and viability of implementing this method in image processing. The theory supporting this method is described and then the method is compared to traditional quantization methods, as scalar quantization and entropy-constrained vector quantization. The main characteristics compared are the entropy versus distortion curves, illustrating the quantizers rate-distortion performance, and the complexity associated with the quantization process, as a function of the computational cost required in digital implementation. Finally, this work introduces a complexity-constrained approach to quantizer design.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Curvas taxa-distorção representam a forma mais direta de caracterizar o desempenho de um sistema de compressão de imagens. O crescente uso da tecnologia CMOS (*complementary metal-oxide-semiconductor*) como alternativa ao CCD (*charge-coupled device*) na última década permite que sinais possam ser processados no plano focal de câmeras CMOS [1]. Para esta abordagem, é necessária a implementação em hardware das operações de processamento. A complexidade dos algoritmos deve ser considerada como um fator importante durante o projeto de blocos para o processamento de imagens [2], uma vez que complexidade de hardware em sistemas eletrônicos é diretamente associada a custos de fabricação, consumo de energia e área utilizada em silício.

Sistemas de compressão de imagens possuem vários estágios. Este trabalho se concentra no estágio de quantização. Algumas técnicas tradicionais de quantização são SQ (*scalar quantization*), VQ (*vector quantization*) e sua variante ECVQ (*entropy-constrained vector quantization*). A quantização vetorial baseada em kernel PCA (KPCA-VQ) é proposta como uma possível alternativa à técnica utilizada em [1]. Conexões entre sistemas de compressão de dados e sistemas de reconhecimento de padrões são conhecidas [3, 4]. Particularmente, entre estas conexões, existem ligações entre kernel PCA e redes neurais artificiais (RNA) que serão discutidas nos próximos capítulos.

A aplicação de redes neurais é amplamente encontrada em implementações de técnicas de quantização [5, 6, 7]. A principal ligação entre as técnicas tradicionais de quantização e as RNAs é uma classe de redes neurais particularmente conhecida

como *self-organizing map* (SOM). Estas redes neurais possuem propriedades que a assemelham a algoritmos utilizados no projeto de quantizadores vetoriais.

1.1 Objetivo

O objetivo desta dissertação é apresentar métodos de implementação do algoritmo do KPCA-VQ e estudar seu desempenho. O desempenho é estudado através da eficiência de compressão (a ser expressa como uma curva taxa-distorção) e da complexidade associada à implementação. Comparando resultados numéricos de quantizadores baseados em KPCA com quantizadores tradicionais, visamos analisar a viabilidade do método proposto, que pode ser visto como um projeto de rede neural cuja otimização não é baseada em algoritmos clássicos, como *backpropagation*.

1.2 Estrutura

O Capítulo 2 deste trabalho fornece informações sobre os conceitos básicos de compressão de imagens. O processo de quantização e as técnicas tradicionais que são utilizadas neste trabalho são descritas, bem como redes neurais e suas aplicações em sistemas de compressão de imagens. No Capítulo 3, a técnica proposta neste trabalho é apresentada. Uma introdução sobre análise de componentes principais serve como base para o entendimento do quantizador vetorial proposto. Os métodos de projeto utilizados durante o desenvolvimento deste trabalho são apresentados no Capítulo 4. Finalmente, os quantizadores são testados e aplicados em sistemas de compressão de imagens, no Capítulo 5. O Capítulo 6 faz uma conclusão sobre os tópicos discutidos no texto.

Capítulo 2

Compressão de Imagens

A função de um sistema de compressão de dados é reduzir a quantidade de informação necessária para representar estes dados, no domínio digital ou analógico. A necessidade de realizar a compressão surge das limitações dos sistemas de armazenamento ou transmissão de dados. Atualmente, esta necessidade é evidenciada quando arquivos digitais precisam ser armazenados em um dispositivo móvel, como um *Pen Drive*, cuja capacidade de armazenamento, em alguns casos, não passa de MBs ou poucos GBs. Outro exemplo é a transmissão de arquivos através de aplicações para *smartphones*, na qual o custo da utilização da rede de dados móveis torna proibitiva a transmissão de muitos ou grandes arquivos.

Dados a serem comprimidos, geralmente, são produzidos por uma fonte aleatória de dados $\mathcal{X}$. Cada possível valor produzido por essa fonte pode ser representado como uma variável aleatória x_k, $k = 1, \ldots, K$, de um processo aleatório, de modo que

$$\mathcal{X} = \{x_1, x_2, \ldots, x_K\}. \tag{2.1}$$

Associada a cada elemento x_k existe uma probabilidade p_k, que representa a chance de x_k ocorrer quando um valor é selecionado aleatoriamente da fonte de dados. Baseado nas probabilidades p_k, Shannon introduziu a entropia de informação H [8, 9] para o caso onde o processo aleatório é discreto em amplitude, dada por

$$H = -\sum_{k=1}^{K} p_n \log p_k, \tag{2.2}$$

e para o processo aleatório contínuo

$$H = -\int_{-\infty}^{\infty} p(x) \log p(x) \, \mathrm{d}x, \tag{2.3}$$

onde $p(x)$ é a função densidade de probabilidade do processo contínuo. A entropia de uma fonte aleatória representa o valor esperado da informação contida em um valor gerado por $\mathcal{X}$. Em sistemas digitais, dados são representados, geralmente, usando códigos binários. Para representar K valores de um processo $\mathcal{X}$, cada valor x_k pode ser codificado como uma palavra binária de comprimento $L = \lceil \log_2 K \rceil$ bits. A codificação de dados em que todos os valores são codificados com o mesmo comprimento L é chamada codificação de taxa fixa. Em taxa fixa, se p_k é igual para todos os valores codificados, a entropia da fonte de dados é igual ao comprimento L.

A compressão de dados pode ocorrer sem perdas (*lossless*) ou com perdas (*lossy*) de informação. No caso sem perdas, os dados após a compressão são idênticos aos dados antes da compressão. Este tipo de compressão explora redundâncias estatísticas dos dados e é encontrado em sistemas de compressão sob a forma de codificação de taxa variável ou codificação *run-length*. Codificação de taxa variável pode ser utilizada quando os valores de p_k variam com k. Algoritmos como a codificação de Huffman [10] utilizam as diferentes probabilidades para determinar diferentes comprimentos L_k para cada valor da fonte. Valores mais prováveis são codificados com menores comprimentos. O comprimento médio $\bar{L}$, definido como a média aritimética de todos os comprimentos L_k dos valores da fonte, é menor do que o comprimento L na codificação em taxa fixa. Teoricamente, o comprimento médio $\bar{L}$ está limitado inferiormente pela entropia H da fonte de dados [8]. A codificação *run-length* tem aplicação quando se deseja transmitir uma sequência de valores em que há repetição. Por exemplo, nos valores dos pixels de uma imagem sintética, o plano de fundo pode conter uma sequência em que 115 pixels apresentem o valor 0,5. Transmitir uma vez o par (115,0.5) pode ser mais eficiente que transmitir 115 vezes o valor 0,5.

O caso da compressão de dados com perdas é o foco deste trabalho. Parte da informação contida em uma fonte de dados pode ser mais importante, em casos específicos, do que o resto da informação. A compressão de dados é aplicada, muitas vezes, a dados utilizados diretamente por seres humanos, como áudio, imagem e

(a) (b)

Figura 2.1: (a) Lena em formato TIFF, sem perdas; (b) Lena em formato JPEG, com perdas.

vídeo. O corpo humano possui limitações que impedem que alguns níveis de detalhes sejam percebidos por pessoas que não tenham um treinamento especial para aquela tarefa. Descartando estes níveis de detalhes, é possível comprimir a informação contida nos dados sem que a perda seja perceptível ou prejudicial para o usuário. A Figura 2.1 mostra um exemplo de imagem que passou por compressão sem perdas e com perdas. A imagem sem perdas tem 768 KB e a imagem com perdas tem 31,5 KB. A eficiência de um sistema com perdas está associada à relação entre o fator de redução de informação e os efeitos que a compressão causa aos dados.

Uma forma de definir matematicamente a perda de dados na compressão com perdas é através da distorção D definida como o erro médio quadrático entre N dados originais e comprimidos, dado por

$$D = \frac{1}{N} \sum_{n=1}^{N} (x_n - \hat{x}_n)^2, \tag{2.4}$$

onde $\hat{x}_n$ é o valor do dado x_n após a compressão. A distorção pode ser interpretada como a distância quadrática média entre os dados originais e os dados comprimidos. Limites teóricos para a compressão de dados em sistemas com perdas também foram estudados por Shannon em [11].

2.1 Quantização

A quantização é um estágio de um sistema de compressão de dados que o torna com perdas. Matematicamente, pode-se representar a quantização como um mapeamento $Q : \mathcal{X} \to \mathcal{Y}$, onde $\mathcal{Y}$ é um conjunto discreto de tamanho K composto por valores y_k, denominado dicionário (*codebook*), dado por

$$\mathcal{Y} = \{y_1, y_2, \ldots, y_K\}, \tag{2.5}$$

de modo que $K < N$, sendo N o tamanho de $\mathcal{X}$, possivelmente infinito. O objetivo da quantização é reduzir a quantidade de dados que serão processados, resultando em uma quantidade reduzida de dados que representam de forma satisfatória os dados originais. A quantização mapeia um intervalo de valores de $\mathcal{X}$ em um único valor de $\mathcal{Y}$. Por ser capaz de mapear valores contínuos em discretos, a quantização é facilmente encontrada em conversores analógico/digital. Quando o valor de entrada x_n é um escalar, isto é, tem apenas uma dimensão, a quantização deste dado é denominada quantização escalar (*SQ*, de *scalar quantization*). Um dispositivo ou algoritmo que realiza quantização é denominado quantizador.

Um quantizador é composto por dois estágios básicos [12]. O primeiro estágio é o bloco codificador (*encoder*), que terá sua função representada por α. O segundo estágio é o decodificador (*decoder*), a ser referido como β. O bloco codificador é responsável por determinar os limiares que definem os intervalos nos quais $\mathcal{X}$ será dividido. O conjunto destes limiares é denominado partição do quantizador. O codificador recebe um valor de entrada x_n e mapeia este valor em um índice binário $i_n = \alpha(x_n)$. A cada intervalo de $\mathcal{X}$ definido pela partição, está associado um índice i_n, de modo que todos os valores x_n pertencentes a um dado intervalo serão mapeados no mesmo índice i_n. No caso de quantização para transmissão de dados, este índice é transmitido no lugar do valor original. O decodificador β recebe o índice i_n como entrada. Este índice representa a posição de um elemento no dicionário $\mathcal{Y}$. O decodificador usa i_n para selecionar este elemento que será a saída do quantizador para uma entrada x_n. A saída do quantizador, então, é um dado comprimido $\hat{x}_n = y_n = \beta(\alpha(x_n))$, em que y_n é um dos valores y_k que foi associado ao valor de entrada x_n. Os valores y_k do dicionário são denominados pontos de

reconstrução do quantizador. Estes pontos podem ser calculados, durante o projeto do quantizador, como centros de massa dos intervalos a que estão associados. Neste caso, os pontos de reconstrução são chamados centroides do quantizador. A Figura 2.2 mostra uma representação básica dos blocos de um quantizador.

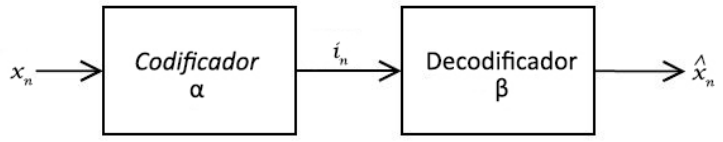


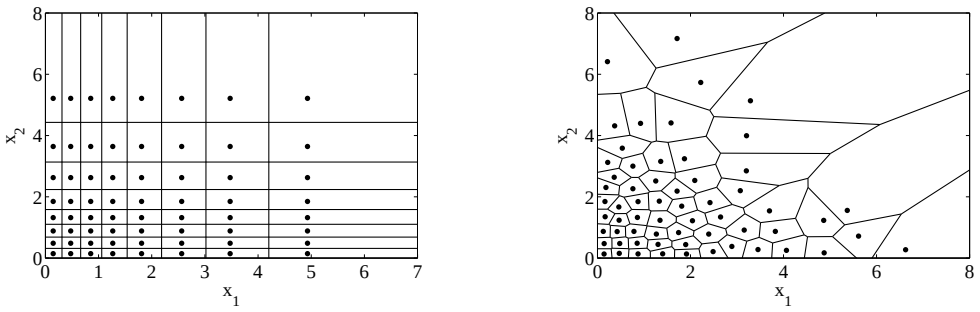
Figura 2.2: Blocos do quantizador: Codificador e Decodificador.

A perda de dados gerada pelo quantizador pode ser calculada a partir da Equação (2.4), com $\hat{x}_n$ associado a um y_k do dicionário. Quanto maior for K, isto é, quanto maior for o dicionário, menor pode ser a distorção D resultante. Mais valores y_k representam mais limiares na partição e menores intervalos nos dados de entrada. Com isso, os valores x_n da entrada se aproximam mais do valor $\hat{x}_n$, reduzindo as perdas. Um dicionário de tamanho $K = N$ poderia resultar em $D = 0$, com $y_n = x_n \forall n$. Aumentar K tende a elevar a entropia dos dados quantizados, uma vez que é necessário codificar uma maior quantidade de valores. Durante o projeto do quantizador, deve-se procurar um *trade-off* entre a distorção e entropia para a aplicação desejada.

2.1.1 Quantização Vetorial

Dados a serem quantizados, muitas vezes, não são apenas valores escalares. Sistemas de compressão podem processar dados com uma dimensão M maior que a unidade. Dados multidimensionais podem ser quantizados com quantizadores escalares, realizando a quantização de cada componente de forma independente. A quantização conjunta de todas as componentes atinge melhor desempenho em termos de taxa e distorção. Além disso, em muitos casos, as múltiplas dimensões de um dado possuem correlação entre si. Fazer a quantização dos dados com quantizadores escalares desperdiça esta propriedade e proporciona quantização de baixa eficiência. Realizando uma quantização conjunta de todas as dimensões do dado

de entrada, é possível gerar partição com dependência de todas as dimensões. Este tipo de quantização é denominado quantização vetorial (VQ, de *vector quantization*, ou *vector quantizer*). Utilizar quantizadores escalares gera partição separável para cada dimensão, independente das outras dimensões. A Figura 2.3 exemplifica as diferenças entre as partições e centroides nos dois casos, para vetores, não mostrados na figura, sorteados a partir de uma distribuição conjuntamente exponencial com $M = 2$.



(a) (b)

Figura 2.3: Partição e centroides para quantização de dados bidimensionais: (a) Quantizador escalar; (b) Quantizador vetorial.

O conjunto de dados de entrada de um quantizador vetorial será tratado como uma matriz $\mathbf{X} \in \mathbb{R}^{M \times N}$. Esta matriz é denominada matriz de dados e é composta por N vetores coluna $\mathbf{x}_n$, $n = 1, \ldots, N$ com M dimensões cada. Estes vetores são os dados de entrada do quantizador vetorial. O dicionário do quantizador será $\mathbf{Y} \in \mathbb{R}^{M \times K}$, composto pelos vetores de reconstrução $\mathbf{y}_k$, $k = 1, \ldots, K$.

A VQ é capaz de alcançar desempenho igual ou superior, em termos de distorção, a qualquer outra forma de quantização. Para isso, basta que seu dicionário $\mathbf{Y}$ seja definido com valores iguais aos pontos de reconstrução gerados por esta outra forma de quantização e que o codificador α implemente a partição ótima para estes pontos de reconstrução. Isso faz com que o VQ seja uma técnica de compressão com fácil aplicação. O tamanho do dicionário de um VQ influencia diretamente no custo computacional para a busca do índice i_n. Aplicações que requerem dicionários relativamente grandes podem tornar o VQ inviável, como técnicas de codificação de voz [13], que podem utilizar dicionários com milhares de pontos de reconstrução, por exemplo. Valores elevados para K tornam a busca de i_n uma tarefa de difícil execução devido ao alto custo computacional, uma vez que processadores de sinais possuem limitações tecnológicas.

2.1.2 Quantização Vetorial com Restrição de Entropia

Um sistema de compressão de dados pode combinar diferentes estágios de compressão, utilizando tanto técnicas com perdas como sem perdas ao mesmo tempo. Uma codificação de Huffman pode ser aplicada aos índices i_n gerados pelo codificador α. É desejado que a entropia do dicionário $\mathbf{Y}$ seja a menor possível, uma vez que a capacidade de compressão da codificação sem perdas está limitada por $H(\mathbf{Y})$. O projeto de um VQ leva em consideração partição e centroides que reduzem apenas a distorção D de quantização. Geralmente, sistemas otimizados apenas para reduzir D resultam em entropia alta. A idéia da quantização vetorial com restrição de entropia (ECVQ, de *entropy-constrained vector quantization* [14], ou *entropy-constrained vector quantizer*) é adicionar o fator H na etapa de otimização do projeto do quantizador. Desta forma, quantizadores vetoriais projetados tendem a reduzir a distorção dado um limite para a entropia associada.

2.2 Redes Neurais

Redes neurais artificiais (RNAs) são modelos computacionais que buscam soluções de problemas através de sistemas estruturais inspirados pelo funcionamento do cérebro humano [15], com aplicações em diversos sistemas, como reconhecimento de padrões, *clustering* e compressão de dados. A idéia de uma rede neural artificial é implementar a capacidade de aprendizado existente em redes neurais naturais.

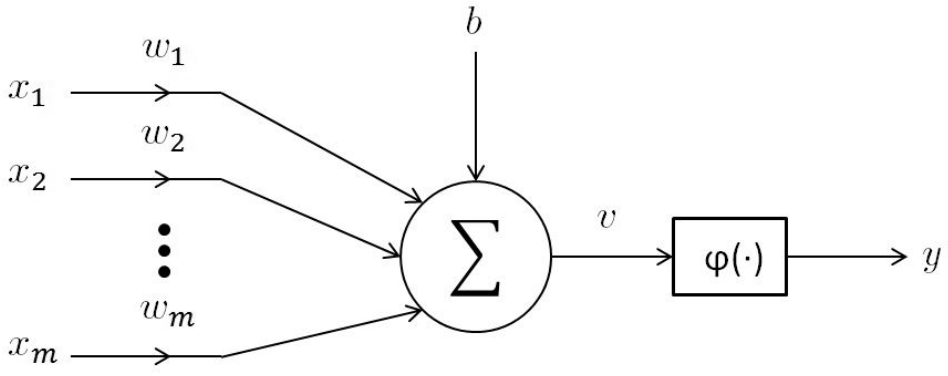


Figura 2.4: Modelo de um neurônio de redes neurais artificiais.

O elemento básico de uma RNA, assim como de um sistema nervoso natural, é chamado neurônio. Um neuronio, em RNAs, é representado por um sistema que

recebe m entradas $x_1, \ldots, x_m$ multiplicadas por seus respectivos pesos $w_1, \ldots, w_m$. Além das entradas, um sinal de bias b é adicionado ao somatório das entradas multiplicadas pelos pesos do neurônio. O núcleo do neurônio realiza o somatório dos valores na entrada, multiplicados pelos respectivos pesos, e do bias, gerando o sinal

$$v = \sum_{j=1}^{m} w_j x_j + b. \tag{2.6}$$

Se as entradas forem interpretadas como um vetor $\mathbf{x} = [x_1, \ldots, x_m]$ e os pesos como $\mathbf{w} = [w_1, \ldots, w_m]$, a Equação (2.6) pode ser escrita como

$$v = \mathbf{w}^T \mathbf{x} + b. \tag{2.7}$$

Após a combinação linear feita no núcleo, o sinal v passa por uma função de ativação $\varphi(\cdot)$. A função de ativação é responsável por adicionar não-linearidade ao modelo. Em geral, é dada por uma função sigmoide, isto é, uma função cuja curva possui o formato de uma letra "S". Em modelos mais simples de neurônios, $\varphi(\cdot)$ pode ser apenas um comparador. Em muitos casos, a função de ativação é uma função logística, de modo que

$$\varphi(v) = \frac{1}{1 + \exp(-av)}, \quad a > 0, \tag{2.8}$$

ou uma função tangente hiperbólica, dada por

$$\varphi(v) = a\tanh(bv), \quad a, b > 0. \tag{2.9}$$

2.2.1 Perceptron

Em 1957, Frank Rosenblatt introduziu o conceito do perceptron, que consiste numa camada de neurônios cujas funções de ativações são comparadores simples, dados pelo modelo de neurônio McCulloch-Pitts. Este modelo é como o apresentado na Figura 2.4, usando um comparador como função de ativação. A função inicial do perceptron era atribuir um vetor de entrada $\mathbf{x} = [x_1, \ldots, x_m]$ a uma entre duas classes $\mathcal{C}_1$ ou $\mathcal{C}_2$. Este perceptron contém um neurônio com um vetor de pesos

$\mathbf{w} = [w_1, \ldots, w_m]$ e um bias b. Através de métodos de treinamento iterativos, os valores dos pesos $w_1, \ldots, w_m$ do neurônio são adaptados de forma a implementar um hiperplano de $(m-1)$ dimensões que define o limite de decisão entre as duas classes $\mathcal{C}_1$ e $\mathcal{C}_2$ [16].

2.2.2 Perceptron Multicamadas

O perceptron de camada única proposto por Rosenblatt possui limitações quando os problemas envolvem dados que não são linearmente separáveis, uma vez que o modelo é capaz de gerar apenas um hiperplano para o limiar das classes. O uso de perceptrons multicamadas (MLP, de *Multilayer Perceptron*) é uma solução para este tipo de problema. MLPs são compostos por uma ou mais camadas de neurônios denominadas camadas ocultas ou camadas escondidas. Além disso, os resultados das camadas ocultas servem como entrada para uma camada de saída. A Figura 2.5 mostra um MLP com duas camadas ocultas e uma camada de saída com três neurônios, resultando em três saídas para o modelo.

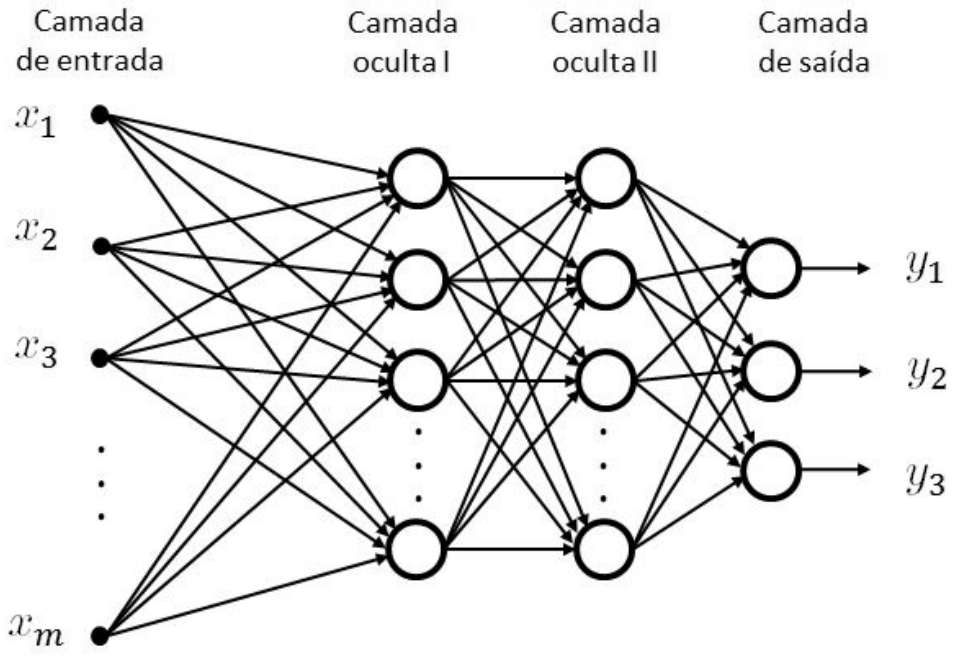


Figura 2.5: Modelo de um MLP de duas camadas ocultas.

2.2.3 Mapas Auto-Organizáveis de Kohonen

Mapas auto-organizáveis (SOM, de *Self-Organizing Maps*), ou mapas de Kohonen, são uma classe de redes neurais capaz de mapear dados de entrada com uma quantidade elevada de dimensões em um grupo de dados com dimensões reduzidas, preservando as partes mais essenciais da informação contida nos dados. O treinamento destes mapas é não-supervisionado, ou seja, não depende de controle externo, além dos próprios dados de entrada, para atualizar os pesos da rede. Estas redes usam aprendizagem por competição, de modo que os neurônios de saída competem entre si, de maneira que apenas um tem sinal de ativação por vez.

O processo de aprendizagem por competição busca o neurônio de saída cujo vetor de pesos $\mathbf{w}$ mais se aproxima do vetor de entrada $\mathbf{x}$. Seja o vetor de pesos de cada neurônio $\mathbf{w}_j$, sendo $j = 1, \ldots, l$ a posição do neurônio entre os l totais neurônios da rede. Um neurônio k é ativado se seu produto interno $\mathbf{w}_k^T\mathbf{x} - ||\mathbf{w}_k||^2 > \mathbf{w}_j^T\mathbf{x} - ||\mathbf{w}_j||^2, \forall j \neq k$. As normas são descontadas considerando o caso em que vetores $\mathbf{w}_k$ possuem normas diferentes. Este processo é denominado *winner-takes-all*, no qual o neurônio "vencedor" é o único ativado. Matematicamente, a busca pelo maior produto interno $\mathbf{w}_k^T\mathbf{x} - ||\mathbf{w}_k||^2$ é equivalente a buscar o vetor $\mathbf{w}$ que mais reduz a distância Euclideana para o vetor $\mathbf{x}$. Dessa forma, uma das principais propriedades de um SOM é retornar, através de um vetor $\mathbf{w}_k$, uma aproximação do espaço de entrada. Esta propriedade mostra que esta classe de redes neurais pode desempenhar o mesmo papel de um quantizador vetorial e provê uma conexão entre mapas de Kohonen e compressão de dados.

Capítulo 3

Quantização Vetorial Baseada em Kernel PCA

Quando um sistema de compressão aplica métodos que geram perdas aos dados, é importante que estas perdas estejam concentradas em partes menos significativas destes dados. Uma forma comum utilizada para reduzir o efeito das perdas é a extração de *features* dos dados. *Features* são representações destes dados que concentram a informação, de modo que seja mais fácil processar a informação de forma seletiva. Transformadas podem ser utilizadas antes de métodos de compressão para extrair *features* e tornar o sistema mais eficiente em termos da distorção gerada.

3.1 Análise de Componentes Principais

A correlação existente entre componentes de um conjunto de vetores com M dimensões foi utilizada como motivação para o uso de quantização vetorial em vez da quantização escalar. A ideia da Análise de Componentes Principais (PCA) é aplicar uma transformada sobre os vetores de modo a descorrelacionar a informação entre suas múltiplas componentes e concentrar a energia dos dados em um número menor de coeficientes. Desta forma, dados descorrelacionados podem ser processados de forma eficiente em termos de distorção e com técnicas mais simples.

Para um conjunto de dados $\mathbf{X} \in \mathbb{R}^{M \times N}$, normalizado de modo que cada linha, associada a cada componente dos vetores, tenha média zero e variância igual a um, a PCA é implementada como uma matriz cujas colunas são os autovetores da matriz

de covariâncias dos vetores de $\mathbf{X}$. A matriz de covariâncias $\mathbf{C_X}$ é dada por

$$\mathbf{C_X} = \frac{\mathbf{X}\mathbf{X}^T}{N}. \tag{3.1}$$

Os autovetores de $\mathbf{C_X}$ são denominados componentes principais da distribuição de dados e indicam as direções onde estão concentradas as partes mais significativas da informação dos dados. Nestas direções, os dados possuem maior variância, ou seja, maior energia. A Figura 3.1 exemplifica um conjunto de dados com duas dimensões e $\mathbf{v}_1$ e $\mathbf{v}_2$ indicam os autovetores de sua matriz de covariâncias.

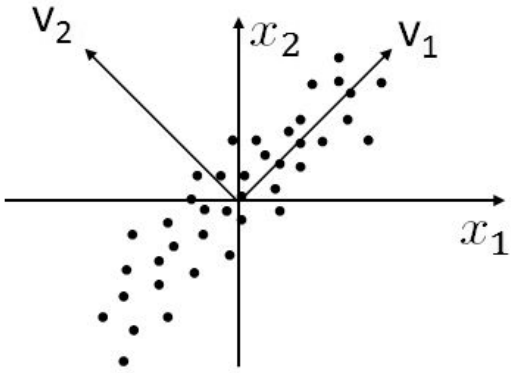


Figura 3.1: Exemplo de um conjunto de dados e indicadores dos seus autovetores.

Os autovetores de $\mathbf{C_X}$ são organizados em ordem decrescente de energia, definindo as colunas da matriz de transformada $\mathbf{W}$. Na Figura 3.1, $\mathbf{v}_1$ aponta na direção de maior variância dos dados. Neste exemplo, $\mathbf{v}_1$ apresenta mais energia que $\mathbf{v}_2$. A transformada é implementada como a projeção dos dados de entrada sobre as componentes principais, através da equação

$$\mathbf{F} = \mathbf{W}^T\mathbf{X}. \tag{3.2}$$

Aplicando a Equação (3.2), os vetores da matriz $\mathbf{X}$ são mapeados para o espaço $\mathcal{F}$, chamado espaço de *features*. As colunas de $\mathbf{F}$ pertencem a este espaço. O resultado da projeção é mostrado na Figura 3.2.

Nas novas coordenadas dos vetores após a projeção, existe concentração de energia na primeira componente, correspondente às projeções sobre o autovetor associado

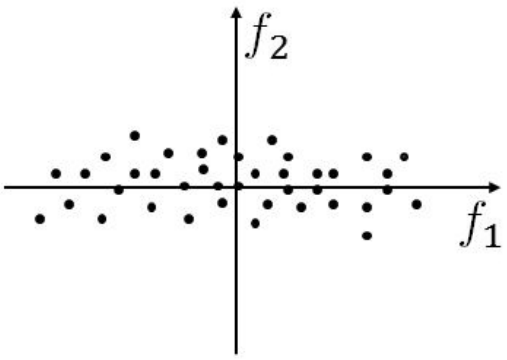


Figura 3.2: Exemplo de resultado da análise de componentes principais.

à direção de maior variância. Desta forma, técnicas de compressão podem explorar esta concentração de energia para tornar a compressão de dados mais eficiente.

3.2 Kernel PCA

O método de quantização proposto neste trabalho faz o uso de Kernel PCA (KPCA), que começa com um mapeamento não-linear $\Phi : \mathbb{R}^M \rightarrow \mathbb{R}^I$, que leva os vetores originais, com M dimensões, para um novo espaço, em que I é uma quantidade grande, possivelmente infinita, de dimensões. O vetor $\mathbf{q}_n$ é o resultado do mapeamento de um vetor de entrada $\mathbf{x}_n$ para o $\mathbb{R}^I$ e a matriz $\mathbf{Q} = [\mathbf{q}_1\, \mathbf{q}_2\, \ldots\, \mathbf{q}_N]$ é o resultado do mapeamento completo da matriz de dados $\mathbf{X}$ [17]. Os autovetores da matriz de covariâncias associada aos vetores após o mapeamento são computados. A nova matriz de covariâncias $\mathbf{C}'_{\mathbf{X}} \in \mathbb{R}^{I \times I}$ é dada por

$$\mathbf{C}'_{\mathbf{X}} = \frac{\mathbf{Q}\mathbf{Q}^T}{N}. \tag{3.3}$$

Computar estes autovetores a partir da equação de autovalores e autovetores

$$\mathbf{C}'_{\mathbf{X}}\mathbf{v}_j = \rho_j \mathbf{v}_j \tag{3.4}$$

é uma tarefa computacionalmente difícil ou impossível de ser executada, uma vez que $\mathbf{C}'_{\mathbf{X}}$ possui dimensão muito elevada. O desenvolvimento completo da Kernel

PCA é dado em [17], mas, por conveniência, uma descrição sucinta deste método é dada a seguir. Os autovetores de $\mathbf{C}'_{\mathbf{X}}$ podem ser descritos como uma combinação linear das colunas da matriz $\mathbf{Q}$, de modo que

$$\mathbf{v}_j = \sum_i a_{ij}\mathbf{q}_i = \mathbf{Q}\mathbf{a}_j. \tag{3.5}$$

Substituindo as Equações (3.3) e (3.5) na Equação (3.4) e multiplicando os dois lados do resultado por $\mathbf{Q}^T$, chega-se a

$$\mathbf{Q}^T\frac{\mathbf{Q}\mathbf{Q}^T}{N}\mathbf{Q}\mathbf{a}_j = \rho_j\mathbf{Q}^T\mathbf{Q}\mathbf{a}_j. \tag{3.6}$$

Nos dois lados da Equação (3.6) o produto $\mathbf{Q}^T\mathbf{Q}$ pode ser substituído por uma nova matriz $\mathbf{K} = \mathbf{Q}^T\mathbf{Q}$. A equação pode ser escrita como

$$\mathbf{K}^2\mathbf{a}_j = N\rho_j\mathbf{K}\mathbf{a}_j \tag{3.7}$$

e simplificada [17] para

$$\mathbf{K}\mathbf{a}_j = N\rho_j\mathbf{a}_j. \tag{3.8}$$

É necessário computar a matriz $\mathbf{K}$ para que seus autovetores $\mathbf{a}_j$ possam ser calculados. Cada elemento k_{ij} da matriz $\mathbf{K}$ consiste em um produto interno da forma $(\mathbf{q}_i \cdot \mathbf{q}_j)$. Como os vetores $\mathbf{q}$ são vetores em $\mathbb{R}^I$, com dimensionalidade elevada, computar estes produtos internos é computacionalmente proibitivo e, portanto, não se pode calcular a matriz $\mathbf{K}$ através do cálculo direto de $(\mathbf{q}_i \cdot \mathbf{q}_j)$. O cálculo da matriz $\mathbf{K}$ depende apenas dos resultados dos produtos internos. Os valores dos vetores mapeados em $\mathbb{R}^I$ não são relevantes diretamente para montar a matriz $\mathbf{K}$. O uso de funções kernel fornece uma forma eficiente de calcular os elementos k_{ij} sem que os vetores $\mathbf{q}$ precisem ser conhecidos. Uma função kernel retorna o resultado do produto interno no espaço $\mathbb{R}^I$ usando apenas os vetores do espaço original $\mathbb{R}^M$, como

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = (\mathbf{q}_i \cdot \mathbf{q}_j). \tag{3.9}$$

Funções kernel são utilizadas em diversas aplicações [18, 19] por serem capazes

de prover um caminho relativamente simples entre espaços de diferentes dimensões. Muitas funções podem ser aplicadas como funções kernel. Este trabalho se concentra em três funções, apresentadas na Tabela 3.1.

Tabela 3.1: Funções kernel.

Nome	Função kernel
Polinomial	$k(\mathbf{x}, \mathbf{y}) = (\alpha \mathbf{x}^T \mathbf{y} + c)^d$
Radial-basis-function (RBF)	$k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{\|\mathbf{x}-\mathbf{y}\|^2}{\gamma}\right)$
Tangente hiperbólica	$k(\mathbf{x}, \mathbf{y}) = \tanh(\alpha \mathbf{x}^T \mathbf{y} + c)$

A matriz $\mathbf{Q}$ tem dimensão $I \times N$, de modo que a matriz $\mathbf{K}$ tem dimensões $N \times N$. N é, geralmente, um valor alto. Por exemplo, utilizando vetores descritores de textura extraídos de blocos de 4×4 pixels de uma única imagem com 480×640 pixels, são obtidos 19200 vetores, um por bloco. Normalmente, várias imagens podem ser utilizadas para gerar uma matriz $\mathbf{X}$. Desta forma, calcular os autovetores da matriz $\mathbf{K}$ ainda é uma tarefa inviável, pois o cálculo de N autovetores está sujeito a erros numéricos e cada autovetor seria representado por N valores imprecisos. Uma solução para este problema é selecionar um subconjunto com N' vetores, sendo que $N' \ll N$, de modo que a matriz $\mathbf{K}$ gerada a partir destes vetores tem $N' \times N'$ dimensões, possibilitando o cálculo dos seus autovetores $\mathbf{a}_j$. A matriz $\mathbf{K}$ tem posto limitado superiormente pelo tamanho da matriz de dados N' e, no máximo, N' autovalores maiores que zero. A quantidade de autovalores maiores que zero é igual à quantidade de autovetores relevantes. Como a PCA possui a propriedade de concentrar grande parte da energia dos dados em poucos coeficientes, é possível aplicar redução de dimensionalidade, descartando alguns autovetores correspondentes a direções associadas a variâncias menores. Desta forma, apenas $M' < N'$ autovetores são mantidos. Ao contrário do que acontece com a PCA convencional, na KPCA pode acontecer $M' > M$. Estes autovetores calculados são autovetores da matriz $\mathbf{K}$. Os autovetores associados à matriz de covariâncias $\mathbf{C}'_{\mathbf{X}}$ são os autovetores $\mathbf{v}_j$, relacionados com $\mathbf{a}_j$ pela Equação (3.5).

Precisamos calcular o produto interno $(\mathbf{v}_j \cdot \Phi(\mathbf{x}_t))$ para computar a projeção em $\mathcal{F}$ de um vetor de entrada $\mathbf{x}_t$ mapeado para o $\mathbb{R}^I$, $\Phi(\mathbf{x}_t)$, sobre um autovetor $\mathbf{v}_j$. Utilizando a versão transposta da Equação (3.5), temos

$$f_{t,j} = (\mathbf{v}_j \cdot \Phi(\mathbf{x}_t)) = \mathbf{a}_j^T \mathbf{Q}^T \mathbf{q}_t. \tag{3.10}$$

A matriz $\mathbf{Q}$ resulta do mapeamento dos N' vetores da matriz de dados. Estes vetores serão referidos como os vetores $\mathbf{l}_1, \ldots, \mathbf{l}_{N'}$. O vetor $\mathbf{q}_t$ resulta do mapeamento de $\mathbf{x}_t$. Para computar $\mathbf{Q}^T\mathbf{q}_t$, devemos utilizar novamente as funções kernel, de modo que um novo vetor $\mathbf{z}_t$ é definido a partir dos N' produtos internos:

$$\mathbf{z}_t = \begin{bmatrix} k(\mathbf{l}_1, \mathbf{x}_t) \\ \vdots \\ k(\mathbf{l}_{N'}, \mathbf{x}_t) \end{bmatrix}. \tag{3.11}$$

Geralmente, dados de entrada são compostos por muitos vetores, em vez de apenas um vetor $\mathbf{x}_t$. Para fins de implementação, consideramos que o número de vetores de entrada é igual a L. Desta forma, aplicando a Equação (3.11) a todos os vetores, obtemos uma matriz composta por L vetores $\mathbf{z}_t$ como colunas, dada por $\mathbf{Z} \in \mathbb{R}^{N' \times L}$.

Para realizar a extração de *features* dos dados, é desejado que os vetores de entrada em suas versões mapeadas por Φ estejam centralizados em torno de zero, isto é, $\sum_{n=1}^{N} \Phi(\mathbf{x}_t(n)) = \mathbf{0}$. A mesma condição se aplica à matriz que gera os autovetores, no caso $\mathbf{K}$, de modo que $\sum_{n=1}^{N'} \Phi(\mathbf{l}(n)) = \mathbf{0}$ também é necessário. Realizar centralização em torno de zero no espaço original $\mathbb{R}^M$ é relativamente simples, mas para garantir que ela ocorra em $\mathcal{F}$, definimos $\mathbf{K}_c$ como a versão centralizada [17] de $\mathbf{K}$ como

$$\mathbf{K}_c = \mathbf{K} - \mathbf{U}_1\mathbf{K} - \mathbf{K}\mathbf{U}_1 + \mathbf{U}_1\mathbf{K}\mathbf{U}_1 \tag{3.12}$$

e a versão centralizada de $\mathbf{Z}$ como

$$\mathbf{Z}_c = \mathbf{Z} - \mathbf{K}\mathbf{U}_2 - \mathbf{U}_1\mathbf{Z} + \mathbf{U}_1\mathbf{K}\mathbf{U}_2, \tag{3.13}$$

onde $\mathbf{U}_1 \in \mathbb{R}^{N' \times N'}$ e $\mathbf{U}_2 \in \mathbb{R}^{N' \times L}$ com todos elementos iguais a $1/N'$. Após as centralizações em torno de zero, as projeções de um único vetor de entrada $\mathbf{x}_t$ ao longo dos autovetores de $\mathbf{K}_c$ são calculadas a partir da Equação (3.10) resultando em um vetor de projeções $\mathbf{f}_t$ dado por

$$\mathbf{f}_t = \mathbf{A}^T \mathbf{z}_t, \tag{3.14}$$

onde $\mathbf{A}$ é uma matriz $N' \times M'$ composta pela concatenação dos autovetores $\mathbf{a}_j$ de $\mathbf{K}_c$ em colunas. Para L vetores de entrada, após a centralização em torno de zero, a equação pode ser escrita como

$$\mathbf{F} = \mathbf{A}^T \mathbf{Z}_c, \tag{3.15}$$

onde a matriz $\mathbf{F}$ é $M' \times L$ formada por um vetor de M' *features* calculado para cada um dos L vetores de entrada $\mathbf{x}_t$. A Equação (3.15) pode ser reescrita, combinando as Equações (3.13) e (3.15), de modo que a centralização em torno de zero fique implícita e a matriz $\mathbf{F}$ possa ser calculada diretamente através de $\mathbf{Z}$. Isto resulta na equação em forma de rede neural, com saída linear, e com uma camada escondida (que tem saídas $\mathbf{z}_t$ e com uma segunda camada que tem uma matriz de pesos $\mathbf{W}$ e um vetor bias de $\mathbf{b}$:

$$\mathbf{F} = \mathbf{W}\mathbf{Z} + \mathbf{b}. \tag{3.16}$$

O modelo de implementação para a KPCA dado pela Equação (3.16) apresenta a tarefa de extração de *features* baseada em funções kernel como a tarefa de calcular as saídas de uma rede neural com uma camada escondida, com função de ativação dada pela função kernel. As saídas dos neurônios desta camada escondida fornecem os valores de $\mathbf{Z}$. Além disso, uma camada de saída linear executa a matriz de pesos e o vetor de bias dados pela Equação (3.16).

3.3 VQ baseada em KPCA

Uma vez implementada a KPCA sobre os dados, é possível utilizar os resultados para projetar um quantizador que explore as propriedades da KPCA. Ao selecionar *features* de maior energia em $\mathcal{F}$, é possível aplicar a quantização a estas *features* de modo que a distorção gerada pela quantização no espaço de entrada seja minimizada.

Para projetar um quantizador, é necessário projetar a partição no espaço das *features* e os pontos de reconstrução correspondentes no espaço de entrada $\mathbb{R}^M$. Para

gerar a partição explorando as propriedades da KPCA, aplicamos quantizadores escalares às *features*. Quantizadores escalares são escolhidos por sua baixa complexidade e facilidade de implementação. Cada *feature* é codificada por um SQ, criando uma partição separável no espaço das *features*. Esta partição separável em $\mathcal{F}$ gera uma partição não-separável no espaço de entrada, dado o mapeamento não-linear Φ. Em $\mathcal{F}$, a partição define células. Células consistem em conjuntos de vetores de *features* pertencentes a uma dada região M'-dimensional, que foi determinada pelos limiares da partição. Como cada vetor em $\mathcal{F}$ é resultado do mapeamento de um vetor de entrada, podemos usar os conjuntos em $\mathcal{F}$ para definir os respectivos conjuntos dos vetores de entrada. Esse processo define a partição no espaço de entrada. Cada ponto de reconstrução do quantizador vetorial baseado em kernel PCA (KPCA-VQ) é calculado como o centro de massa do conjunto associado, no espaço de entrada. O diagrama de blocos apresentado da Figura 3.3 representa, de forma simplificada, o projeto do KPCA-VQ. Mais detalhes sobre o projeto de KPCA-VQ realizado neste trabalho serão apresentados na Seção 4.3.

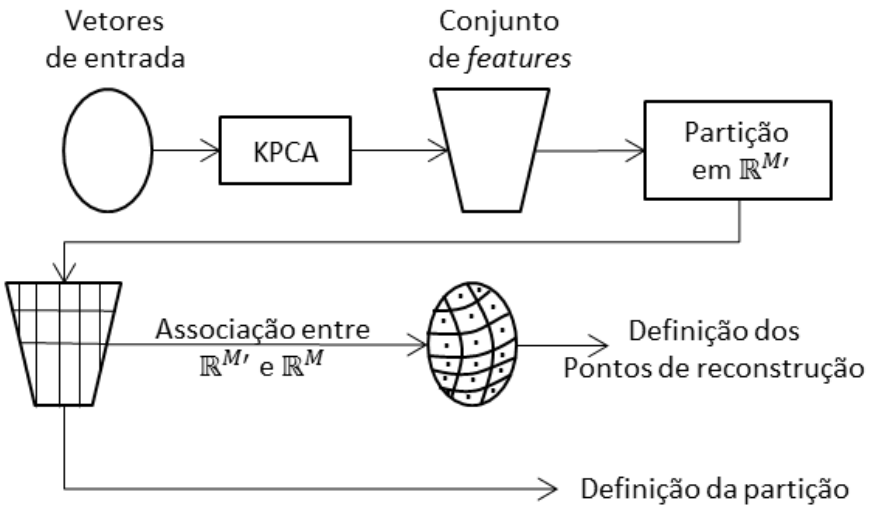


Figura 3.3: Diagrama de blocos do projeto de um KPCA-VQ. Um conjunto de *features* é calculado para um conjunto de vetores de entrada. Uma partição é definida no espaço das *featues*. Esta é a partição do KPCA-VQ. Uma partição correspondente pode ser obtida no espaço de entrada. Uma vez definida a partição no espaço de entrada, são calculados centros de massa das células geradas. Estes centros de massa são os pontos de reconstrução do KPCA-VQ.

Os pontos de reconstrução são armazenados em um dicionário. Cada posição deste dicionário está associada a um índice gerado a partir da partição no espaço das *features*. Para quantizar um vetor M-dimensional, é calculado o vetor de *features* $\mathbf{f}$ correspondente através da Equação (3.16) e cada elemento de $\mathbf{f}$ é comparado aos limiares dos SQs definidos no projeto do quantizador. Com os resultados das comparações, um índice é gerado. Este índice representa a posição do dicionário onde está o ponto de reconstrução do quantizador. Este processo é demonstrado na Figura 3.4.

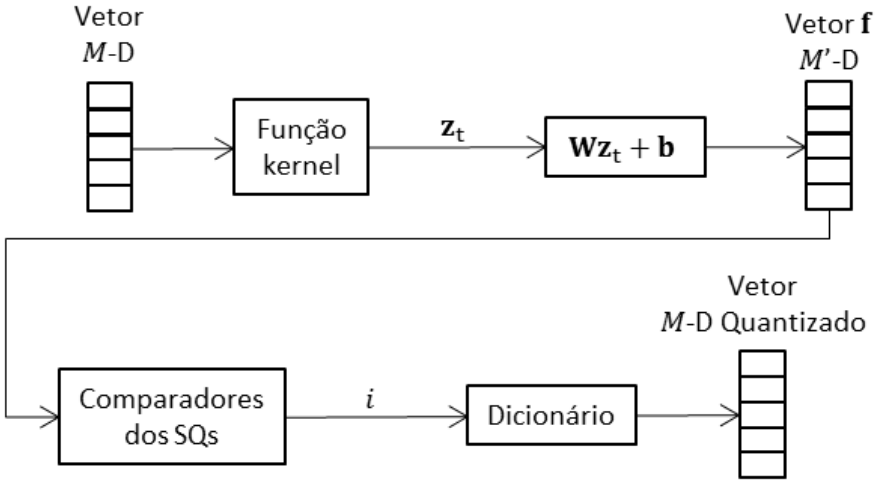


Figura 3.4: Diagrama de blocos da quantização de um vetor de entrada via KPCA-VQ.

Capítulo 4

Projetos e Implementações

Para este trabalho, quantizadores baseados em KPCA-VQ foram projetados e testados. As técnicas de quantização tradicionais apresentadas, SQ e ECVQ, também foram projetadas e testadas para a mesma base de dados, de modo a servir como parâmetros de comparação. A base de dados utilizada para projeto será tratada como $\mathbf{X}_{treino} \in \mathbb{R}^{M \times N_{treino}}$, composta por N_{treino} vetores coluna de treino M-dimensionais. Uma base de dados diferente é utilizada para teste. A forma como estas bases de dados são construídas é detalhada no Capítulo 5.

4.1 Projeto do SQ

Para projetar o SQ, utilizamos o algoritmo de Lloyd [20]. O algoritmo é inicializado com 2^{R_m} pontos 1-D aleatórios y_k distribuídos ao longo da componente m dos dados de treino, onde $m = 1, \ldots, M$ representa a m-ésima linha da base de dados de treino $\mathbf{X}_{treino}$. O valor R_m representa a taxa de bits alocada para o projeto do quantizador da m-ésima componente. Para cada componente, isoladamente, o algoritmo define $(2^{R_m} - 1)$ limiares que minimizam o erro médio quadrático em função dos valores y_k gerados na inicialização. Isto é feito agrupando cada valor de treino pertencente à componente m com o valor y_k mais próximo, gerando células. De modo equivalente, pode-se dizer que a condição de partição do algorimo de Lloyd minimiza uma função custo J_m dada por $J_m = D_m$, onde D_m é a distorção associada à m-ésima dimensão. Definidas as células, o algoritmo calcula novos valores para cada y_k como o centro de massa para cada célula, fazendo a média de todos os

valores de treino associados a cada uma. O algoritmo repete o processo, redefinindo partição e centroides, iterativamente. A cada iteração, o erro médio quadrático é reduzido. Cada quantizador converge para partição e dicionário ótimos quando as variações no erro médio quadrático forem desprezíveis. Uma vez que os M quantizadores escalares tenham sido projetados, é possível realizar a quantização dos dados M-dimensionais com SQs.

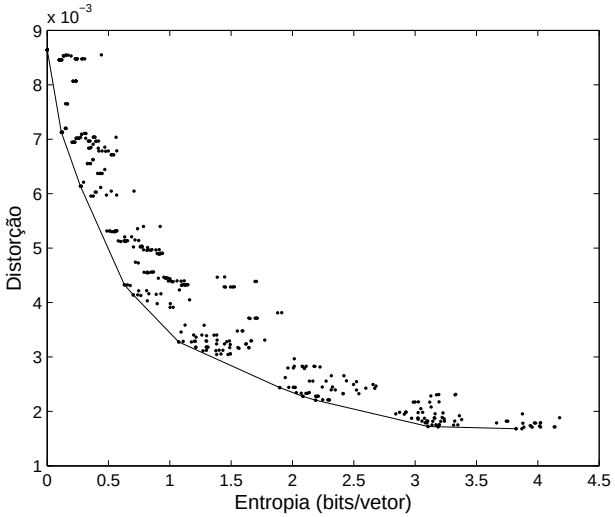


Figura 4.1: Nuvem de pares taxa-distorção e sua casca convexa inferior.

Neste trabalho, o algoritmo de Lloyd é modificado de modo a minimizar uma função custo $J_m = D_m + \lambda_m H_m$, onde H_m é a entropia associada ao uso das células criadas na m-ésima componente. O critério de parada para a otimização é a ocorrência de uma variação na função custo menor ou igual a $\epsilon = 10^{-7}$. O valor escalar λ_m é um multiplicador de Lagrange escolhido de forma a definir o peso da entropia H_m na função custo. O cálculo de λ_m é feito após o projeto de vários SQs para $\lambda_m = 0$. São realizados diferentes projetos de SQ, com inicializações diferentes e variando a quantidade de bits utilizada. Isto gera pares taxa-distorção que são espalhados em uma nuvem de pontos pelo plano H_m-D_m. A casca convexa inferior desta nuvem de pontos é extraída. Uma casca convexa inferior é uma curva formada pelos melhores pares taxa-distorção de uma nuvem de pares, de modo que nenhum outro par taxa-distorção esteja abaixo desta curva. Para cada par taxa-distorção da casca convexa, podemos aproximar um valor para a derivada naquele ponto fazendo a média entre as inclinações dos segmentos anterior e posterior àquele ponto. Este

valor é atribuído a λ_m para o projeto de um SQ com a mesma quantidade de bits do SQ que gerou o par taxa-distorção. A Figura 4.1 mostra uma nuvem de pares taxa-distorção e a casca convexa inferior correspondente.

O algoritmo de Lloyd é utilizado em conjunto com um algoritmo que busca a melhor alocação de bits entre os quantizadores de cada componente. Ou seja, o algoritmo busca o conjunto de valores R_m, $m = 1, \ldots, M$, que forneça o melhor desempenho taxa-distorção para um número de bits total desejado. Este algoritmo é apresentado na Figura 4.2.

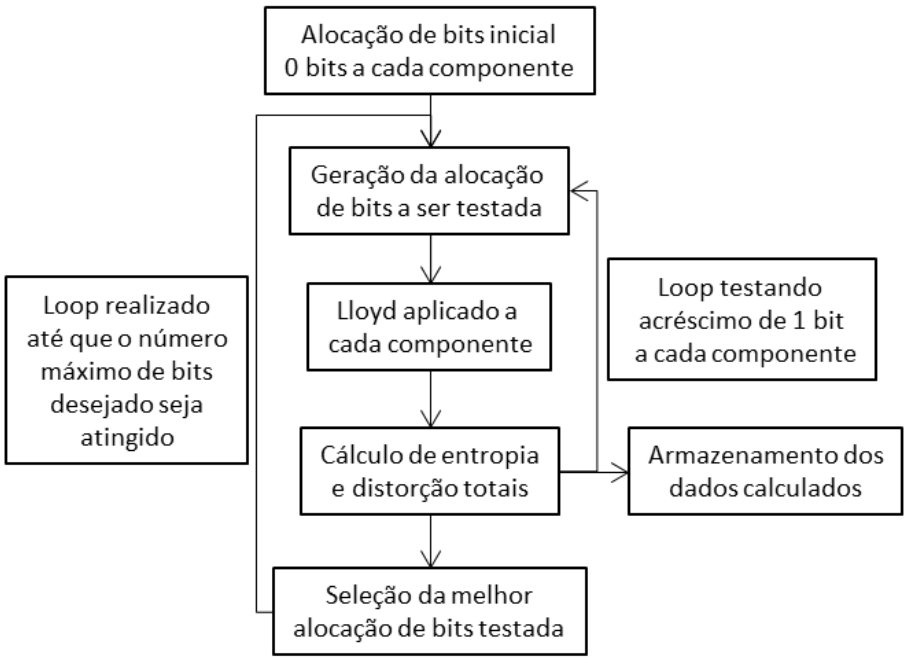


Figura 4.2: Diagrama de blocos do algoritmo de projeto do SQ. A alocação inicial gera um vetor base de M zeros, no qual cada elemento indica a quantidade de bits R_m alocada ao m-ésimo SQ. A cada iteração de um loop, um vetor de teste de alocação é gerado a partir do vetor base. O vetor de teste corresponde ao vetor base acrescido de uma unidade em cada componente diferente para cada iteração. Na primeira iteração, o vetor de teste é $[1\,0\,0\,\ldots]$. Na segunda, $[0\,1\,0\,\ldots]$, e assim por diante. SQs são projetados utilizando o algoritmo de Lloyd, com a quantidade de bits definida pelo vetor de teste. Entropia e distorção correspondentes a quantização dos vetores M-dimensionais são calculadas e armazenadas. Quando a adição de 1 bit é realizada e testada para todas as componentes, os resultados são comparados e a alocação que fornece o melhor resultado é selecionada como a nova alocação base. O loop externo acontece até que a soma de todos os valores R_m alcance a quantidade de bits desejada.

4.2 Projeto do ECVQ

O projeto do VQ usa a versão generalizada do algoritmo de Lloyd (GLA, de *Generalized Lloyd Algorithm*) [21]. O GLA aplica a otimização de partição e de centroides de forma iterativa, similar ao algoritmo de Lloyd para uma dimensão. A inicialização é feita com vetores M-dimensionais $\mathbf{y}_k$, em vez de valores independentes para cada dimensão. Células são definidas em função das mínimas distâncias Euclideanas dos vetores de treino $\mathbf{x}_{treino}$ para os vetores do dicionário $\mathbf{y}_k$. O cálculo dos novos centroides é feito em função da média dos vetores pertencentes a cada célula. Este algoritmo converge para um mínimo local quando o erro médio quadrático deixa de variar significativamente entre as iterações.

Para o ECVQ, a adaptação no algoritmo ocorre na condição utilizada para definir a partição. Para o VQ sem restrição de entropia, os vetores são selecionados para as células apenas em função da distância Euclideana para o $\mathbf{y}_k$ associado àquela célula. No ECVQ, a função custo minimizada no momento de definir as células é dada pela expressão Lagrangiana $J = D + \lambda H$. O critério de parada para o GLA é a ocorrência de uma variação de J entre iterações menor ou igual a $\epsilon = 10^{-7}$.

4.3 Projeto do KPCA-VQ

O KPCA-VQ tem sua partição implementada no espaço de *features* $\mathcal{F}$, enquanto a otimização de centroides é realizada no espaço de entradas. Ou seja, a partição é definida em $\mathbb{R}^{M'}$ e centroides em $\mathbb{R}^{\mathbb{M}}$. Dado o mapeamento não-linear e a mudança de número de dimensões entre espaços, projetar os SQs das *features* como proposto na Seção 3.3, utilizando o algoritmo de Lloyd, não garante otimização do quantizador projetado. O projeto de um KPCA-VQ é realizado em dois estágios. O primeiro estágio consiste no projeto de SQs para as *features* individualmente e serve como inicialização para o segundo estágio de projeto. No segundo estágio, todos os limiares, de todas as *features*, são otimizados de forma conjunta via *simulated annealing* (SA) [22]. SA é um método de otimização estocástico, que simula o processo físico de aquecer um material e resfriá-lo lenta e controladamente, de modo que suas moléculas se reorganizem, eliminando defeitos do material. O algoritmo gera perturbações nos limiares dos SQs. Toda perturbação que forneça melhores resultados,

reduzindo a função custo J, é aceita. No início da otimização, com temperatura elevada, perturbações que geram resultados piores são aceitas com maior probabilidade. Conforme a temperatura é reduzida, tendendo ao final da otimização, estas perturbações são menor probabilidade. Aceitar perturbações que pioram J permite que o algoritmo escape de mínimos locais.

O algoritmo de projeto do KPCA-VQ também decide a alocação de bits entre as *features*, como realizado para o projeto dos SQs, utilizando o algoritmo apresentado na Figura 4.2. O algoritmo desta vez é aplicado às *features*, em vez de ser aplicado às componentes no espaço de entrada. Algumas *features* são mais relevantes para a codificação dos dados do que outras. Quantizar estas *features* com mais níveis de quantização resulta em menor distorção. O processo para encontrar a melhor alocação de bits começa por otimizar um único limiar, equivalente a um bit em cada *feature* e zero bits nas outras. A posição do bit que fornecer o menor custo J é utilizada como ponto de partida para otimização dos limiares correspondentes ao segundo bit. O processo segue até que se obtenha o número máximo de bits desejado. Este processo é realizado no primeiro estágio de otimização do KPCA-VQ. Os limiares são otimizados com o algoritmo de Lloyd, de acordo com o número de bits alocado a cada *feature*. Estes limiares são otimizados localmente, dependendo apenas da *feature* associada. O custo minimizado neste primeiro estágio é dado por $J_m = D_m + \lambda_m H_m$, onde o parâmetro λ_m e as variáveis D_m e H_m são associadas à *feature* m individualmente. No segundo estágio, o custo minimizado é $J = D + \lambda H$, implementando otimização com restrição de entropia sobre a partição que foi previamente gerada como ponto de partida.

Os parametros da função kernel escolhida são escolhidos manualmente. Os quantizadores são otimizados repetitivamente e o desempenhos taxa-distorção para diferentes parâmetros são comparados, como mostrado na Figura 4.3. Por exemplo, na optimização de um KPCA-VQ baseado em função kernel RBF, dada por $k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{||\mathbf{x}-\mathbf{y}||^2}{\gamma}\right)$, escolher $\gamma = 1$ gera uma matriz $\mathbf{K}$ relativamente bem condicionada, mas uma relação taxa-distorção ruim. Valores maiores para γ possibilitam desempenho superior, mas geram problemas relacionados ao condicionamento da matriz $\mathbf{K}$.

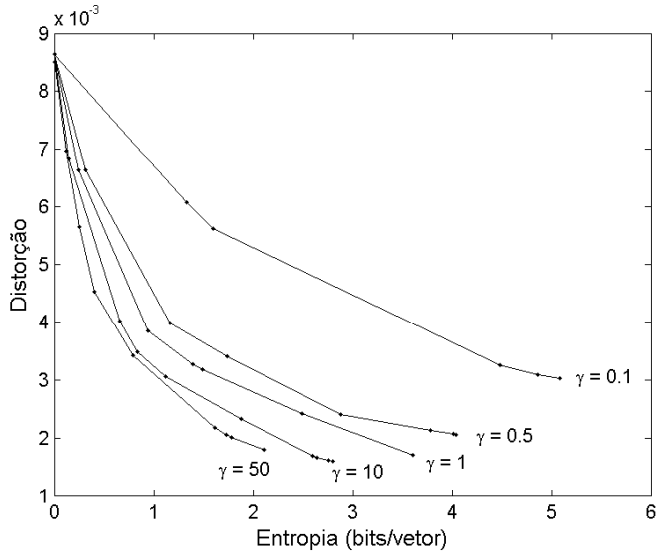


Figura 4.3: Variação das curvas distorção versus entropia para o KPCA-VQ, de acordo com o parâmetro da função kernel RBF.

4.4 Complexidade

Como mostrado na Seção 2.1, a quantização consiste em dois estágios básicos: codificador e decodificador, em que o codificador recebe um vetor de entrada $\mathbf{x}_n$ e calcula o índice binário associado ao vetor do dicionário que melhor representa $\mathbf{x}_n$. A complexidade de um quantizador está associada ao esforço computacional necessário para calcular este índice binário i_n a partir de um vetor de entrada. Este esforço está relacionado às operações matemáticas que o quantizador realiza no cálculo de i_n, como multiplicações, comparações e outras operações mais específicas, como cálculos de algumas funções matemáticas. Operações matemáticas são intrínsecas de cada tipo de quantizador utilizado, mas não só o tipo de quantizador define sua complexidade. Variações em parâmetros de projeto fazem com que quantizadores de um mesmo tipo resultem em complexidades diferentes. Por exemplo, a taxa de bits de um quantizador altera o tamanho do dicionário. Projetar um quantizador com taxa bits elevada resulta em um quantizador mais complexo, dado que mais operações são necessárias para buscar o índice do melhor vetor de reconstrução.

Neste trabalho, alguns custos de complexidade foram atribuídos a operações básicas envolvidas na implementação do SQ, ECVQ e KPCA-VQ, dado um vetor de entrada $\mathbf{x}_t(n)$. A lista a seguir indica estes custos, em termos de unidades de complexidade.

i) Comparação escalar: 1 unidade;

ii) Multiplicação escalar: 1 unidade;

iii) Cálculo de distância Euclideana através do método do produto interno: M unidades;

iv) Calculo da função kernel RBF (assumindo que: função exponencial custa 1 unidade, M cálculos de quadrados custam M unidades; e incluindo o cálculo da distancia Euclideana entre o vetor $\mathbf{x}_t(n)$ e $\mathbf{l}_n$, com $n = 1, \ldots, N'$): $N'(M+1)+M$ unidades;

v) Cálculo da função kernel tangente hiperbólica (assumindo que o cálculo da função tangente hiperbólica de um escalar custa 1 unidade e incluindo o custo computacional do produto interno entre $\mathbf{x}_t(n)$ e $\mathbf{l}_n$): $N'(M+1)$ unidades;

vi) Cálculo da função kernel polinomial: $N'(M+1)$ unidades;

Nas próximas subseções, são calculados os custos totais de complexidade para cada quantizador, de acordo com detalhes de cada implementação.

4.4.1 Complexidade dos Quantizadores Tradicionais

No caso do ECVQ, é necessário encontrar a menor distância Euclideana $d(k)$ de um vetor de entrada $\mathbf{x}_t(n)$ para cada um dos K vetores do dicionário. A distância Euclideana é dada por

$$d(k) = \mathbf{x}_t^T(n)\mathbf{x}_t(n) - 2\mathbf{x}_t^T(n)\mathbf{y}_k + \mathbf{y}_k^T\mathbf{y}_k. \tag{4.1}$$

Para um vetor de entrada $\mathbf{x}_t(n)$, o termo $\mathbf{x}_t^T(n)\mathbf{x}_t(n)$ é igual para todo k. O valor exato de cada $d(k)$ não é relevante. Queremos apenas identificar k para qual $d(k)$ é mínimo. O termo $\mathbf{x}_t^T(n)\mathbf{x}_t(n)$ não precisa ser calculado. O termo $\mathbf{y}_k^T\mathbf{y}_k$ depende apenas dos vetores já conhecidos pelo quantizador e podem ser pré-calculados. Desta forma, apenas o termo $\mathbf{x}_t^T(n)\mathbf{y}_k$ precisa ser calculado. Este termo consiste no produto interno de dois vetores com M dimensões, calculado a partir de M multiplicações escalares. O cálculo de cada distância Euclideana custa M multiplicações, isto é, M unidades de complexidade. São necessários K cálculos de distância Euclideana

para determinar o vetor $\mathbf{y}_k$ mais próximo do vetor de entrada. O custo total do ECVQ é MK unidades de complexidade. Esta é uma estimativa simplificada para a complexidade, dado que algumas operações menos significativas para estimar a complexidade não são consideradas. Este é o caso da operação argmin, que identifica a menor entre K distâncias $d(k)$.

No caso do SQ, existe um quantizador escalar para cada uma das M componentes do vetor de entrada. Estes quantizadores podem possuir taxas de bits diferentes e, portanto, quantidades de limiares diferentes. A quantidade de bits alocada a cada componente é R_m, com $m = 1, \ldots, M$. Cada componente do vetor de entrada é comparada aos limiares do quantizador associado. São realizadas $(2^{R_m} - 1)$ comparações para cada componente. Dado o custo de uma comparação escalar igual a 1 unidade de complexidade, a estimativa total de complexidade para o SQ é $\sum_{m=1}^{M}(2^{R_m} - 1)$ unidades.

4.4.2 Complexidade dos KPCA-VQs

A primeira etapa de um KPCA-VQ é calcular o vetor $\mathbf{z}_t(n)$ dado um vetor de entrada $\mathbf{x}_t(n)$ através da Equação (3.11). A partir deste ponto, a função kernel utilizada no implementação do KPCA-VQ já não afeta diretamente a complexidade do quantizador. Utilizando a Equação (3.16), o vetor de *features* $\mathbf{f}_t$ é calculado a partir de $\mathbf{z}_t$. Para calcular $\mathbf{f}_t = \mathbf{W}\mathbf{z}_t + \mathbf{b}$, são necessárias $M'N'$ multiplicações, em que M' é o número de autovetores $\mathbf{a}_j$ de $\mathbf{K}_c$ e o mesmo número de *features* calculadas. N' é o número de vetores de dados utilizados para gerar a matriz $\mathbf{K}$ e o número de componentes em $\mathbf{z}_t$. Após o calculo das *features*, SQs são aplicados a cada *feature* n, $n = 1, \ldots, M'$. Sendo R_n o número de bits alocados a cada *feature*, o SQs adicionam $\sum_{n=1}^{M'}(2^{R_n} - 1)$.

Para a função kernel RBF, a estimativa total, considerando o cálculo da função kernel, é $N'(M+1) + M + M'N' + \sum_{n=1}^{M'} 2^{R_n} - 1$ unidades de complexidade, onde $N'(M+1)+M$ unidades se devem ao cálculo da função kernel RBF, $M'N'$ se devem ao cálculo de $\mathbf{f}_t = \mathbf{W}\mathbf{z}_t + \mathbf{b}$ e as últimas $\sum_{n=1}^{M'} 2^{R_n} - 1$ unidades se devem aos SQs. Como os custos para as funções kernel tangente hiperbólica e polinomial são iguais a $N'(M+1)$ unidades, os KPCA-VQs para estas funções tem custo estimado total de $N'(M+1) + M'N' + \sum_{n=1}^{M'} 2^{R_n} - 1$ unidades de complexidade.

4.4.3 Condicionamento da Matriz K

Como citado na Seção 4.3, a escolha dos parâmetros da função kernel afeta o condicionamento das matrizes do KPCA-VQ. A Equação (4.2) mostra a matriz $\mathbf{W}$ obtida a partir da função kernel RBF com $\gamma = 50$. Alguns valores maiores que zero são mostrados como $0,000$, pois são pequenos demais para a notação adotada, enquanto outros valores são relativamente grandes. Temos o menor valor de $\mathbf{W}_{50}$, em módulo, igual a $0,23$, enquanto o maior valor é dado por $2,23 \times 10^3$, enquanto os outros valores estão espalhados por todo o intervalo intermediário. Esta disparidade de valores torna a implementação desta matriz através de circuitos analógicos em tecnologia CMOS uma tarefa inviável.

$$\mathbf{W}_{50} = \begin{bmatrix} -0,002 & -0,016 & -0,172 & 0,004 & -0,082 & 0,180 \\ 0,001 & -0,015 & 0,035 & 0,054 & 0,452 & 2,227 \\ -0,001 & -0,065 & 0,052 & 0,055 & -0,240 & -0,864 \\ 0,000 & 0,008 & 0,006 & -0,082 & 0,307 & -0,311 \\ 0,001 & 0,013 & 0,026 & -0,120 & -1,061 & 0,757 \\ 0,000 & 0,008 & 0,012 & -0,081 & 0,418 & -0,447 \\ -0,025 & 0,028 & 0,017 & 0,166 & -0,118 & -0,275 \\ 0,024 & 0,023 & -0,000 & 0,185 & -0,121 & -0,414 \\ 0,000 & 0,008 & 0,012 & -0,081 & 0,418 & -0,447 \\ 0,001 & 0,009 & 0,013 & -0,100 & 0,027 & -0,406 \end{bmatrix} \times 10^3 \qquad (4.2)$$

A Equação (4.3) mostra a matriz $\mathbf{W}_{0.1}$, calculada para a função RBF com $\gamma = 0.1$. Neste caso, a matriz está melhor condicionada e a implementação em hardware é mais acessível. Como visto na Figura 4.3, o desempenho do quantizador para $\gamma = 0.1$ é muito inferior ao desempenho obitdo para $\gamma = 50$. Desta forma, os quantizadores projetados terão matrizes mal-condicionadas. Por causa do mau condicionamento, as *features* calculadas tendem a ocupar uma faixa dinâmica várias ordens de grandeza inferior à das variáveis que compõem os vetores de entrada da rede neural. No caso de uma implementação do método KPCA proposto utilizando circuitos integrados analógicos (o que aconteceria, por exemplo, no caso implementação no plano focal de câmeras CMOS), a perda de faixa dinâmica causaria problemas de quali-

dade da imagem decodificada,por causa do ruído típico da operação destes circuitos. Para lidar com este problema de perda de faixa dinâmica, a utilização de circuitos amplificadores deve ser levada em consideração no futuro, inclusive sendo levada em conta a complexidade associada ao uso destes mesmos amplificadores. Por simplicidade, no presente trabalho, a análise da complexidade do KPCA-VQ será feita com base em uma implementação em software, e a análise de complexidade específica para a implementação baseada em circuitos integrados analógicos será deixada para investigação futura.

$$\mathbf{W}_{0.1} = \begin{bmatrix} 0,378 & -0,420 & -0,696 & 7,594 & 0,091 & 5,626 \\ 0,526 & 0,005 & -0,587 & -1,452 & -21,230 & 10,352 \\ 0,261 & -0,326 & -3,754 & -2,327 & 7,289 & -2,321 \\ 0,479 & -0,026 & 0,905 & -0,368 & 2,281 & -19,462 \\ 0,457 & 0,141 & 1,155 & -1,422 & 7,293 & 44,659 \\ 0,484 & 0,002 & 0,875 & -0,641 & 1,103 & -18,561 \\ -2,046 & -1,898 & 0,546 & -0,603 & -0,958 & 0,837 \\ -1,503 & 2,474 & -0,268 & 0,614 & -0,168 & -1,440 \\ 0,484 & 0,002 & 0,875 & -0,641 & 1,103 & -18,561 \\ 0,480 & 0,046 & 0,950 & -0,754 & 3,197 & -1,129 \end{bmatrix} \tag{4.3}$$

Capítulo 5

Aplicações e Resultados

A base de dados utilizada no projeto e teste dos quantizadores é extraída de imagens naturais, divididas em blocos de 4×4 pixels. A transformada discreta de cosseno (DCT, de *discrete cosine transform*) é aplicada a cada bloco de pixels, resultando em 16 coeficientes no domínio da transformada. O primeiro coeficiente é o coeficiente de frequência zero, associado à componente DC do bloco de pixels. Estamos interessados em realizar quantização da informação de textura dos blocos e, portanto, o primeiro coeficiente será descartado. Dos 15 coeficientes restantes, mantemos apenas os cinco coeficientes de maior energia [23]. Com isso, compressão de dados com perdas já é aplicada à imagem antes da quantização. Os vetores de textura 5-D que serão utilizados no projeto e teste dos quantizadores são formados pelos 5 coeficientes mantidos. Duas bases de dados são utilizadas para treinos e cada uma consiste de vetores 5-D extraídos de seis imagens naturais. A primeira base de treino é dedicada à quantização de imagens de rostos humanos e as seis imagens da base são imagens de rostos. A segunda base é uma base mista, com três imagens de rostos humanos e três imagens de natureza, envolvendo animais e florestas. Duas bases de dados para teste são utilizadas. A primeira é formada por vetores 5-D extraídos de uma imagem de rosto humano feminino. A segunda é formada por vetores 5-D extraídos de uma imagem de araras na natureza. Todas as imagens utilizadas estão em escala de cinza e possuem tamanho 480×640 pixels [1]. As análises feitas dos quantizadores incluem análise de desempenho taxa-distorção-complexidade,

[1]Para solicitar mais informações sobre a base de dados ou solicitar a própria base de dados, pode-se entrar em contato pelo e-mail: vtrmeireles@pads.ufrj.br

como apresentado na Figura 5.1 e análise do desempenho dos quantizadores como um dos estágios de um sistema de compressão de imagens.

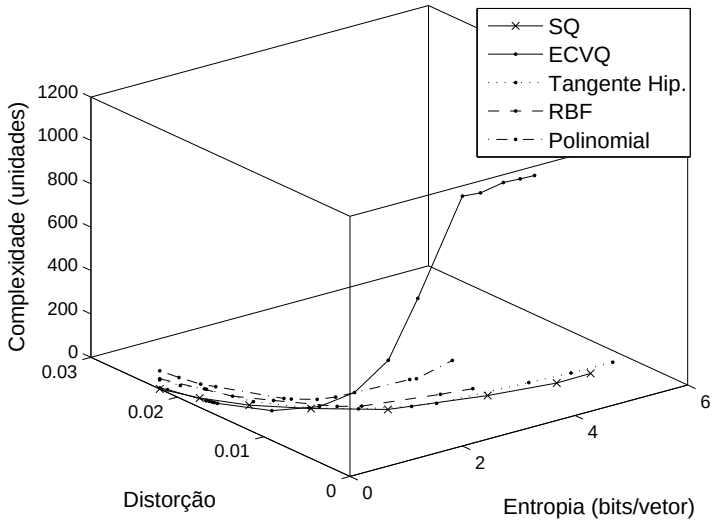


Figura 5.1: Curvas taxa-distorção-complexidade.

5.1 Análise Taxa-Distorção

Analisamos o desempenho taxa-distorção dos quantizadores apresentados, variando a quantidade de bits utilizada no projeto dos quantizadores e calculando a distorção e entropia alcançadas por cada sistema projetado. Para cada tipo de quantizador e quantidade de bits utilizada, vários projetos são realizados, com diferentes inicializações. Diferentes projetos geram diferentes pares taxa-distorção que são espalhados em uma nuvem de pontos pelo plano H-D. As cascas convexas inferiores destas nuvens de pontos são utilizadas para comparar os quantizadores.

Projetos dos KPCA-VQs foram conduzidos para dois diferentes conjuntos de parâmetros. O primeiro tem a quantidade limite de *features* $M'_{\max}$ igual a 6 *features*. O segundo tem $M'_{\max}$ igual a 3 *features*. Definir $M'_{\max}$ com um dado valor não significa que o KPCA-VQ terá exatamente aquela quantidade de *features*. O algoritmo que decide a alocação de bits entre as *features* discutido na Seção 4.3 pode encontrar melhores resultados mantendo 0 bits em uma ou mais das *features* possíveis. Desta forma, o valor M' efetivo para cada quantizador está apenas limitado superiormente por 6 ou 3 *features*. Esta informação será mais relevante ao

tratarmos as complexidades dos quantizadores. A base de dados utilizada para as análises taxa-distorção é a base de dados dedicada à quantização de rostos.

A Figura 5.2 mostra os resultados de desempenho taxa-distorção para os quantizadores tradicionais, SQ e ECVQ, bem como para os KPCA-VQs, com as três funções kernels apresentadas. Nesta análise, os gráficos mostram KPCA-VQs projetados com $M'_{\max} = 6$. Para a região onde a entropia é menor que um, neste exemplo, os KPCA-VQs apresentam desempenho taxa-distorção igual ao do ECVQ, como pode ser observado pelas curvas superpostas. Para maiores entropias, os KPCA-VQs apresentam desempenho intermediário em relação ao SQ e ECVQ. Nestes casos, o KPCA-VQ baseado na função kernel $\tanh(x)$ alcança melhor desempenho que os KPCA-VQs baseados nas funções kernel RBF e polinomial. É importante notar que, embora a curva para o ECVQ esteja pior que a curva para o KPCA-VQ em alguns pontos, ECVQs podem ser projetados de modo que tenham resultados iguais aos KPCA-VQs que se mostraram melhores. No caso deste trabalho, o desempenho do ECVQ está limitado pelo algoritmo de otimização adotado.

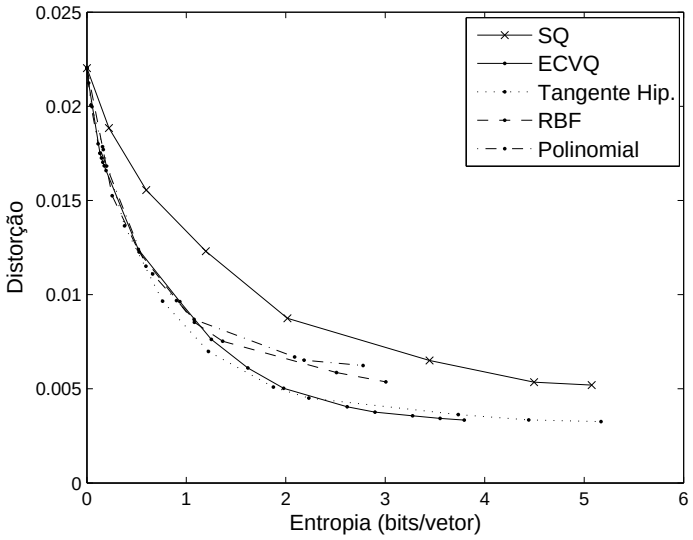


Figura 5.2: Curvas taxa-distorção.

As Figuras 5.3, 5.4 e 5.5 mostram a comparação de desempenho taxa-distorção entre KPCA-VQs projetados para $M'_{\max} = 3$ com os projetados para $M'_{\max} = 6$.

Para o KPCA-VQ baseado na função kernel tangente hiperbólica, existe perda de desempenho em termos de curva taxa-distorção quando o número máximo de *features* é reduzido de $M'_{\max} = 6$ para $M'_{\max} = 3$. Para os outros KPCA-VQs, base-

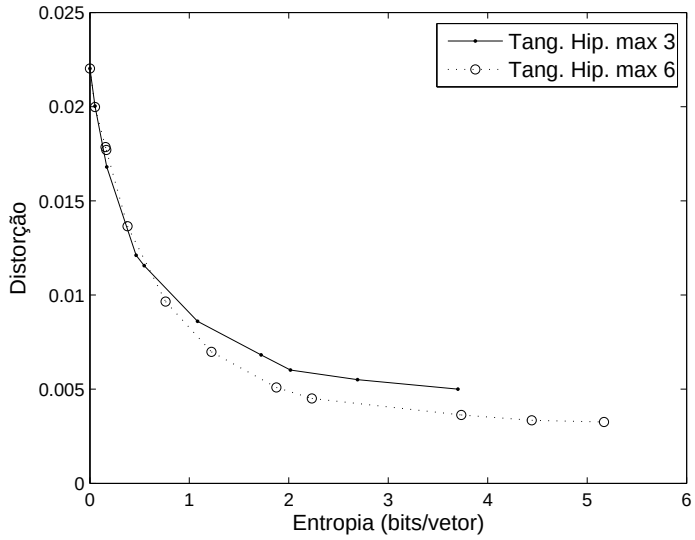


Figura 5.3: Curvas taxa-distorção para KPCA-VQ baseado na função tangente hiperbólica com $M'_{\max} = 3$ e $M'_{\max} = 6$.

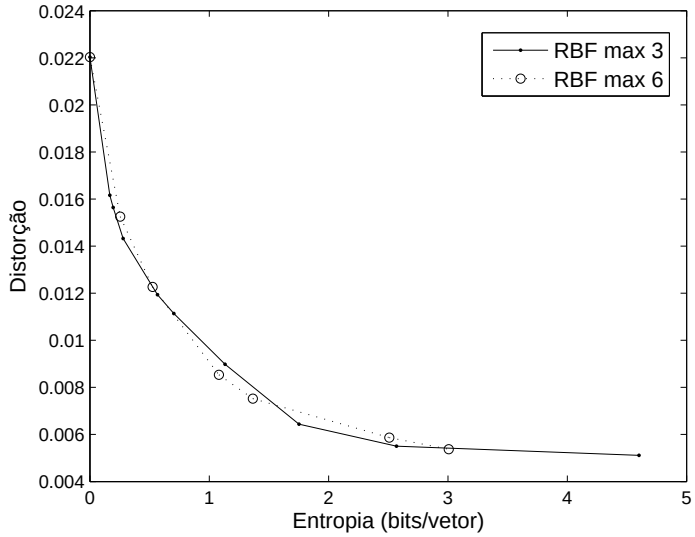


Figura 5.4: Curvas taxa-distorção para KPCA-VQ baseado na função RBF com $M'_{\max} = 3$ e $M'_{\max} = 6$.

ados em função kernel RBF e polinomial, o desempenho para diferentes $M'_{\max}$ não sofreu alterações significativas. Baseado nestes dados, pode-se supor que a energia fica mais uniformemente distribuída ao longo das diferentes *features* no KPCA-VQ baseado na função tangente hiperbólica do que nos outros KPCA-VQs. Quando a alocação de bits resultante do algoritmo de otimização é analisada, temos que, para sistemas de 8 bits, $\mathbf{binalloc}_{\tanh} = [2\,2\,1\,1\,1\,1]$, onde $\mathbf{binalloc}_{\tanh}$ é o vetor que representa a alocação de bits ao longo das *features* para o KPCA-VQ baseado na função

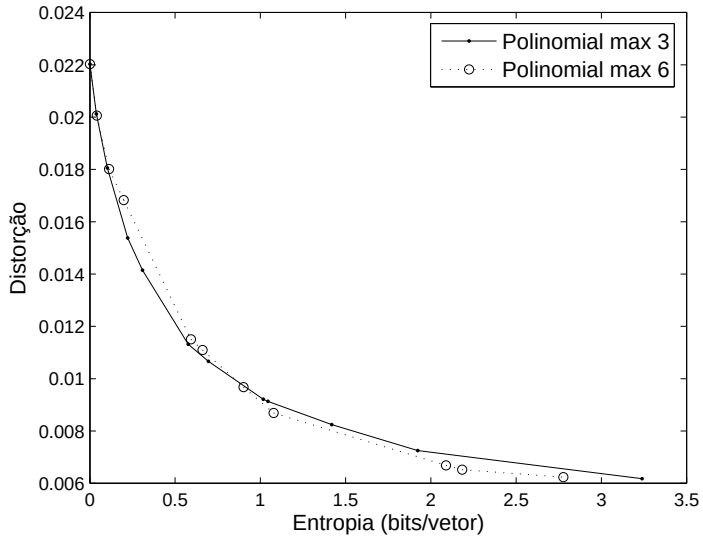


Figura 5.5: Curvas taxa-distorção para KPCA-VQ baseado na função polinomial com $M'_{\max} = 3$ e $M'_{\max} = 6$.

kernel tangente hiperbólica. Neste caso, temos 2 bits alocados a cada uma das duas primeiras *features* e 1 bit alocado a cada uma das outras quatro *features*. Para o KPCA-VQ baseado em função kernel RBF, temos $\mathbf{binalloc}_{\mathrm{RBF}} = [2\,4\,1\,0\,0\,1]$ e para o KPCA-VQ baseado em função kernel polinomial, $\mathbf{binalloc}_{\mathrm{poly}} = [3\,4\,0\,0\,1\,0]$. Estes dados mostram que o algoritmo de otimização deu maior preferência às primeiras *features* nos KPCA-VQs baseados em função kernel RBF e polinomial, enquanto todas as seis *features* mostraram-se relevantes no KPCA-VQ baseado em função tangente hiperbólica. Esta análise reforça a hipótese de que utilizar a função kernel tangente hiperbólica resulta em maior espalhamento da energia nas *features*, quando comparada às outras funções kernel.

5.2 Análise de Complexidade

As complexidades necessárias para quantizar um vetor de teste de cada quantizador foram estimadas em função dos custos propostos na Seção 4.4. A Figura 5.6 mostra as complexidades estimadas para todos os quantizadores, em unidades de complexidade, como uma função da entropia. Entre os quantizadores testados, é esperado que o SQ seja o menos complexo, enquanto o ECVQ seja o quantizador que requer maior custo computacional. Para entropia baixa, abaixo de 1, o ECVQ é menos complexo que o KPCA-VQ. A partir de $H = 1,2$, o ECVQ se torna mais com-

plexo que todos KPCA-VQs, enquanto o SQ é menos complexo que todos os outros para toda faixa de entropia. Levando em conta KPCA-VQs com a mesma função kernel e o mesmo número M' de *features* utilizadas, grande parte da complexidade dos KPCA-VQs está associada aos cálculos da função kernel e das *features*. Neste caso, esta parte da complexidade é constante, dados quantizadores com diferentes alocações de bits na camada de saída da rede neural. A única parte da estimativa de complexidade que depende da taxa de bits é a parte associada aos SQs aplicados às *features*. Desta forma, a curva de complexidade do KPCA-VQ é semelhante à curva do SQ, a menos do deslocamento no eixo das unidades de complexidade dado pela parte constante da estimativa. Tanto o KPCA-VQ como o SQ apresentam inclinações da curva de complexidade em função da entropia relativamente baixa, quando comparadas à mesma inclinação no ECVQ. Com isso, para entropias mais altas, o ECVQ se torna consideravelmente mais complexo.

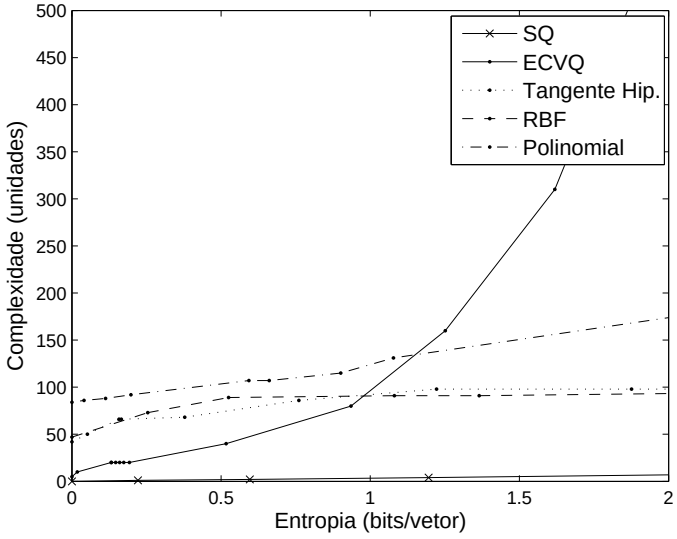


Figura 5.6: Complexidade de quantização como função da entropia de quantização.

A Figura 5.7 mostra as curvas de estimativas de complexidade como função da distorção resultante dos quantizadores. Para quantizadores com distorção elevada, os KPCA-VQs são mais complexos que o ECVQ, mas conforme quantizadores alcançam menores distorções, o ECVQ se torna mais complexo que todas as outras técnicas. Como visto para as curvas de complexidade em função da entropia, KPCQ-VQs e SQ são pouco dependentes da complexidade, enquanto ECVQ requer alta complexidade para alcançar baixas distorções.

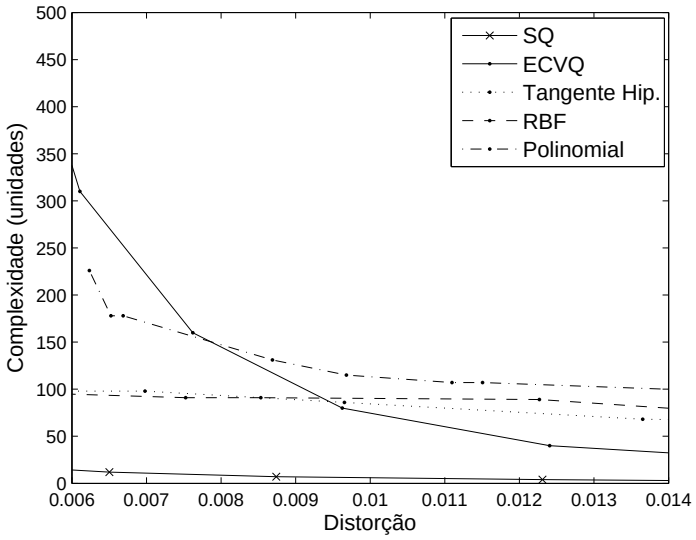
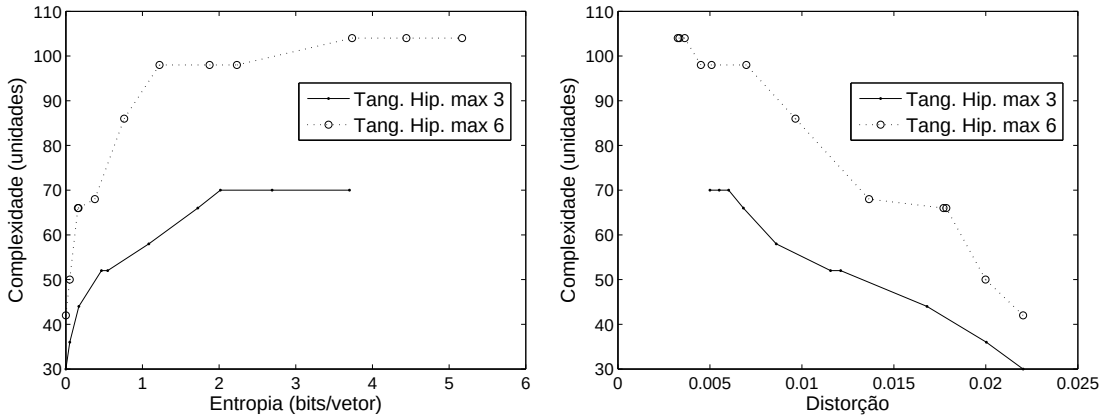


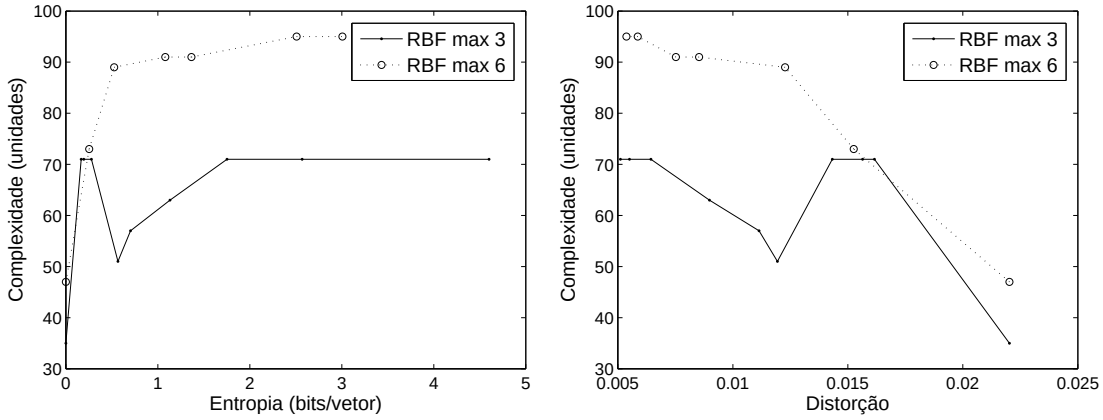
Figura 5.7: Complexidade de quantização como função da distorção de quantização.

Para a análise de complexidade, as comparações para KPCA-VQs com $M'_{\max} = 3$ e $M'_{\max} = 6$ são mostradas nas Figuras 5.8, 5.9 e 5.10. Para a complexidade, os efeitos da limitação em M' são mais diretos que para as curvas taxa-distorção. Reduzir a quantidade de *features* reduz diretamente a quantidade de cálculos necessários para a quantização. Desta forma, o custo computacional para $M'_{\max} = 3$ é menor que o custo para $M'_{\max} = 6$ em todos os casos.



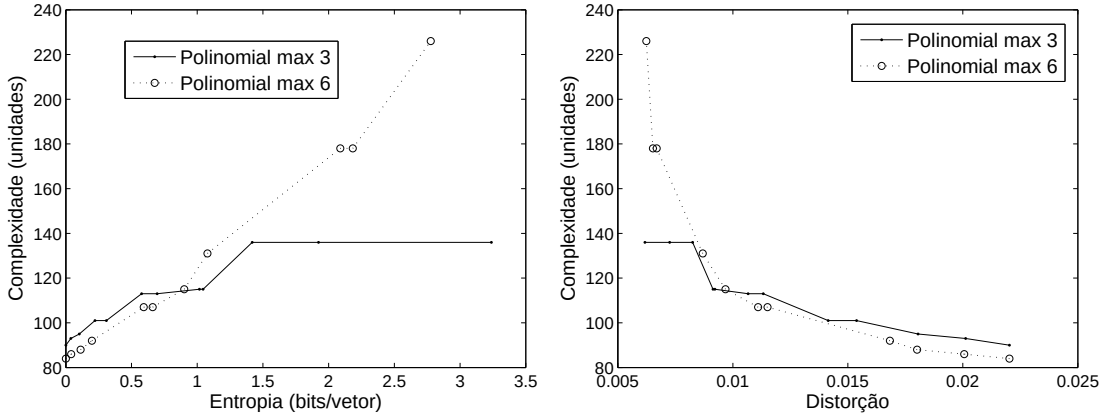
(a) (b)

Figura 5.8: Comparação entre complexidades para $M'_{\max} = 3$ e $M'_{\max} = 6$ para KPCA-VQ baseado na função kernel tangente hiperbólica. (a) Complexidade como função da entropia; (b) Complexidade como função da distorção.



(a) (b)

Figura 5.9: Comparação entre complexidades para $M'_{\max} = 3$ e $M'_{\max} = 6$ para KPCA-VQ baseado na função kernel RBF. (a) Complexidade como função da entropia; (b) Complexidade como função da distorção.



(a) (b)

Figura 5.10: Comparação entre complexidades para $M'_{\max} = 3$ e $M'_{\max} = 6$ para KPCA-VQ baseado na função kernel polinomial. (a) Complexidade como função da entropia; (b) Complexidade como função da distorção.

5.3 Otimização Taxa-Distorção-Complexidade

As Seções 5.1 e 5.2 mostraram análises de desempenho taxa-distorção e de complexidade das técnicas estudadas. Dada uma representação em função destes três parâmetros – entropia, distorção e complexidade – para cada quantizador, podemos sintetizar um processo de otimização no projeto do quantizador que considere a complexidade associada. Assim, é possível projetar um quantizador que busque o melhor desempenho taxa-distorção, dado um limite de complexidade, por exemplo $\theta < 100$ unidades. Até então, os projetos dos quantizadores foram realizados de

modo independente da complexidade. A função custo de otimização dupla utilizada até agora é dada por $J = D + \lambda H$. Para realizar a otimização tripla, devemos projetar um quantizador minimizando uma função custo taxa-distorção

$$J_{tripla} = D + \lambda H + \gamma\theta. \tag{5.1}$$

Dado o custo J e os custos de complexidade estimados na Seção 5.2, podemos definir um plano θ-J e a otimização tripla passa a ser, então, conduzida como uma otimização dupla (uma solução similar, particular para os casos de SQ e ECVQ, foi proposta em [24]).

Para um tipo de quantizador, pares (θ,J) formam um conjunto de pontos no plano θ,J. Usando regressão polinomial, é possível definir uma função polinomial que se aproxime da curva $J(\theta)$, dada pelo custo taxa-distorção como função da complexidade do quantizador. Conhecendo a função que aproxima $J(\theta)$ e dada uma restrição $\theta < \theta_0$ para a complexidade, é possível calcular o valor do Lagrangiano γ. Substituindo o valor do Lagrangiano na Equação (5.1), pode-se realizar uma otimização conjunta de três variáveis equivalente à uma otimização taxa-distorção com restrição de entropia.

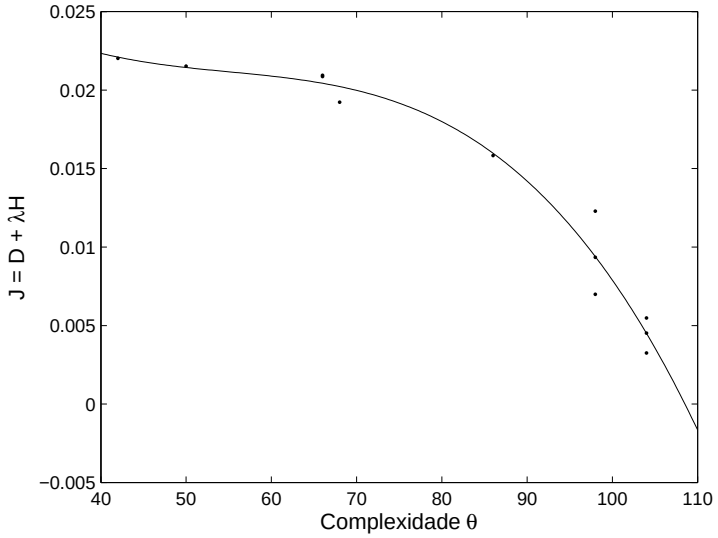


Figura 5.11: Pares (θ,J) calculados para o KPCA-VQ baseado em função kernel tangente hiperbólica e curva resultante da regressão polinomial aplicada aos pares.

A Figura 5.11 mostra um exemplo do plano θ-J para os dados da quantização com o KPCA-VQ baseado em função kernel tangente hiperbólica. Os pontos espalhados

pelo plano são os pares (θ,J) obtidos neste trabalho. A curva formada entre estes pontos consiste de um polinômio de grau 3 obtido através da regressão polinomial dos pares (θ,J). O polinômio que forma a curva é dado por $-1.20 \times 10^{-7}\theta^3 + 1.98 \times 10^{-5}\theta^2 - 1.14 \times 10^{-3}\theta + 4.40 \times 10^{-2}$. Calculando o valor do Lagrangiano γ para uma restrição, por exemplo, de $\theta < 100$, obtemos $\gamma = -7.80 \times 10^{-4}$.

5.4 Compressão de Imagens

Dos quantizadores analisados, selecionamos um projeto de ECVQ e dois projetos do KPCA-VQ baseado na função kernel tangente hiperbólica a serem testados como parte de um sistema de compressão de imagens, sendo um KPCA-VQ com $M'_{\max} = 6$ e um KPCA-VQ com $M'_{\max} = 3$. O sistema de compressão consiste, inicialmente, da divisão da imagem a ser quantizada em blocos de 4×4 pixels. A DCT é aplicada a estes blocos e são gerados 16 coeficientes. O coeficiente DC e os cinco coeficientes de frequência não-nula com maior energia são mantidos. Todos os outros coeficientes são descartados. Os cinco coeficientes formam o vetor 5-D que é quantizado pelo ECVQ ou pelo KPCA-VQ. O coeficiente DC não sofre quantização. Apenas a informação de textura é quantizada. A reconstrução da imagem começa com a formação de novos blocos 4×4 com o coeficiente DC e os cinco coeficientes quantizados. Os outros onze coeficientes recebem o valor zero. A transformada discreta de cosseno inversa (IDCT, de *inverse discrete cosine transform*) é aplicada aos blocos e os blocos de 4×4 pixels da imagem quantizada são gerados. O conjunto de todos os blocos gerados é a imagem quantizada completa. O diagrama de blocos na Figura 5.12 descreve o sistema de compressão.

Os testes dos quantizadores em sistemas de compressão foram realizados para as duas imagens que geram as duas bases de vetores de teste utilizadas neste trabalho. Para a primeira imagem, apresentada na Figura 5.13, à qual nos referimos como *smile*, a base de dados dedicada à quantização de imagens de rostos foi utilizada no projeto dos quantizadores. Para a segunda imagem, Figura 5.14, denominada *araras*, a base de dados de treino mista foi utilizada.

As Figuras 5.15 e 5.16 mostram os resultados da compressão com baixa entropia e com alta entropia da imagens *smile*, respectivamente. Os resultados para a imagem

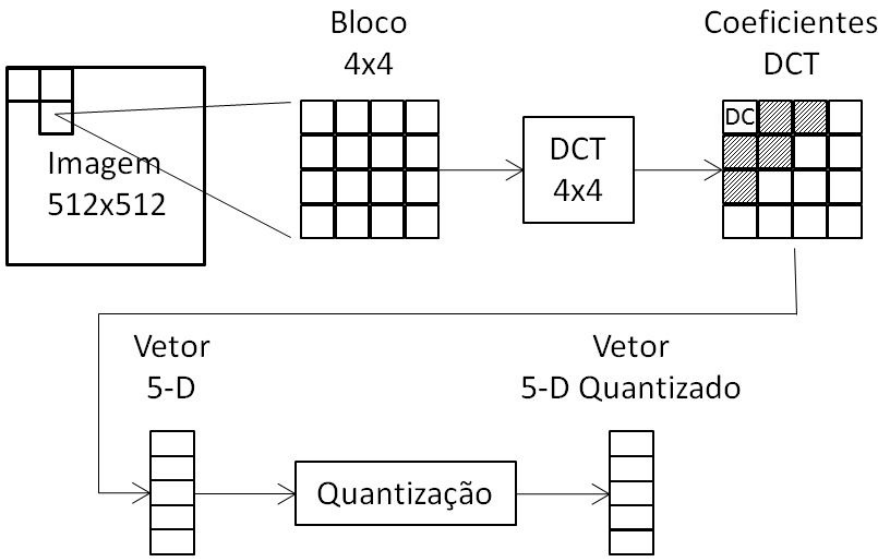


Figura 5.12: Diagrama de blocos do sistema de compressão.

Figura 5.13: Imagem de teste *smile*.

araras são mostrados nas Figuras 5.17 e 5.18. As imagens cujos vetores foram quantizados pelo ECVQ estão no topo das figuras, enquanto aquelas quantizadas por KPCA-VQ com $M'_{\max} = 6$ estão no meio e com $M'_{\max} = 3$ na parte inferior.

As Tabelas 5.1, 5.2, 5.3 e 5.4 apresentam os dados dos quantizadores utilizados nos testes de compressão. Estes dados são referentes apenas aos quantizadores e não

Figura 5.14: Imagem de teste *araras*.

ao sistema de compressão completo, de modo que os valores das distorções apresentados são calculados entre os vetores 5-D reconstruídos e os originais. Estes valores refletem os resultados obtidos nas análises de desempenho taxa-distorção e de complexidade realizadas neste trabalho. Para entropia relativamente baixa, apresentada em torno de $H = 0,17$ para imagem *smile* e $H = 0.56$ para a imagem *araras*, temos desempenhos taxa-distorção próximos ao comparar o ECVQ e o KPCA-VQ baseado na função tangente hiperbólica. Nesta faixa de entropia, os KPCQ-VQs são mais complexos que o ECVQ. Para maiores entropias, o KPCA-VQ baseado na função kernel tangente hiperbólica alcança resultados próximos ao ECVQ para uma complexidade consideravelmente mais baixa. Para o KPCA-VQ com $M'_{\max} = 3$, o desempenho taxa-distorção fica abaixo do desempenho para 6 *features*, em troca de menor custo computacional.

Analisando as imagens comprimidas, todos os quantizadores mostram desempenho semelhante. Todos causam artefatos similares em função do processamento em blocos. Para uma mesma faixa de entropia e distorção, um KPCA-VQ pode substituir um ECVQ em sistemas de compressão, alcançando resultados que, na prática, apresentam diferenças desprezíveis para o usuário.

Tabela 5.1: Dados dos quantizadores aplicados na compressão com baixa entropia da imagem *smile*, correspondentes aos resultados apresentados na Figura 5.15

Quantizador	Entropia	Distorção	Complexidade
ECVQ	0,16	$0,17 \times 10^{-1}$	20
KPCA Tangente Hip. $M'_{\max} = 6$	0,17	$0,18 \times 10^{-1}$	66
KPCA Tangente Hip. $M'_{\max} = 3$	0,17	$0,17 \times 10^{-1}$	44

Tabela 5.2: Dados dos quantizadores aplicados na compressão com alta entropia da imagem *smile*, correspondente aos resultados apresentados na Figura 5.16

Quantizador	Entropia	Distorção	Complexidade
ECVQ	1,98	$0,50 \times 10^{-2}$	590
KPCA Tangente Hip. $M'_{\max} = 6$	1,88	$0,51 \times 10^{-2}$	98
KPCA Tangente Hip. $M'_{\max} = 3$	2,01	$0,60 \times 10^{-2}$	70

Tabela 5.3: Dados dos quantizadores aplicados na compressão com baixa entropia da imagem *araras*, correspondentes aos resultados apresentados na Figura 5.17

Quantizador	Entropia	Distorção	Complexidade
ECVQ	0,54	$0,38 \times 10^{-1}$	20
KPCA Tangente Hip. $M'_{\max} = 6$	0,57	$0,36 \times 10^{-1}$	60
KPCA Tangente Hip. $M'_{\max} = 3$	0,58	$0,36 \times 10^{-1}$	44

Tabela 5.4: Dados dos quantizadores aplicados na compressão com alta entropia da imagem *araras*, correspondentes aos resultados apresentados na Figura 5.18

Quantizador	Entropia	Distorção	Complexidade
ECVQ	1,92	$1,4 \times 10^{-2}$	320
KPCA Tangente Hip. $M'_{\max} = 6$	1,95	$1,5 \times 10^{-2}$	80
KPCA Tangente Hip. $M'_{\max} = 3$	2,14	$1,7 \times 10^{-2}$	66

Figura 5.15: Resultados da compressão com baixa entropia da imagem *smile*. No topo, vetores quantizados por ECVQ. No meio, KPCA-VQ baseado na função tangente hiperbólica com $M'_{max} = 6$. Na parte inferior, KPCA-VQ baseado na função tangente hiperbólica com $M'_{max} = 3$.

Figura 5.16: Resultados da compressão com alta entropia da imagem *smile*. No topo, vetores quantizados por ECVQ. No meio, KPCA-VQ baseado na função tangente hiperbólica com $M'_{max} = 6$. Na parte inferior, KPCA-VQ baseado na função tangente hiperbólica com $M'_{max} = 3$.

Figura 5.17: Resultados da compressão com baixa entropia da imagem *araras*. No topo, vetores quantizados por ECVQ. No meio, KPCA-VQ baseado na função tangente hiperbólica com $M'_{max} = 6$. Na parte inferior, KPCA-VQ baseado na função tangente hiperbólica com $M'_{max} = 3$.

Figura 5.18: Resultados da compressão com alta entropia da imagem *araras*. No topo, vetores quantizados por ECVQ. No meio, KPCA-VQ baseado na função tangente hiperbólica com $M'_{max} = 6$. Na parte inferior, KPCA-VQ baseado na função tangente hiperbólica com $M'_{max} = 3$.

Capítulo 6

Conclusões

Este trabalho estudou as propriedades da quantização vetorial baseada em Kernel PCA aplicada a vetores 5-D extraídos de DCTs 4×4 de imagens naturais. O KPCA-VQ foi proposto como um modelo de rede neural com uma camada escondida e a viabilidade de sua implementação em sistemas de compressão de imagens foi analisada, através de comparações com quantizadores tradicionais, como SQ e ECVQ. Aplicações atuais requerem cada vez mais que custo, dimensões e consumo de energia sejam reduzidos. Dessa forma, a complexidade computacional apresenta um papel importante na análise das técnicas de quantização e de qualquer outro estágio de um sistema eletrônico. Por isso, o foco deste trabalho não foi apenas o desempenho taxa-distorção do KPCA-VQ, mas também consideramos seu custo computacional.

Apresentamos KPCA-VQs baseados em três diferentes funções kernel e descrevemos os resultados obtidos para as três funções. Cada KPCA-VQ foi analisado em duas configurações diferentes, com relação ao número de *features* máximo permitido ao projeto dos quantizadores. As estruturas dos quantizadores foram descritas e os algoritmos utilizados na otimização destas estruturas foram apresentados. A KPCA é proposta como uma camada escondida de rede neural que executa a função kernel, seguida por uma camada de saída linear que executa a extração das *features* dos dados no $\mathbb{R}^I$. A quantização é feita através de uma partição definida por SQs em $\mathcal{F}$ associada a pontos de reconstrução no espaço de entrada $\mathbb{R}^M$. O método para transição entre $\mathbb{R}^{M'}$ e $\mathbb{R}^M$ durante a quantização também foi detalhado. Foi proposto, ainda, um método para otimização tripla de quantizadores, com função custo

baseada nos três parâmetros discutidos – entropia, distorção e complexidade.

Todos os KPCA-VQs apresentaram, em geral, resultados intermediários em relação às técnicas tradicionais de quantização. Em relação ao desempenho taxa-distorção, o KPCA-VQ baseado em função tangente hiperbólica apresentou resultados competitivos com o ECVQ, enquanto os KPCA-VQs baseados nas funções RBF e polinomial apresentaram curvas taxa-distorção melhores que o SQ, mas ainda consideravelmente abaixo do melhor KPCA-VQ e do ECVQ. Como esperado, o ECVQ mostrou-se consideravelmente dependente da complexidade conforme buscamos sistemas com melhor desempenho taxa-distorção. As curvas taxa-complexidade e distorção-complexidade mostraram inclinações elevadas para o ECVQ, resultando em crescimentos rápidos da complexidade em função de D e H. Para as outras técnicas, incluindo KPCA-VQ e SQ, as curvas têm inclinação relativamente muito menor que o ECVQ, de forma que esses quantizadores são menos dependentes de operações computacionais para alcançar baixa distorção. Finalmente, a técnica de quantização proposta foi aplicada num sistema de compressão de imagens e comparada com o ECVQ. Os quantizadores foram treinados com duas bases de dados de treino diferentes e a compressão foi aplicada a duas imagens de teste. Os resultados da aplicação do KPCA-VQ em sistemas de compressão mostram que esta técnica pode ser utilizada sem perdas consideráveis em relação a técnicas tradicionais, como o ECVQ, ao mesmo tempo que requer um custo computacional consideravelmente menor.

Em trabalhos anteriores, foi realizado o estudo de quantizadores baseados em kernel PCA com o uso de bases de dados genéricas [25, 26]. Este trabalho mostra a aplicação do método proposto em imagens reais e faz uma análise mais aprofundada dos quantizadores baseados em função kernel, mostrando sua relação com redes neurais e analisando a viabilidade de sua implementação. O conteúdo deste trabalho foi aceito para publicação em forma de artigo [27].

Como trabalho futuro, pretende-se estudar com mais detalhes o efeito das funções kernel no processo de quantização e desenvolver métodos de otimização dos parâmetros destas funções, uma vez que esta etapa do projeto foi realizada através de tentativa e erro. Além disso, como consequência dos parâmetros utilizados nas funções kernel, a aplicação em tecnologia CMOS da técnica proposta mostrou-se, por enquanto, inviável, de modo que o estudo do condicionamento de matrizes, voltado para imple-

mentação, é um tema aberto para trabalhos futuros. Além disso, os quantizadores podem ser testados para bases de dados diferentes e maiores. As bases utilizadas para treinos foram extraídas de apenas seis imagens cada, dadas as limitações das ferramentas de cálculo numérico utilizadas no projeto. Utilizar bases maiores comprometeria o desenvolvimento do trabalho. Por fim, novas formas de análise de desempenho podem ser aplicadas ao KPCA-VQ, como outras métricas objetivas e também métricas subjetivas de avaliação de imagens.

Referências Bibliográficas

[1] OLIVEIRA, F., HAAS, H., GOMES, J., *et al.*, "CMOS Imager with Focal-Plane Analog Image Compression Combining DPCM and VQ", *IEEE Trans. Circuits and Systems I: Regular Papers*, v. 60, n. 5, pp. 1331 – 1344, Maio 2013.

[2] OLIVEIRA, F., GOMES, J., PETRAGLIA, A., "Comparison of Low-Complexity Image Compression Algorithms for Analog Circuit Implementation", *Cellular Nanoscale Networks and their Applications (CNNA), 14th International Workshop on*, pp. 1 – 2, Julho 2014.

[3] HEIDERMANN, G., RITTER, H., "Data compression - A generic principle of pattern recognition?", *Computer Vision and Computer Graphics – Theory and Application*, pp. 202 – 212, 2009.

[4] RUEDA, L. G., *Advances in Data Compression and Pattern Recognition*. Carleton University, Ottawa, ON, Canada, 2002.

[5] KALAIVANI, K., THIRUMARAISELVI, C., SUDHAKAR, R., "Multi resolution transform and artificial neural network based digital image coder", *Electronics and Communication Systems (ICECS), International Conference on*, pp. 1 – 4, Fevereiro 2014.

[6] CHANG, C.-H., XU, P., XIAO, R., *et al.*, "New adaptive color quantization method based on self-organizing maps", *Neural Networks, IEEE Transactions on*, v. 16, n. 1, pp. 237 – 249, Janeiro 2005.

[7] LAWRENCE, S., GILES, C., TSOI, A. C., *et al.*, "Face recognition: a convolutional neural-network approach", *Neural Networks, IEEE Transactions on*, v. 8, n. 1, pp. 98 – 113, Janeiro 1997.

[8] SHANNON, C. E., "A mathematical theory of communication", *Bell System Technical Journal, The*, v. 27, n. 3, pp. 379 – 423, Julho 1948.

[9] SHANNON, C. E., "A mathematical theory of communication", *Bell System Technical Journal, The*, v. 27, n. 4, pp. 623 – 656, Outubro 1948.

[10] HUFFMAN, D., "A Method for the Construction of Minimum-Redundancy Codes", *Proceedings of the IRE*, v. 40, pp. 1098 – 1101, Setembro 1952.

[11] SHANNON, C. E., "Coding Theorems for a Discrete Source With a Fidelity Criterion". In: Sloane, N., Wyner, A. (eds.), *Claude Elwood Shannon: Collected Papers*, Wiley-IEEE Press, 1993.

[12] GERSHO, A., GRAY, R. M., *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1991.

[13] JAYANT, N. S., NOLL, P., *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice-Hall, 1984.

[14] CHOU, P. A., LOOKABAUGH, T., GRAY, R. M., "Entropy-constrained vector quantization", *IEEE Trans. Acoustics, Speech and Signal Processing*, v. 37, pp. 31 – 42, Janeiro 1989.

[15] HAYKIN, S., *Neural Networks and Learning Machines*. Prentice Hall, 2008.

[16] DUDA, R. O., HART, P. E., *Pattern Classification and Scene Analysis*. J. Wiley and Sons, 1973.

[17] SCHÖLKOPF, B., SMOLA, A., MÜLLER, K.-R., "Nonlinear component analysis as a kernel eigenvalue problem", *Neural Comput.*, v. 10, n. 5, pp. 1299–1319, July 1998.

[18] SATISH, D., SEKHAR, C., "Kernel based Clustering and Vector Quantization for Speech Segmentation", *Neural Networks, International Joint Conference on*, pp. 1636 – 1641, Julho 2006.

[19] YEH, C.-Y., LEE, S.-J., "A Kernel-Based Two-Stage Nu-Support Vector Clustering Algorithm", *Machine Learning and Cybernetics, International Conference on*, v. 4, pp. 2251 – 2256, Agosto 2007.

[20] LLOYD, S., "Least squares quantization in PCM", Bell Telephone Laboratories Paper, 1957.

[21] LINDE, Y., BUZO, A., GRAY, R., "An Algorithm for Vector Quantizer Design", *IEEE Trans. Communications*, v. 28, pp. 84 – 95, Janeiro 1980.

[22] KIRKPATRICK, S., GELATT, C. D., VECCHI, M. P., "Optimization by Simulated Annealing", *Science*, v. 220, pp. 671–680, Maio 1983.

[23] OLIVEIRA, F., GOMES, J., "Imageador CMOS utilizando tecnologia de 0.18um para captura e compressão de imagens no plano focal", *Dissertação de Mestrado - COPPE/UFRJ*, pp. 5 – 15, Dezembro 2013.

[24] SERACO, E., GOMES, J., "Computation of the complexity of vector quantizers by affine modeling", *Signal Processing*, v. 91, n. 5, pp. 1134 – 1142, Maio 2011.

[25] ELIAS, V., GOMES, J., "Estudo das Propriedades de Algoritmos de Quantização Vetorial baseados em Kernel PCA", *Projeto de Graduação - Escola Politécnica - UFRJ*, pp. 1 – 52, Agosto 2013.

[26] MAURICIO, L., GOMES, J., "Estudo e Comparação de Técnicas de Quantização para Compressão de Imagens", *Projeto de Graduação - Escola Politécnica - UFRJ*, pp. 1 – 50, Agosto 2014.

[27] ELIAS, V., GOMES, J., "Neural Network Design for Data Compression based on Kernel PCA: Rate-Distortion and Complexity Analysis", *Latin American Congress on Computation Inteligence*, pp. 1 – 6, Agosto 2015.