Trabalho de Conclusão de Curso

Classificação de Doenças Pulmonares Difusas obtidas via Tomografia de Alta-Resolução

Isadora Cardoso Pereira da Silva
isadora.cardoso@laccan.ufal.br

Orientador:

Prof. Dr. Heitor Soares Ramos Filho

Maceió, Setembro de 2017

Isadora Cardoso Pereira da Silva

Classificação de Doenças Pulmonares Difusas obtidas via Tomografia de Alta-Resolução

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas.

Orientador:

Prof. Dr. Heitor Soares Ramos Filho

Maceió, Setembro de 2017

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação do Instituto de Computação da Universidade Federal de Alagoas, aprovada pela comissão examinadora que abaixo assina.

Prof. Dr. Heitor Soares Ramos Filho - Orientador
Instituto de Computação
Universidade Federal de Alagoas

Prof. Dr. André Luiz Lins de Aquino - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Dr. Ivan César Martins - Examinador
Instituto de Computação
Universidade Federal de Alagoas

Maceió, Setembro de 2017

Agradecimentos

A Deus, por todas as bençãos diárias;

A minha família, em especial aos meus pais, Roseneide e Enaldo, e minhas irmãs, Christiane e Isabella, por todo o esforço para que eu possa ter uma boa educação, além da dedicação e paciência comigo – e, claro, minha paciência com vocês;

A todos os meus amigos que, apesar das batalhas da vida adulta, estão sempre comigo, me apoiando e me ajudando a crescer, de uma forma ou de outra;

A todos os membros do LaCCAN, por terem me aberto o mundo científico e me ensinarem o quão interessante pode ser a pesquisa.

"Who can say if I have been changed for the better? I do believe I have been changed for the better. And, because I knew you, I have been changed for good."

– Stephen Schwartz, *Wicked*

"Não se pode aprender nada de uma lição que não seja acompanhada da dor, já que não se pode obter nada sem um sacrifício. Mas quando você aguenta essa dor e a supera, as pessoas conseguem um coração forte que não perde para nada. Sim, um coração forte como aço."

– Hiromu Arakawa, *Full Metal Alchemist*

Resumo

Este trabalho apresenta um estudo de técnicas de aprendizado de máquina aplicados à análise de imagens médicas, relativas a doenças pulmonares difusas, obtidas via Tomografia Computadorizada de Alta-Resolução. Este estudo objetiva contribuir para o desenvolvimento de um Diagnóstico Auxiliado por Computador. Para isso, os seguintes padrões pulmonares foram avaliados: pneumonia, áreas enfisematosas, espessamento de septo, favo de mel, opacidade em vidro fosco e tecido pulmonar saudável. Dessas imagens, extraiu-se 28 atributos de textura de regiões de interesse previamente escolhidas e avaliadas por médicos especialistas. Utilizou-se as técnicas de redução de dimensionalidade Análise de Componentes Principais (PCA) e Análise de Discriminante Linear (LDA) e as técnicas de seleção de atributos Seleções *Stepwise – Forward* (SSF), *Backward* (SSB) e *Forward-Backward* (SSFB), para reduzir o número total de dimensões e assim evitar uma possível Maldição da Dimensionalidade. Esses atributos foram então classificados utilizando-se as seguintes técnicas de classificação: k-Vizinhos Mais Próximos (kNN), Máquina de Vetores de Suporte (SVM), Modelo de Mistura de Gaussiana (GMM) e Rede Neural Artificial (ANN) (em sua versão rasa e profunda). Utilizou-se também Redes Neurais Convolucionais (CNN), que, diferente das outras técnicas de classificação utilizadas, recebe diretamente as imagens como entrada, não os atributos extraídos. Obteve-se 99.6 % e 99.24 % de acurácia ao utilizar DFNN e CNN, respectivamente, o que foi considerado um resultado satisfatório. Esses resultados podem auxiliar na obtenção de diagnósticos mais precisos.

Palavras-chave: Aprendizado de Máquina, Diagnóstico Auxiliado por Computador, Doenças Pulmonares Intersticiais Difusas, Redes Neurais Convolucionais, Redes Neurais Profundas

Abstract

This work presents a study of Machine Learning techniques applied to the analysis of medical images, related to diffuse lung diseases, obtained through High-Resolution Tomography. This work aims to help in the development of a Computer-Aided Diagnosis. For this, the following lung patterns were evaluated: pneumonia, emphysematous areas, septal thickening, honeycomb, ground glass opacity and normal lung tissues. From these images, 28 texture features from regions of interest (previously selected and evaluated by especialists) were extracted. We applied dimensionality reduction techniques, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), and feature selection techniques, such as Stepwise Selection – Forward (SSF), Backward (SSB), and Forward-Backward (SSFB), in order to reduce the number of dimension and may avoid a possible Curse of Dimensionality. These extracted features were classified using the following classification techniques: k-Nearest Neighbors (kNN), Support Vector Machine (SVM), Gaussian Mixture Model (GMM), and Artificial Neural Network (ANN) (the shallow and deep version). Convolutional Neural Network (CNN) was used as well, but, differently from the others classification techniques used here, it receives directly the images as input, not the extracted features. An accuracy of 99.6 % and 99.24 % were obtained by DFNN and CNN, respectively, which is considered as a satisfactory result. These results may help to obtain more accurate diagnoses.

Keywords: Computer-Aided Diagnosis, Convolutional Neural Network, Deep Neural Network, Intertistial Diffuse Lung Diseases, Machine Learning

Conteúdo

Lista de Figuras

Lista de Tabelas

1

Introdução

1.1 Motivação

Dos 7.6 bilhões de habitantes da população mundial atual, cerca de 4 bilhões moram em regiões urbanas (United Nations and Social Affairs, 2017). Até 2030, projeta-se que mais 1 bilhão de pessoas passem a viver em centros urbanos, alcançando 5 bilhões de pessoas, e que essa população aumente para 6.4 bilhões até 2050 (Angel et al., 2011). Assim, com a previsão mundial de aproximadamente 9.8 bilhões de pessoas para 2050, em apenas 33 anos, a população urbana passará de cerca de 52 % para 65 % da população mundial.

Devido ao não planejamento desse crescimento urbano, aumenta-se a poluição, congestionamento, segregação, desmatamento, violência e muitas outras consequências desagradáveis. É preciso desenvolver soluções capazes de proporcionar os recursos essenciais à vida, como água, saneamento, energia, saúde e etc, de forma eficiente a todos. Essas soluções devem criar um balanço adequado entre o crescimento social, cultural e sustentável, especialmente dentro das áreas urbanas e, com isso, aumentar a qualidade de vida dos cidadãos. Para tal, utiliza-se diversas Tecnologias da Informação e Comunicação (TICs), visando a criação das chamadas Cidades Inteligentes (CIs).

Dados relacionados a saúde são de suma importância dentro do contexto de CIs, visto que obter uma saúde de qualidade, o que inclui diagnósticos rápidos e precisos, compõe os direitos dos cidadãos. Uma forma de melhorar os diagnósticos é combinar informações clínicas sobre o paciente com achados visuais em exames baseados em imagens. Dentre tais exames, destaca-se a Tomografia Computadorizada de Alta-Resolução (TCAR), uma vez que melhora significativamente a sensibilidade e a especificidade do diagnóstico clínico (Elicker et al., 2008), além de reduzir a exposição da estrutura torácica (Pereyra et al., 2014).

Não há dúvidas quanto a importância da utilização de imagens no auxílio a diagnósticos. Porém, imagens médicas são geralmente deterioradas por ruído provindos de diversas

fontes de interferência e outros fenômenos, o que afeta os processos de aquisição de dados (Bankman, 2000). Sistemas de Diagnóstico Auxiliado por Computador (*computer-aided diagnosis* - CAD) podem ajudar os médicos em seus diagnósticos, combinando informações provindas de especialistas com métodos de aprendizagem de máquina (*Machine Leaning* - ML).

Computadores apresentam diversas vantagens em relação aos humanos durante uma avaliação, tendo como exemplo: não sofrem de tédio, fadiga, e distrações; os resultados obtidos podem ser replicados em outras máquinas sem diferenças significativas, ou seja, seus resultados são reprodutíveis; pode-ser reunir o conhecimento de vários especialistas em uma única análise; entre outras. Porém, apesar das vantagens citadas, é importante ressaltar que os computadores servem apenas como um auxílio aos médicos, uma vez que, para realizar bons diagnósticos, os médicos utilizam informações clínicas que não seguem regras lógicas ou quantificadas, como histórico do paciente e sua família, situação sócio-econômica, exame físico, teste laboratorial, e muitas outras (Rangayyan, 2004).

Em imagens médicas, este trabalho lida com imagens de doenças pulmonares difusas (DPDs), um grupo de mais de 150 condições patológicas que causam disfunção respiratória (Pereyra et al., 2014). Essas doenças representam um grande desafio para o especialista, devido ao grande número de patologias, que possuem alta morbidade, sendo necessário para o profissional conhecer profundamente diversas áreas médicas para detectar algumas dessas doenças. Por isso, frequentemente, o diagnóstico das DPDs acontece tardiamente (Rubin et al., 2012). Neste trabalho serão analisados seis padrões pulmonares relativos a DPDs, a saber: pneumonia (*pulmonary consolidation* - PC), áreas enfisematosas (*emphysematous areas* - EA), espessamento de septo (*septal thickening* - ST), favo de mel (*honeycomb* - HC), opacidade em vidro fosco (*ground-glass opacities* - GG) e tecido pulmonar saudável (*normal lung tissues* - NL).

Há diversos estudos utilizando técnicas de ML para o auxílio no diagnóstico de diversas doenças, como Floresta Aleatória (*Random Forest* - RF) e Máxima Relevância e Mínima Redundância (*Maximum Relevancy and Minimum Redundancy* - mRMR) para Câncer Pulmonar (Cai et al., 2015); *K-means* e Máquina de Vetores de Suporte (*Support Vector Machine* - SVM) para Câncer de Mama (Zheng et al., 2014); e SVM para Doença de Parkinson e Paralisia Supranuclear Progressiva (Salvatore et al., 2014). Recentemente, Aprendizagem Profunda (*Deep Learning* - DL) vem contribuindo para diversas áreas com o avanço do estado-da-arte, como no reconhecimento de falas (Ravanelli et al., 2017) e objetos (He et al., 2016). Litjens et al. (2017) lista mais de 300 contribuições de DL para a análise de imagens médicas, com sua maioria aparecendo nos últimos anos.

1.2 Objetivo

O objetivo deste trabalho é classificar seis padrões relativos a DPDs utilizando técnicas de ML, visando subsidiar o desenvolvimento de um CAD.

1.3 Solução proposta

Dentre todas as técnicas de ML disponíveis, este trabalho utiliza as seguintes técnicas para classificação das DPDs: k-Vizinhos Mais Próximos (*k-Nearest Neighbors* - kNN), SVM, Modelo de Mistura de Gaussianas (*Gaussian Mixture Models* - GMM) e Rede Neural Artificial (*Artificial Neural Network* - ANN) – além de sua forma mais tradicional, duas especializações de ANN são também utilizadas: a Rede Neural Profunda *feedforward* (*Deep Feedforward Neural Network* - DFNN) e Rede Neural Convolucional (*Convolutional Neural Network* - CNN).

Porém, antes de aplicar as técnicas de ML citadas, aplicou-se as seguintes técnicas de redução de dimensionalidade e seleção de atributos nos atributos extraídos das imagens disponíveis, a fim de evitar uma possível Maldição da Dimensionalidade: Análise do Componente Principal (PCA), Análise do Discriminante Linear (LDA) e Seleções *Stepwise – Forward* (SSF), *Backward* (SSB) e *Forward-Backward* (SSFB).

1.4 Contribuições

As contribuições deste trabalho são:

- Uma compreensão de técnicas de classificação e redução de dimensionalidade aplicadas em DPDs;
- Uma comparação do desempenho de técnicas de ML ao utilizar redução de dimensionalidade para DPDs;
- O avanço do estado-da-arte na classificação dos padrões pulmonares estudados.

Note que essas contribuições podem ajudar na caracterização e seleção de atributos que melhor identificam os padrões de DPDs estudados. Consequentemente, isso pode ajudar no desenvolvimento de um CAD, levando a diagnósticos mais rápidos e precisos.

1.5 Estrutura do texto

Este trabalho se divide como segue: Capítulo 2 introduz alguns conceitos necessários para a descoberta do conhecimento em base de dados; Capítulo 3 apresenta as técnicas de redução de dimensionalidade e seleção de atributos utilizadas; Capítulo 4 apresenta as técnicas

de ML utilizadas para classificar os padrões pulmonares; Capítulo 5 apresenta os materiais utilizados neste trabalho, assim como os trabalhos correlatos; Capítulo 6 apresenta os resultados obtidos; e, finalmente, o Capítulo 7 apresenta as considerações finais, concluindo este trabalho.

2

Descoberta de Conhecimento em Base de dados

Com o avanço tecnológico, cada vez mais dados passam a ser coletados e armazenados, oriundos das mais diversas fontes. Porém, a simples recuperação dos dados armazenados não representa todo o potencial de informação que esses dados podem fornecer. A Descoberta de Conhecimento em Base de dados (*Knowledge Discovery in Databases* - KDD) é o processo que permite a extração não-trivial de conhecimento previamente desconhecido e potencialmente útil de uma base de dados (Adriaans and Zantinge, 1997). É constituído das seguintes etapas:

- **Pré-processamento**: os dados são selecionados e transformados para o formato mais adequado da análise. Inclui limpeza, fusão de diversas fontes, remoção de ruídos, etc;
- **Mineração dos dados**: algoritmos são utilizados para encontrar padrões, associações, e/ou anomalias relevantes. Utiliza conhecimentos de Estatística, ML, etc;
- **Pós-processamento**: os resultados da mineração são interpretados e avaliados de acordo com o problema proposto.

Estas etapas são abordadas nos capítulos 3, 4 e 6, respectivamente.

É importante notar que essa extração de informações úteis pode se tornar uma tarefa desafiadora, devido as limitações das ferramentas tradicionais de análise. Essa dificuldade pode ocorrer não só em consequência da quantidade massiva, mas também pela natureza não tradicional dos dados (mesmo em pequena quantidade), ou seja, dados constituídos de imagens, vídeos, tabelas etc. Dessa forma, torna-se necessário misturar os métodos tradicionais com sofisticados algoritmos (Tan et al., 2005). Para isso, técnicas provenientes da ML são largamente utilizadas.

2.1 Aprendizado de Máquina

ML é uma área de Inteligência Artificial (IA) que estuda algoritmos capazes de aprender a realizar atividades no geral, tornando a máquina capaz de se comportar de forma não programada. Esse aprendizado é feito através da observação dos dados, instruções explícitas ou experiências. Em geral, o aprendizado pode ser dividido em dois grandes grupos:

- **Supervisionado**: Para cada observação há uma resposta associada. Deseja-se encontrar um modelo que relacione a resposta com as observações de formar a caracterizar as observações atuais e prever as respostas de observações futuras. Inclui técnicas de classificação e regressão;
- **Não-supervisionado**: Tem-se as observações, porém nenhuma resposta associada. A aprendizagem se dá através do entendimento das relações observações e seus respectivos valores, nem sempre claras para humanos. Inclui as técnicas de agrupamento.

Ao associar estatística com ML, obtém-se novas ferramentas para a construção de modelos capazes de explorar a natureza de complexos conjuntos de dados, reconhecendo seus padrões. Essa associação vem se tornando mais importante ao longo dos anos, principalmente com o advento de análises de quantidade massiva de dados, isto é, *Big Data* (James et al., 2013). Em particular, essa quantidade massiva de dados pode levar a um outro problema, chamado de Maldição da dimensionalidade.

2.2 Maldição da Dimensionalidade

Quanto mais dimensões, mais escassos os dados podem se tornar no espaço em que encontram. Isso impacta tanto em uma aprendizagem supervisionada, visto que essa escassez pode significar que não há dados o suficiente para criar um modelo confiável, quanto em uma não supervisionada, uma vez que a densidade e a distância entre os pontos, medidas de grande importância, se tornam pouco significativas para o agrupamento.

Nos dois casos, tem-se como consequência o aumento da dificuldade das análises, devido a pouca acurácia e baixa qualidade dos resultados obtidos, além da possibilidade de um *overfitting* e resultados falsos (Tan et al., 2005). A este fenômeno é dado o nome de Maldição da Dimensionalidade.

As alternativas mais utilizadas na atenuação da maldição da dimensionalidade são a redução da dimensionalidade e a seleção de atributos.

2.3 Redução da Dimensionalidade e Seleção de Atributos

Embora possuam critérios diferentes, a redução da dimensionalidade e a seleção de atributos possuem o mesmo objetivo: conter o máximo possível de informação relevante para

posterior análise, enquanto diminui a dimensão dos dados original.

A redução de dimensionalidade é utilizada para se obter novos atributos a partir da combinação dos atributos originais, transformando o espaço dos dados. Isso pode prover uma variedade de benefícios. Um deles se refere a capacidade de simplificação dos dados, que por unir as informações mais relevantes em um número menor de dimensões, conduz a um melhor entendimento e também ajuda na visualização dos dados.

Já a seleção de atributos é utilizada para a obtenção de um subconjunto dos dados originais, descartando aquelas dimensões irrelevantes e/ou redundantes.

Atributos redundantes são aqueles que, por duplicarem uma parte ou toda a informação contida em uma ou mais variáveis, podem criar uma tendência não desejada nos dados. Atributos irrelevantes são aqueles que contêm informações quase inúteis para a mineração dos dados. Esses dois tipos de atributos podem reduzir a acurácia e a qualidade da aprendizagem desejada.

A diminuição de dimensões contribui para a redução na quantidade de tempo e memória requerida no processamento dos dados. E, além de evitar a maldição da dimensionalidade, também pode melhorar o funcionamento de algoritmos de mineração de dados, que lidam melhor com pequenas dimensões, devido à redução de ruídos (Tan et al., 2005).

Após a utilização de técnicas de mineração de dados, a visualização dos resultados é primordial para saber se o algoritmo está se comportando da maneira esperada. Para isso, métricas de avaliação de desempenho são utilizadas.

2.4 Métricas de avaliação de desempenho

Em problemas de múltiplas classes, as métricas de desempenho podem ser calculadas de forma separada para cada classe, em um esquema um-contra-todos, o que chamamos de métricas *micro-averaged*, assim como também podem ser calculadas como uma média não ponderada de todas as classes, chamadas de métricas *macro-averaged*. Porém, enquanto métricas *macro-averaged* tratam todas as classes igualmente, métricas *micro-averaged* favorecem classes maiores. O escolha de qual tipo de métricas utilizar depende do objetivo do problema (Sokolova and Lapalme, 2009).

A forma mais simples de avaliar o desempenho do modelo de classificação é através da matriz de confusão, que expõe as predições realizadas *versus* as classificações anteriormente conhecidas. É uma matriz quadrada do tamanho do número de classes, isto é, $L \times L$, sendo L o número de classes.

Embora a matriz de confusão agrupe informações o suficiente para determinar o quão bem o algoritmo de classificação se comporta, reduzir essa informação para um simples número pode ser mais conveniente ao se comparar o desempenho de diversos algoritmos. Isso pode ser feito utilizando diversas métricas de desempenho, como a acurácia, definida

por:

$$\text{Acc} = \frac{\text{Predições corretas}}{\text{Total de observações}} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}} \tag{2.1}$$

Onde:

- TP (*True Positive* - Verdadeiro Positivo) representa os casos onde é predito que as observações pertencem a determinada classe e, de fato, estas pertencem;
- FP (*False Positive* - Falso Positivo) representa os casos onde é predito que as observações pertencem a determinada classe, mas na verdade não pertencem (também conhecido como erro do Tipo I);
- TN (*True Negative* - Verdadeiro Negativo) representa os casos onde é predito que as observações não pertencem a determinada classe, e, de fato, estas não pertencem;
- e, finalmente, FN (*False Negative* - Falso Negativo) representa os casos onde é predito que as observações não pertencem a determinada classe, mas na verdade estas pertencem (também conhecido como erro do Tipo II) (Tan et al., 2005).

Em especial, para problemas envolvendo dados médicos, é comum utilizar as métricas especificidade, sensibilidade e precisão.

Especificidade (Esp) descreve o quão efetivamente o classificador identifica predições negativas. Isto é, a habilidade de um modelo excluir corretamente os indivíduos que não possuem uma determinada doença. É definida por:

$$\text{Esp} = \frac{\sum_{i=1}^{L} \frac{\text{TN}_i}{\text{TN}_i + \text{FP}_i}}{\text{L}} \tag{2.2}$$

Sensibilidade (Sen) explica o quão efetivamente o classificador identifica as predições positivas. Ou seja, a habilidade de um modelo identificar corretamente quais indivíduos possuem uma determinada doença. É definida por:

$$\text{Sen} = \frac{\sum_{i=1}^{L} \frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}}{\text{L}} \tag{2.3}$$

Precisão (Pre) representa a variação aleatória de um modelo, assim, quanto menor, melhor. É definida por:

$$\text{Pre} = \frac{\sum_{i=1}^{L} \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i}}{\text{L}} \tag{2.4}$$

Ademais, pode-se utilizar a métrica F1-score, que é a média harmônica entre precisão e sensibilidade, sendo definida por:

$$\mathrm{F1} = 2 \times \frac{\mathrm{Pre} \times \mathrm{Sen}}{\mathrm{Pre} + \mathrm{Pre}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{2.5}$$

3

Pré-processamento

Neste trabalho utilizou-se algumas das técnicas consagradas de redução de dimensionalidade (PCA e LDA) e seleção de atributos (SSF, SSB, e SSFB) durante o pré-processamento dos dados, que estão descritas a seguir.

3.1 Análise de Componente Principal (PCA)

PCA é uma técnica não-supervisionada com o objetivo de encontrar um novo conjunto de d dimensões que melhor capture a variabilidade dos dados. Essas dimensões são chamadas de componentes principais (PC).

Pode-se entender PCA como um problema de otimização, onde é necessário maximizar a variância de cada componente principal d_{PC}. Assim:

$$\max_{d_{PC}} J(d_{PC}) = d_{PC}^{T} \Sigma_{PCA} d_{PC} - \alpha(d_{PC}^{T} d_{PC} - 1) \tag{3.1}$$

Onde Σ_{PCA} representa a matriz de covariância dos dados. Sendo D um conjunto de dados, $D = n \times d$, com n sendo o número de observações e d o número de dimensões. Centralizando este conjunto, ou seja, fazendo sua média $\mu_D = 0$, definimos covariância por $\Sigma_{PCA} = D^{T} D$.

A covariância é utilizada pois, para descrever a variância de dados multivariados, o desvio padrão e a variância não podem ser utilizados, por descreverem somente uma única dimensão por vez. Covariância é a medida do quanto uma dimensão varia em relação a outra (Tan et al., 2005).

Ao derivar a equação 3.1 em função de d_{PC}, obtemos $\Sigma_{PCA} d_{PC} = \alpha d_{PC}$, que implica que α é um autovalor da matriz de covariância Σ_{PCA}, com seu respectivo autovetor d_{PC}.

Autovetores são k_a vetores, em matrizes $k_a \times k_a$, que, quando a matriz sofre uma transformação, podem mudar sua magnitude, mas não mudam sua direção. Por exemplo,

quando aumentamos uma imagem, os autovetores são aqueles que preservam o aspecto desta. Autovetores possuem as propriedades de serem sempre diferentes de zero e ortogonais entre si. Nesse caso, a projeção representa uma transformação. Os autovalores são os valores associados a magnitude dos autovetores, ou seja, seu tamanho (Zaki and Wagner Meira, 2014).

O maior autovalor obtido indica a direção com maior variância, e seu autovetor representa a componente principal. O mesmo pensamento é seguido para extrair todas as d_{PC} componentes principais. Cada uma das dimensões encontradas pelo PCA são uma combinação linear dos d atributos do conjunto D.

PCA apresenta diversas vantagens, como por exemplo:

- PCA identifica o padrão mais forte dos dados – logo, PCA pode ser utilizado para encontrar padrões;
- Geralmente, a maioria da variância pode ser capturada por uma pequena porção da dimensão dos dados. Assim, o conjunto de dados obtido por PCA é relativamente pequeno;
- Como geralmente os ruídos nos dados são mais fracos que os padrões, PCA pode ajudar a diminuir o ruído dos dados (Tan et al., 2005).

3.2 Análise do Discriminante Linear (LDA)

Apesar de vantajoso em diversos casos, se tivermos duas classes de dados dispostas de maneira paralela, PCA irá escolher uma componente principal que esteja na direção das classes, juntando-as. É notável que essa escolha acarretará erros de classificação. Nessas situações, LDA é uma melhor alternativa.

LDA é um algoritmo supervisionado com o objetivo de diminuir a dimensão preservando tanto quanto possível a informação discriminatória das classes. Assim, é escolhido o hiperplano (um subplano com $d_{LD} - 1$ dimensões) que melhor separe os dados de acordo com as classes as quais pertencem. Cada vetor do hiperplano é chamado de discriminante linear ideal (LD).

LDA leva em conta não só a máxima separação entre as classes, mas também a variância dentro das classes, que não deve ser muito grande. Uma variância grande possibilitaria sobreposições entre os pontos das classes devido ao grande espalhamento, o que pode piorar a separação. Em outras palavras, LDA assegura uma dispersão pequena entre os pontos dentro de cada classe ao mesmo tempo que maximiza a separação das classes.

Dessa forma, há dois pontos fundamentais sobre LDA: deve-se encontrar a direção que maximiza a separação entre as médias e miniminizar a dispersão das classes.

Para a separação das médias entre as classes 1 e 2 temos $B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$, onde B é uma $d_{LD} \times d_{LD}$ matriz *rank-one* (cada coluna é múltipla da primeira) chamada matriz de dispersão entre classes (***between-class scatter matrix***). A separação das médias de todas as classes é análoga.

Para a dispersão projetada das classes c_i, temos $s_i^2 = d_{LD}^T S_i d_{LD}$. e S_i é a matriz de dispersão para c_i. A soma das matrizes de dispersão $S_i = S_1 + ... + S_n$ é a matriz de dispersão dentro da classe (***within-class scatter matrix***).

Semelhante a PCA, LDA pode ser entendido como um problema de otimização. Seu objetivo é maximizar a separação das médias e minimizar a dispersão das classes. Então, sua função objetivo é:

$$\max_{d_{LD}} J(d_{LD}) = \frac{d_{LD}^T B d_{LD}}{d_{LD}^T S d_{LD}} \tag{3.2}$$

Na qual devemos encontrar a melhor direção. Derivando em função de d_{LD} obtemos:

$$B d_{LD} = \lambda S d_{LD} \tag{3.3}$$

Onde $\lambda = J(d_{LD})$.

A equação 3.3 representa o problema do autovalor generalizado e corresponde a um autovalor generalizado de B e S (o vetor λ que satisfaz a equação $\det(B - \lambda S) = 0$). Deve-se encontrar o maior autovalor generalizado λ, com d_{LD} correspondendo ao autovetor associado. Se S é *nonsingular*, isto é, S^{-1} existe, então a equação 3.3 corresponde à equação autovalor-autovetor regular, $(S^{-1}B)d_{LD} = \lambda d_{LD}$ e assim $\lambda = J(d_{LD})$ será um autovalor e d_{LD} um autovetor da matriz $(S^{-1}B)$.

Encontrando o maior autovalor λ, encontramos o autovetor associado, que é o melhor vetor LD. Os LD seguintes são encontrados de maneira semelhante, sendo ortogonal ao vetor LD anterior (Zaki and Wagner Meira, 2014).

3.3 Seleções Stepwise

Um dos lados negativos das técnicas de redução de dimensionalidade que transformam o espaço dos dados é a dificuldade de visualização que esse novo conjunto possui. Apesar de possuir bons resultados, no fim são utilizados todos os atributos, tal qual o conjunto original. Por conseguinte, essa é uma das vantagens de se utilizar a seleção de atributos, uma vez que será obtido um menor conjunto de dados que o original. Esse subconjunto também possuirá bons resultados, já que o que é descartado são as possíveis redundâncias e insignificâncias presentes nos atributos, o que ajuda a evitar o *overfitting*.

Para obter o melhor subconjunto, o ideal é tentar todas as combinações possíveis. Porém, uma vez que o número de conjuntos obtidos envolvendo n atributos é 2^n, essa aborda-

gem é impraticável na maioria das situações. Assim, as seleções *stepwise* utilizam critérios de penalidade para os parâmetros do modelo. Neste trabalho utiliza-se o *Bayesian Information Criterion* (BIC), definido por:

$$\mathrm{BIC} = \ln(d)k - 2\ln(\widehat{L})$$

Onde $\widehat{L}$ é o valor maximizado da máxima verossimilhança do modelo, d o número de observações no conjunto de dados e k é o número de parâmetros livres a ser estimado (Tan et al., 2005).

Há três formas básicas de seleção *stepwise*, as quais foram adotadas neste trabalho e são descritas a seguir.

3.3.1 Forward (SSF)

Como pode ser visto na figura 3.1, SSF inicia com um modelo vazio e, testando todas as possibilidades sequencialmente, adiciona no modelo o atributo que mais adicionar ao ganho de informação total do subconjunto. Esse passo se repete até que não seja possível ganhar mais informação adicionando variáveis ou a alteração do ganho seja praticamente nula.

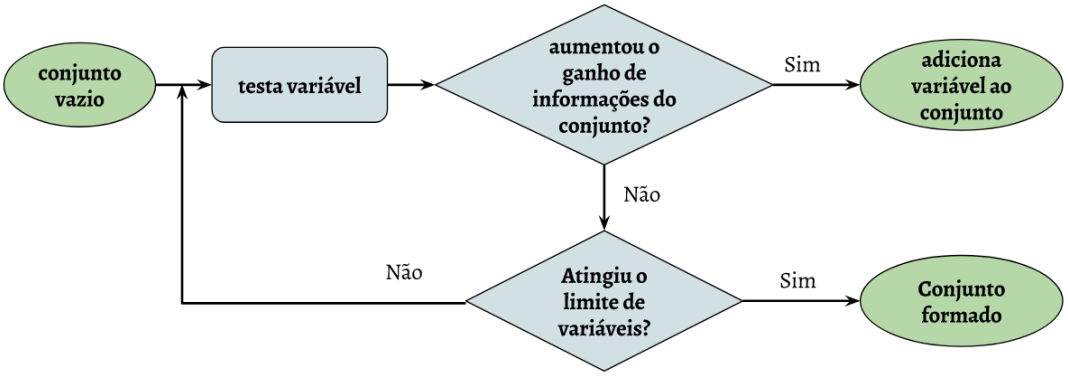


Figura 3.1: Fluxograma do algoritmo SSF

3.3.2 Backward (SSB)

Ao contrário do SSF, SSB inicia com um modelo contendo todos os atributos e, sequencialmente, testa todas as possibilidades de exclusão para encontrar o atributo que tiver o menor ganho de informação, que é retirado. Esses passos são repetidos até que não seja possível tirar mais nenhum, de acordo com um limite de corte. O fluxograma do algoritmo pode ser visto na figura 3.2.

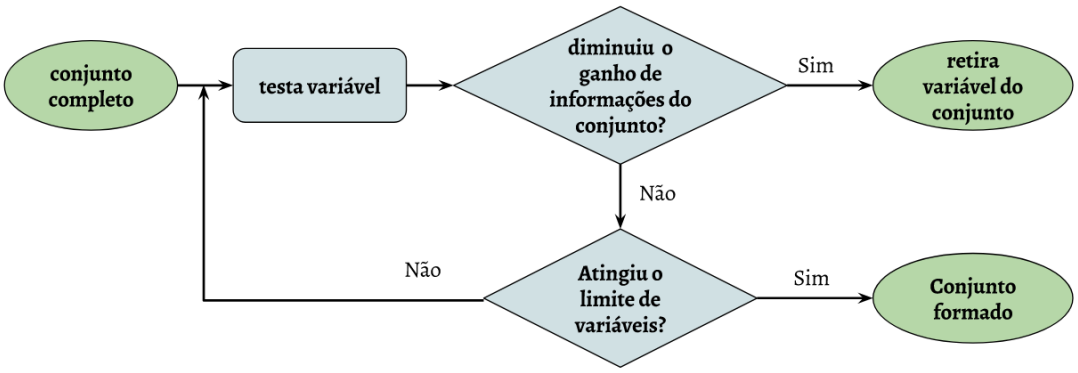


Figura 3.2: Fluxograma do algoritmo SSB

3.3.3 Forward-Backward (SSFB)

SSF e SSB são técnicas gulosas, isto é, a partir do momento que um atributo é adicionado ou retirado, ele não pode mais voltar, o que pode deixar escapar o melhor subconjunto. Assim, SSFB é uma combinação das duas técnicas. Como mostrado na figura 3.3, após adicionar uma variável ao subconjunto (como em SSF), checa-se todos os atributos no modelo a fim de verificar se a significância das mesmas foram alteradas – caso encontre, retira-a (como em SSB) (Hastie et al., 2003).

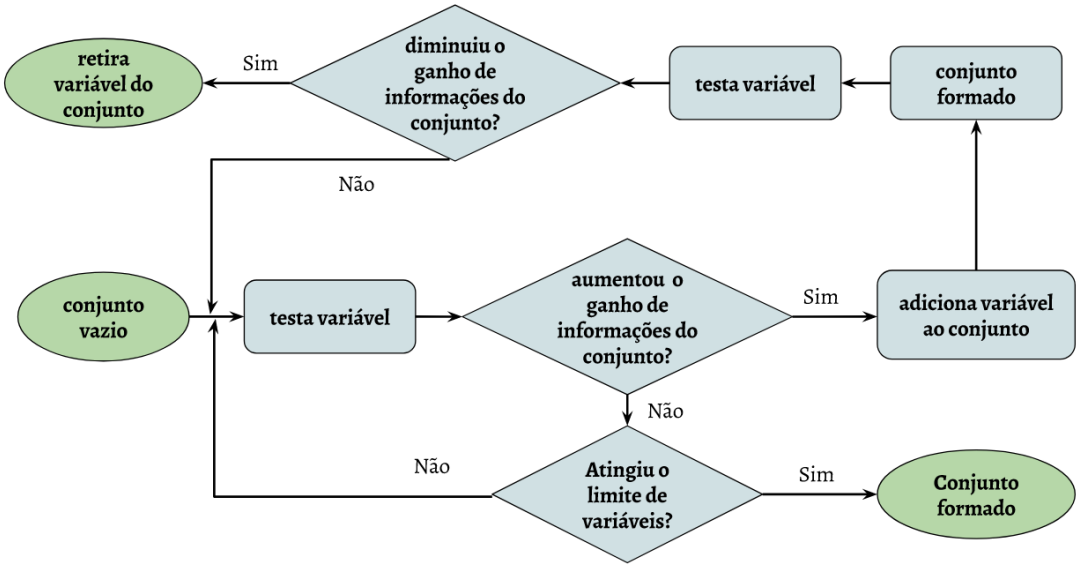


Figura 3.3: Fluxograma do algoritmo SSFB

4

Classificação

4.1 Introdução

Classificação, constituindo o aprendizado supervisionado, é a atividade de aprender uma função objetivo $f : x \rightarrow y$ – também chamada de modelo de classificação – que mapeia cada observação do conjunto de dados x para uma das classes predefinidas y. É útil para caracterizar (enquanto uma ferramenta explanatória para distinguir entre objetos de diferentes classes) e predizer (enquanto uma ferramenta para conjecturar a classe de observações com rótulo desconhecido).

Difere-se da regressão por possuir suas classes y representadas apenas por valores discretos, enquanto regressão utiliza valores contínuos (Tan et al., 2005).

A classificação pode ser utilizada em diversos campos de pesquisa para estudar diversos acontecimentos, como reconhecimento de emoções em falas (Basu et al., 2017), mineração de textos (Brindha et al., 2016), entre muitos outros.

As técnicas de classificação utilizadas neste trabalho estão descritas a seguir.

4.2 k-Vizinhos Mais Próximos (kNN)

Algumas técnicas necessitam construir um modelo previamente para que possam realizar uma classificação. Embora a classificação em si seja muito rápida, a construção do modelo pode demorar muito, podendo até inviabilizar o uso da técnica. Porém, técnicas como kNN, no lugar da construção prévia do modelo, esperam até receber os casos de testes para classificá-los em função da sua similaridade com as observações obtidas previamente. Essa é uma abordagem *lazy*, visto que não há a necessidade de manter uma abstração dos dados. Entende-se a não construção de um modelo como uma vantagem em termos de velocidade e simplicidade, porém, para cada caso de teste, necessita-se treinar o modelo.

O kNN representa cada observação dos dados como um ponto em um espaço d-dimensional, sendo d o número de atributos. Ao receber um caso de teste, o algoritmo computa sua proximidade para o resto dos pontos, geralmente usando a distância Euclidiana:

$$Dist(a,b) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2} \tag{4.1}$$

A classificação do caso de teste se relaciona com o rótulo dos seus k vizinhos mais próximos por maioria de votos, isto é, sua classificação será aquela a qual a maioria de seus k vizinhos pertencem. Todos os vizinhos possuem o mesmo peso decisivo, portanto, a escolha da quantidade de vizinhos influencia na decisão (Tan et al., 2005).

4.3 Modelo de Mistura de Gaussianas (GMM)

GMM é um modelo probabilístico de clusterização onde as observações dos dados possuem uma probabilidade de pertencer a cada grupo, uma vez que não há garantias da classificação para cada ponto.

Seja D o conjunto de dados de n pontos x_j em um espaço d dimensional, $\mathbb{R}^d$. Cada grupo C_g da clusterização é representado como uma mistura de distribuições Gaussianas, isto é:

$$f_g(x) = f(x|\mu_g, \Sigma_g) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_g|^{\frac{1}{2}}} \exp\left[-\frac{(x-\mu_g)^T\Sigma_g^{-1}(x-\mu_g)}{2}\right] \tag{4.2}$$

Onde a média $\mu_g \in \mathbb{R}^d$ e a matriz de covariância $\Sigma_g \in \mathbb{R}^{d\times d}$ são parâmetros desconhecidos. $f_g(x)$ é a densidade de probabilidade em x atribuível ao grupo C_g. Então, assume-se que a função de densidade de probabilidade (FDP) dos dados é uma mistura de gaussianas com k_c grupos, sendo definida por:

$$f_{gm}(x) = \sum_{g=1}^{k_c} f_g(x)P(C_g) = \sum_{g=1}^{k_c} f(x|\mu_g, \Sigma_g)P(C_g) \tag{4.3}$$

Onde as probabilidades *apriori* $P(C_g)$ são chamados de parâmetros de misturas, que devem satisfazer a condição $\sum_{g=1}^{k} P(C_g) = 1$.

GMM é então um modelo caracterizado pela média μ_g, a matriz de covariância Σ_g e a mistura de probabilidades $P(C_g)$ de cada k_c distribuição:

$$\theta_g = \{\mu_1, \Sigma_1, P(C_1) \ldots \mu_{k_c}, \Sigma_{k_c}, P(C_{k_c})\} \tag{4.4}$$

Para estimar esses parâmetros θ_g do modelo, deve-se escolher θ_g de forma a maximizar a verossimilhança.

A verossimilhança de θ_g é a probabilidade condicional do conjunto D dado os parâ-

metros θ_g do modelo, denotado como $P(D|\theta_g)$. Considerando cada n ponto x_j como uma amostra aleatória de X, a verossimilhança é:

$$L(\theta_g) = P(D|\theta_g) = \prod_{j=1}^{n} f(x_j) \tag{4.5}$$

Maximizar a verossimilhança dos dados para diferentes valores dos parâmetros θ_g é uma atividade complicada, pois não sabemos qual ponto pertence a qual distribuição. Dessa forma, utilizamos o algoritmo *Expectation-Maximization* (EM) (Zaki and Wagner Meira, 2014).

EM, descrito em Dempster et al. (1977), é constituído de dois passos: i) O *E-step*, que computa a probabilidade de cada ponto pertencer a cada Gaussiana; ii) o *M-step*, que utiliza a probabilidade do passo anterior para maximizar a esperada verossimilhança, e assim, encontrar os parâmetros θ_g. Ambos são repetidos sequencialmente até que a função convirja, isto é, até que não haja mais mudanças no valor de verossimilhança ou tais mudanças sejam insignificantes (Almeida et al., 2015b).

Como uma extensão, Fraley and Raftery (2000) propõem a utilização dos rótulos das observações para a determinação da melhor parametrização da matriz de covariância Σ_{k_c}, da forma do *cluster* e do número de misturas de Gaussianas para cada classe, transformando GMM em um algoritmo supervisionado. Essa abordagem se mostrou mais eficiente neste trabalho.

4.4 Máquinas de Vetores de Suporte (SVM)

SVM é uma técnica supervisionada de classificação que tem por objetivo encontrar um hiperplano que maximize a separação entre as classes (Tan et al., 2005). Pode-se definir um hiperplano em d dimensões como:

$$h(x) = w^T x + b$$

Onde $x \in \mathbb{R}^d$ e representa os pontos do conjunto de dados, w é um vetor d dimensional de pesos e b um escalar, chamado *bias*. É importante que o hiperplano não só maximize a separação entre as classes, mas também que a margem entre as classes e o hiperplano correspondente seja tão ampla quanto possível. A margem pode ser definida como a distância mínima de um ponto para o hiperplano, descrita como:

$$\delta = \frac{y \times h(x)}{\|w\|} = \frac{y(w^T x + b)}{\|w\|} \tag{4.6}$$

Onde y é a classe previamente conhecida ao qual o ponto x pertence. Assim, o objetivo do SVM pode ser interpretado como uma maximização da margem, o que também pode ser

interpretado como a minimização de $\|w\|$.

Em casos nos quais as classes não possam ser perfeitamente separáveis, aqueles pontos classificados erroneamente, isto é, aqueles que se encontram do lado errado quando comparado à classe que pertence, recebem uma penalidade, gerando a seguinte função objetivo:

$$\min_{w,b,\xi} \frac{\|w\|^2}{2} + C \sum_{i=1}^{n} (\xi_i)^k \tag{4.7}$$

Onde C_s e k_s são constantes que incorporam o custo da má classificação. O termo $\sum_{i=1}^{n} (\xi_i)_s^k$ descreve a perda, que é uma estimativa do desvio do caso onde as classes são perfeitamente separáveis. C_s é escolhido empiricamente e é um termo de regularização: quando $C_s \to 0$, o termo de perda praticamente desaparece, sobrando apenas a maximização da margem; quando $C_s \to \infty$, o objetivo praticamente passa a ser minimizar as perdas (Zaki and Wagner Meira, 2014).

É importante notar que, quanto maior a margem, menos flexibilidade tem-se no modelo. Porém, margens pequenas são mais suscetíveis à *overffiting* e pobre generalização, apesar de sua grande adaptabilidade à modelos diversos. Por essa razão, é importante encontrar modelos que forneçam um balanço adequado entre a distância da margem e o erro de generalização.

Os vetores de suporte são aqueles pontos do conjunto de dados que estão na distância aceita entre a margem e o hiperplano. Os pontos que não são vetores de suporte, ou seja, estão distantes demais para influenciar a margem, podem ser desconsiderados. Por essa razão, esta técnica funciona muito bem com dados de grande dimensionalidade e evita a maldição da dimensionalidade (Tan et al., 2005).

É possível aplicar o SVM em dados que não são separados por funções lineares utilizando o *kernel trick*, que consiste em transformar os dados das coordenadas originais para outro espaço, e assim utilizar uma função linear para separar as classes (Zaki and Wagner Meira, 2014).

4.5 Rede Neural Artificial (ANN)

Sabe-se que cérebro humano é composto basicamente de células nervosas, chamadas neurônios, que conectam-se entre si através dos axônios. Um axônio é a parte do neurônio responsável por transmitir o impulso nervoso de um neurônio a outro quando são estimulados. Um neurônio é ligado ao axônio de outros neurônios através de dendritos, que são extensões do corpo celular do neurônio. O ponto de contato entre um dendrito e um axônio é chamado de sinapse. Neurologistas descobriram que o cérebro humano aprende mudando a resistência da ligação sináptica entre os neurônios sob repetida estimulação pelo mesmo impulso (Tan et al., 2005). Além disso, o cérebro está organizado em diversas camadas e

possui um massivo paralelismo, além de alta distribuição na entrega de suas mensagens, o que compensa a baixa velocidade utilizada – cerca de $100\mathrm{m/s}$ – e a baixa largura de banda – por volta de $100\mathrm{bits/s}$ (Ripley and Hjort, 1995). São essas ligações que, entre outras coisas, tornam o ser humano extremamente eficaz no reconhecimento de padrões e generalizações. Na tentativa de simular essa impressionante capacidade do sistema neural biológico, desenvolveu-se as redes neurais artificiais (ANN).

De forma semelhante ao cérebro humano, uma ANN é composta de um conjunto interligado de nós e ligações direcionadas. Esses conjuntos podem formar três tipos de camadas, que são:

- A camada de entrada, formada por nós de entrada, que representam os atributos de entrada;
- As camadas intermediárias, chamadas de camadas ocultas (*hidden layers*), com seus nós, chamados de nós ocultos (*hidden nodes*);
- E a camada de saída, formada por nós de saída, que representam o modelo obtido (ou o quanto de classes estão sendo avaliadas).

Cada nó de uma camada é totalmente conectado a todos nós da camada seguinte (*fully-connected*) por uma ligação ponderada, usada para simular a resistência das conexões sinápticas entre os neurônios. Ao possuir camadas ocultas, a ANN é chamada de ANN de multicamada.

É possível que, em uma ANN, os nós de uma camada conectem-se entre si (ANN recorrente). Outra possibilidade é que os nós de uma camada se conectem apenas com os nós da próxima camada, formando uma cadeia – modelo conhecido como ANN *feedforward* (Tan et al., 2005).

A ANN *feedforward* de uma camada oculta é definida como segue:

$$y_{nn} = f_{nn}(\alpha_k + \sum_{j=1}^{k} w_{jk} f_j(\alpha_j + \sum_{i=1}^{j} w_{ij} x_i)) \tag{4.8}$$

Onde f_{nn} representa a função de ativação, α o viés, w_{nn} a ponderação (ou pesos das conexões) e x_{nn} os valores de entrada. Em cada camada, o termo de viés pode ser retirado ao se adicionar um nó com valor $+1$, ligado a todos os outros nós da camada (Ripley and Hjort, 1995).

Na figura 4.1 vemos uma representação da arquitetura de uma ANN de uma camada oculta *feedforward*.

Há certas vantagens nas ANN de única camada oculta. O teorema da aproximação universal diz que redes neurais com nós de saída linear e uma única camada oculta podem representar praticamente qualquer função f contínua ao aumentar o tamanho da camada

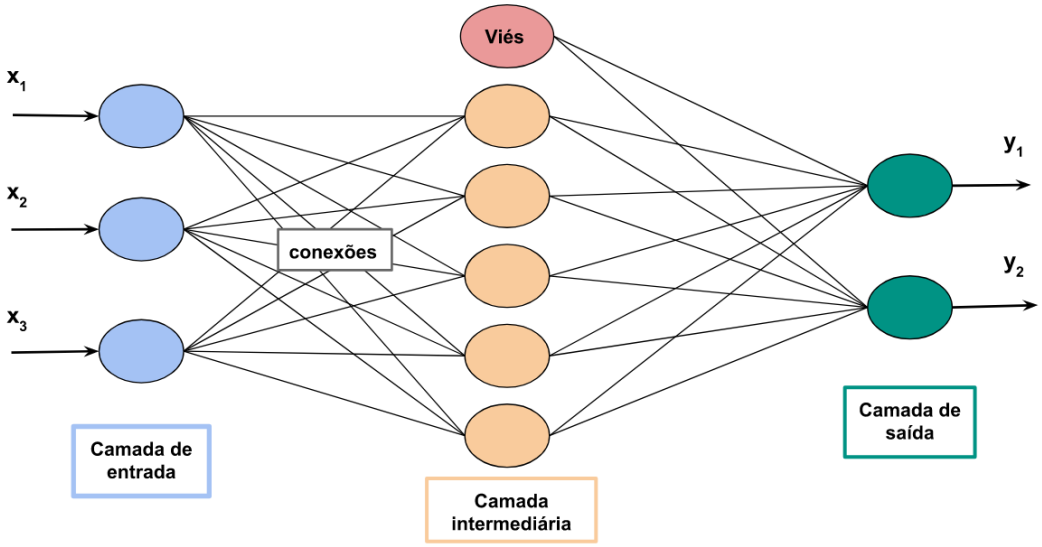


Figura 4.1: Representação de uma ANN

oculta Ripley and Hjort (1995) – mas não há garantias de que o algoritmo será capaz de aprender essa função (Goodfellow et al., 2016). Embora não se tenha guias para o tamanho ideal dessa camada em problemas práticos, ANN *feedforward* podem funcionar bem com um número modesto de nós, devido a transformação em um subespaço de menor dimensionalidade que o primeiro estágio da rede neural sofre Ripley and Hjort (1995).

É importante notar que os nós de entrada apenas transmitem seus valores para as ligações, ou seja, apenas distribuem os atributos de entrada, sem realizar qualquer transformação – isto ocorre apenas nas outras camadas. Por outro lado, os nós das outras camadas são dispositivos matemáticos que computam a soma ponderada das entradas recebidas, subtraem o termo de viés, e então produzem uma saída que será aplicada na função de ativação adotada.

Vários tipos de função de ativação podem ser utilizados, como a função *heaviside*, a linear, a logística, entre outras. Geralmente, adota-se uma função logística, pois esta permite aos nós da camada oculta e aos de saída produzir valores para suas saídas que são não-lineares aos seus parâmetros de entrada (Tan et al., 2005). Porém, sua aplicação serve apenas para problemas binários. Para problemas multi-classes, podemos adotar uma generalização, conhecida como *softmax*:

$$f(x) = p(k_{nn}|x) = \frac{e^{y_{k_{nn}}}}{\sum_{j=1}^{k_{nn}} e^{y_j}} \tag{4.9}$$

Onde k_{nn} representa o número de classes e $y_{k_{nn}}$ a saída da rede neural com nós de saída lineares. Essas ANN com mais de uma classe de saída corresponde à regressões logísticas de uma classe *versus* todas as outras (Ripley and Hjort, 1995).

Uma outra adaptação da função logística comumente adotada é a função tangente hiper-

bólica. Essa função é uma função logística redimensionada e deslocada para o ponto $(0,0)$. Essa simetria ao redor de 0 permite a rede neural convergir mais rapidamente durante o treinamento Candel et al. (2017). Define-se a função tangente hiperbólica por:

$$\tanh(\beta) = \frac{e^{\beta} - e^{-\beta}}{e^{\beta} + e^{-\beta}} \tag{4.10}$$

Onde β é a combinação dos pesos das ligações, denotado por $\beta = \sum_i w_i x_i + b$, sendo x_i e w_i o valor de entrada do neurônio e a ponderação, respectivamente.

Para o treinamento adequado da rede neural, é preciso adaptar os pesos das ligações até que se estabeleça relações de entrada-saída com o mínimo de erro possível. Ou seja, deseja-se otimizar a soma dos quadrados, definido por:

$$E(w) = \sum_j |y_j - \hat{y}_j|^2 \tag{4.11}$$

Onde y_j é o valor conhecido do caso de teste e $\hat{y}_j$ o valor predito pela ANN.

Para resolver esse problema de otimização, pode-se adotar técnicas como a de retro-propagação (*back-propagation*). A retro-propagação utiliza algoritmos gulosos, como *gradient descent*, para otimizar a função 4.11. Após, é dividida em duas fases iterativas: i) na fase de avanço (*forward*), os pesos das ligações na camada j são utilizados para computar a saída dos nós na camada $j+1$; ii) na fase reversa (*backward*), a atualização é obtida primeiramente no nível mais próximo da camada de saída, e vai se propagando do nível $j+1$ para o nível j (Tan et al., 2005). Cada vez que essas ponderações são atualizados para todas as observações disponíveis para treino, é marcada uma época.

Nessa atualização da ponderação, as ligações que contribuem mais para o erro são as que requerem maior ajuste. Porém, os erros não devem ser mudados tão drasticamente, uma vez que o erro é computado apenas para a observação atual. Caso contrário, o ajuste feito anteriormente em iterações anteriores serão desfeitos. A taxa de aprendizado λ, um parâmetro com valores entre 0 e 1, pode ser usado para controlar a quantidade de ajuste feita em cada iteração:

- Se λ é próximo de 0, então o novo peso é mais influenciado por valores anteriores da ponderação;
- Se λ é próximo de 1, então o novo valor é sensível a quantidade de ajuste feito na iteração atual.

Como é muito difícil ajustar esse parâmetro, um λ adaptável pode ser usado em alguns casos: inicialmente, λ é moderadamente grande durante umas primeiras poucas iterações e então gradualmente decresce nas iterações subsequentes (Ripley and Hjort, 1995). Métodos de aprendizado adaptativo apresentam a vantagem de não precisarem ser explicitamente ajustados.

É importante notar que a utilização da retro-propagação não é trivial quando se tem camadas ocultas, já que não é tão simples acessar os termos de erro para cada nó nessas condições. Além disso, utilizar o *gradient descent* para otimização leva à uma pobre generalização, na maioria dos casos. Mais ainda, essa abordagem costuma ser muito lenta para o treinamento de grandes ANN. Por essas razões, é comum a adoção de outros algoritmos, como os métodos *Quasi-Newton*, que apesar de geralmente convergirem super linearmente, são efetivos para evitar mínimos locais.

Para ajudar na otimização, abordagens como decaimento do peso (*decay weight*) são utilizadas. Essa abordagem consiste em adicionar um termo de decaimento à função de otimização, tornando o peso sempre pequeno e assim reduzindo o encontro de mínimos locais na função, além de evitar o *overtfitting* e ajudar na rapidez com a qual a função converge (Ripley and Hjort, 1995).

Métodos de regularização podem ser utilizados para reduzir os erros de generalização e, assim, evitar *overfitting*. Dentre as possibilidades, as mais comuns são as penalidades Ridge e Lasso: Ridge reduz muito pesos a números muito pequenos, enquanto Lasso zera muitos pesos, o que pode adicionar estabilidade ao modelo (Hastie et al., 2009).

Outra forma de regularização é o *Dropout*, que restringe a ativação de cada nó com uma probabilidade P. Assim, em cada rodada da propagação do aprendizado, conjuntos diferentes de nós são ativados. Em outras palavras, a rede neural aprende múltiplas representações, que são independentes entre si, mas derivam do mesmo conjunto de dados (Srivastava et al., 2014).

4.5.1 Rede Neural Profunda *feedforward* (DFNN)

Entende-se DL como um subcampo de ML que almeja aprender as representações de alto nível, isto é, a semântica dos dados, a partir de atributos de baixo nível (Glorot and Bengio, 2010).

O desenvolvimento tecnológico trouxe uma quantidade maior de dados, de naturezas distintas (como imagens e sons), mas também muito mais poder computacional, tornando possível paralelizar operações. Isso contribuiu para o desenvolvimento de arquiteturas mais profundas. Assim, DFNN está totalmente apoiada nos conceitos de ANN, entretanto possui um número maior de camadas ocultas. Pode-se dizer que DFNN é uma versão profunda de ANN *feedforward*, mas com melhor estabilidade, habilidade de generalização e escalabilidade (Candel et al., 2017).

Como discutido na seção 4.5, redes neurais com nós de saída linear e uma única camada oculta podem aproximar qualquer função, bastando apenas aumentar o tamanho da camada oculta. Porém, o tamanho dessa camada oculta pode ser impraticável. Além disso, a ANN pode falhar em aprender a função representada e, consequentemente, ter uma pobre generalização. Uma arquitetura mais profunda pode, em muitas circunstâncias, reduzir o

número de unidades necessárias para representar a função desejada, o que pode reduzir os erros de generalização, por ter uma maior capacidade de aprender a função representada. Porém, estas redes profundas também são, geralmente, mais difíceis de otimizar (Goodfellow et al., 2016).

4.5.2 Rede Neural Convolucional (CNN)

Assim como as ANN, CNN são uma tentativa de simular a compreensão humana do mundo. CNNs são baseadas no córtex visual, área responsável pela memória visual. Hubel and Wiesel (1963) notaram que os neurônios estão organizados em uma arquitetura hierárquica, que formam diversos grupos, cada um responsável por responder a diferentes estímulos e que, juntos, são responsáveis pela percepção visual. Baseado nisso, as CNNs são compostas de múltiplos estágios dedicados a extrair atributos, seguidos por camadas ocultas tradicionais e uma camada de classificação final. Cada estágio das CNN é divido em três: o estágio convolucional, o estágio de ativação, e o estágio *pooling*, como mostrado na figura 4.2.

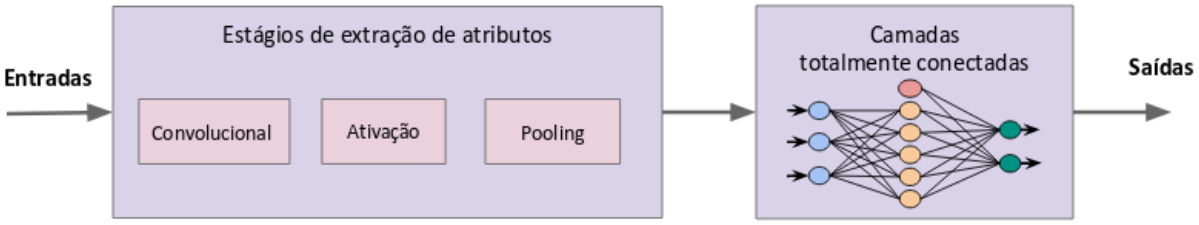


Figura 4.2: Representação de uma CNN

Cada camada desse estágio de extração de atributos é descrito a seguir.

Camada convolucional

A convolução é um tipo especializado de operação linear que, a partir de duas funções, encontra uma terceira, resultado da superposição em função do deslocamento, definida por:

$$conv(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{4.12}$$

Onde as funções x e w estão definidas apenas no tempo t, isto é, são funções discretas. Há também uma versão contínua da convolução, mas fora do escopo deste trabalho.

Em CNN, as funções a convoluir são o vetor multidimensional de dados, geralmente uma imagem I bidimensional, e um filtro K, também chamado de *kernel*, que é adaptado pelo algoritmo de treinamento e geralmente possui o mesmo número de dimensões que a entrada. Assim, a convolução será:

$$conv(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, i-n)K(m,n) \tag{4.13}$$

A imagem 4.3 apresenta um exemplo de convolução em duas dimensões.

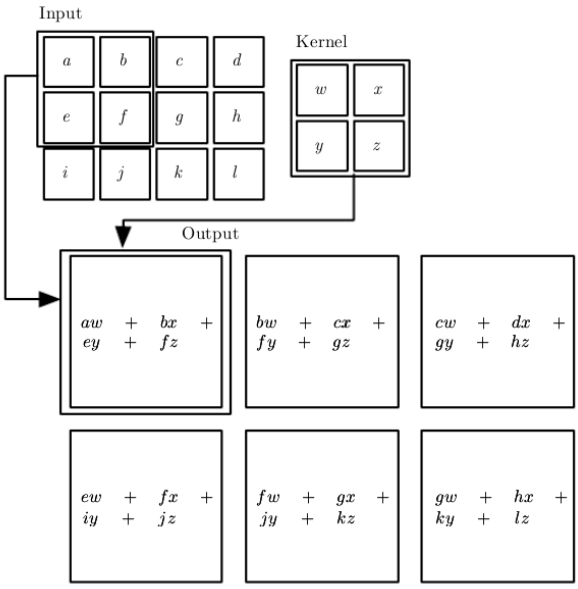


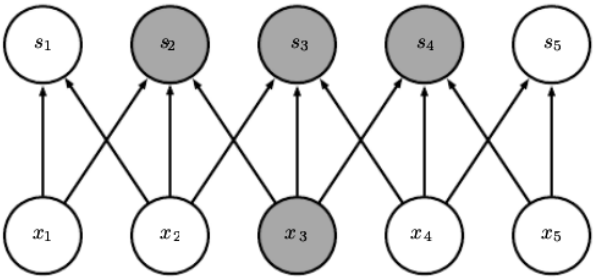
Figura 4.3: Exemplo de convolução discreta 2D (Goodfellow et al., 2016)

Utilizar convolução em ML trouxe três grandes vantagens, que são:

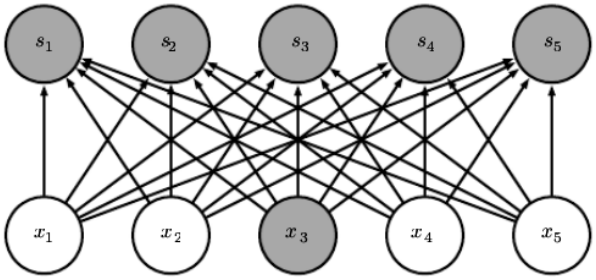
- **Interações esparsas**: como o filtro normalmente é muito menor que o vetor de entrada, há uma melhora na eficiência, visto que se necessita de menos memória de armazenamento e menos operações serão realizadas. Isso pode ser visualizado como se cada nó da camada convolucional se conectasse apenas a uma parte da camada anterior, como visto na figura 4.4, onde a figura 4.4(a) mostra o nó x_3 esparsamente conectado e a figura 4.4(b) mostra o nó x_3 totalmente conectado. Por ser esparso, pode-se pensar que cada pedaço da camada não interagirá com outros – o que não é verdade. Na figura 4.5 vemos que, embora as conexões diretas sejam esparsas, os nós indiretamente estarão quase ou totalmente conectados com os nós das camadas anteriores;

- **Compartilhamento de parâmetros**: refere-se a utilizar o mesmo parâmetro para mais de uma função. Em uma CNN, cada valor do filtro é utilizado em todas as posições do vetor de entrada. Isso significa que em vez de aprender um único valor para cada local, é aprendido um conjunto de valores. Isso não afeta o tempo da propagação do conhecimento, mas reduz o tamanho do armazenamento necessário;

- **Representação equivariante**: a função $f(x)$ é equivariante a função g se $f(g(x)) = g(f(x))$, isto é, se uma entrada muda, a saída muda da mesma forma. Por exemplo,



(a) Esparsamente conectado



(b) Totalmente conectado

Figura 4.4: Exemplo da conectividade esparsa (Goodfellow et al., 2016)

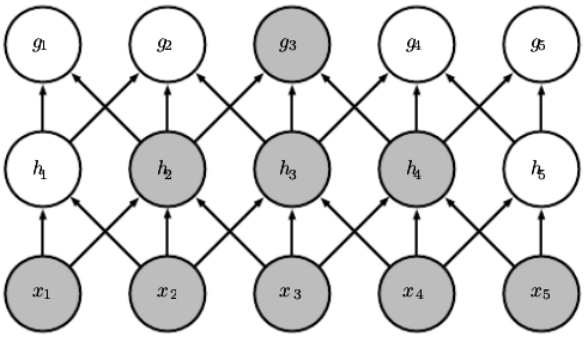


Figura 4.5: Exemplo de propagação através da conectividade esparsa (Goodfellow et al., 2016)

seja I' tal que $I'(x,y) = I(x-1,y)$ (todo pixel de I é mudado 1 unidade para a direita). Aplicar uma transformação em I e depois aplicar convolução é o mesmo que aplicar convolução em I'. Essa propriedade é consequência do compartilhamento de parâmetros (Goodfellow et al., 2016).

Sabe-se que para treinar uma DL necessita-se de uma quantidade massiva de dados para realizar seu treinamento. Porém, graças a representação equivariante, há uma maneira de lidar com isso: criando observações sintéticas a partir das observações do fenômeno real, técnica conhecida como *data augmentation*. Diversos estudos comprovam a eficácia dessa abordagem, como pode ser visto em van Dyk and Meng (2001); Wong et al. (2016).

Para as camadas convolucionais, há três hiperparâmetros que influenciam na sua saída:

- a profundidade da camada, que corresponde ao número de filtros que se deseja ter –

isso influencia diretamente no número de atributos que se deseja extrair;

- o "passo" (*stride*) da convolução, que corresponde a quantos pixels o filtro se desloca ao convoluir, sendo geralmente utilizado 1 ou 2;
- *zero-padding*, que corresponde a preencher as bordas do filtro obtido com zeros e assim controlar o tamanho da saída da convolução. Com esse parâmetro é possível preservar o tanto quanto possível de informação da entrada e assim extrair mais atributos de baixo nível (Karpathy, 2015).

Camada de ativação

Essa camada é composta de uma função de ativação não linear, geralmente um retificador linear – por isso os nós que aplicam essa função são conhecidos por unidade retificada linearmente (*rectified linear unit* - ReLU). A ReLU é definida por:

$$f_{\text{ReLU}}(x) = \max(0, x) \tag{4.14}$$

Apesar de simples, ReLU apresenta várias vantagens, como invariação a escala, eficiência computacional, etc. Essa camada aumenta a não-linearidade da rede, sem alterar os valores saídos do estágio de convolução.

Camada *Pooling*

A camada *pooling*, como o nome sugere, utiliza uma função *pooling*, com o intuito de reduzir o tamanho espacial da representação, e, por consequência, reduzir o número de parâmetros e computação na rede, o que ajuda a controlar o *overfitting*.

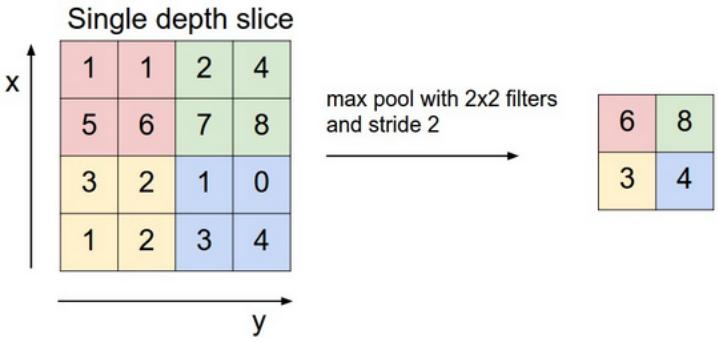


Figura 4.6: Exemplo de *max pooling* (Karpathy, 2015)

A função mais comum é a *max pooling*, que passa o maior valor dos nós conectados à frente. Geralmente, utiliza-se um filtro de tamanho 2×2, com um *stride* de tamanho 2, o que descarta 75 % das ativações de entrada (Karpathy, 2015), como representado na figura 4.6.

Essa operação ajuda a tornar a representação da rede aproximadamente invariante a pequenas transformações. Invariância em translações locais é muito importante, especialmente se é desejado saber se há algum atributo nos dados de entrada, sem se importar com sua localização (Goodfellow et al., 2016).

5

Materiais e Trabalhos Correlatos

5.1 Material

O material utilizado aqui é derivado do trabalho de Pereyra et al. (2014).

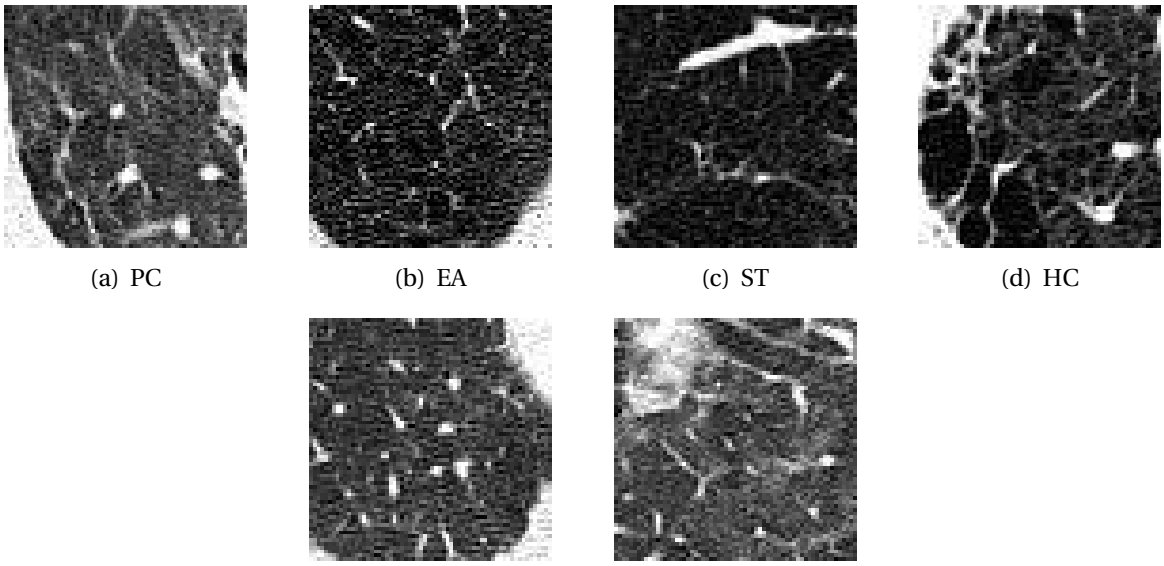
Dentro do Sistema de Informação em Radiologia (*Radiology Information System* - RIS) do Hospital das Clínicas da Faculdade de Medicina de Ribeirão Preto da Universidade de São Paulo (HCFMRP - USP), 247 imagens de TCAR foram recuperadas, pertencentes a 108 diferentes exames realizados no Hospital, com cerca de 35 imagens para cada padrão pulmonar avaliado, que são: pneumonia (*pulmonary consolidation* - PC), áreas enfisematosas (*emphysematous areas* - EA), espessamento de septo (*septal thickening* - ST), favo de mel (*honeycomb* - HC), opacidade em vidro fosco (*ground-glass opacities* - GG) e tecido pulmonar saudável (*normal lung tissues* - NL).

Radiologistas especialistas revisaram os laudos dos exames recuperados para indicar quais cortes e em quais posições dos pulmões se encontravam os exemplos mais típicos dos padrões radiológicos.

Após, esses cortes foram normalizados para 8 bits por pixel. Equalização de Histograma e *centering*, em -400 e -800 unidades Hounsfield (*Hounsfield Units* - HU), foram aplicadas a fim de realçar os parâmetros de atenuação alta e baixa, respectivamente. A segmentação foi feita utilizando morfologia matemática (fechamento), *thresholding* binário (em -800 HU) e seleção de objetos. Todo esse processamento foi feito em Matlab.

Assim, obteve-se um total de 3252 de Regiões de Interesse (*Regions of Interest* - ROIs), com 64×64 pixels cada, agrupadas em: 529 ROIs para HC, 594 para GG, 589 para ST, 450 para PC, 501 para EA, e 589 para NL. A figura 5.1 apresenta exemplos das ROIs para cada padrão radiológico adotado.

Após sua obtenção, as ROIs foram revistas e validadas por radiologistas para garantir que o padrão radiológico estava compatível com a classificação. Porém, infelizmente, as ROIs



(a) PC (b) EA (c) ST (d) HC

(e) NL (f) GG

Figura 5.1: Exemplos das ROIs extraídas

não foram indexadas em relação às imagens originais, impossibilitando a reprodutibilidade dessa base de dados, visto que não se sabe de qual exame estas foram recortadas.

De cada ROI foram extraídas um conjunto de 28 atributos, a saber:

- **Estatísticas de primeira ordem**: média, mediana, desvio padrão, assimetria e curtose;
- **Atributos de textura de Haralick**: energia, momento da diferença, correlação, variância, momento da diferença inversa, soma dos quadrados: média, soma dos quadrados: variância, soma dos quadrados: entropia, entropia, momento da variância, momento da entropia, medida de informação de correlação 1, medida de informação de correlação 2 e máximo coeficiente de correlação (Haralick et al., 1973). Esses atributos foram extraídos para as quatro possíveis vizinhança (0°, 45°, 90° e 135°), usando um pixel como a distância de referência;
- **Medidas de energia de textura de Law**: *Laws' wave measures*, *Laws' wave measures* invariante à rotação, *Laws' ripple measures*, *Laws' ripple measures* invariante à rotação e *Laws level measures* (Laws, 1980). Esses atributos foram extraídos usando as máscaras de convolução apropriadas;
- **Medidas estatísticas da densidade espectral de potência (*power spectral density* - PSD)**: média, desvio padrão e mediana, obtidos da Transformada Discreta de Fourier;
- **Dimensão Fractal (*Fractal Dimension* - FD)**: utilizando o método de Banik et al. (2011).

Detalhes sobre esses atributos, suas obtenções e análise de imagens médicas estão disponíveis no livro de Rangayyan (2004).

5.2 Trabalhos Correlatos

Utilizando esse conjunto de atributos, Pereyra et al. (2014) utilizaram o algoritmo kNN (k = 5) para classificação e obteve até 82.62 % de correta classificação. Seus resultados indicam que os atributos extraídos possuem um bom potencial para a construção de um CAD.

Com esses mesmo dados, Almeida et al. (2015b) analisaram a importância estatística da diferença entre os valores de cada atributo entre suas seis classes através de seus p-valores. Foi mostrado que cerca de 35% dos atributos são úteis para distinguir entre as seis classes com alta significância estatística. Também classificaram os atributos utilizando GMM. Para 78,5% dos atributos, o GMM forneceu, para ao menos uma classe, uma classificação correta mínima de 60%.

Complementar ao trabalho anterior, Almeida et al. (2015a) selecionaram os cinco atributos mais significantes utilizando o algoritmo de Eliminação Recursiva de Atributos (*recursive feature elimination* - RFE) (Liu and Motoda, 2007). Nesses, aplicaram o GMM para obter funções de pertinência Fuzzy (Zadeh, 1965), através de suas possíveis distribuições. Seus resultados obtiveram uma média de 63% de correta classificação.

Neste trabalho, utiliza-se os mesmos algoritmos de classificação, intentando a reprodução dos resultados principais anteriormente obtidos. Além disso, tem-se o interesse de realizar uma comparação de diferentes métodos de ML. Para tal, será utilizado, a depender do algoritmo, tanto o conjunto de atributos, quanto as ROIs adquiridas.

6

Resultados e Discussões

Utilizando os dados provindos de Pereyra et al. (2014), realizou-se a aplicação das técnicas de classificação descritas no capítulo 4 deste trabalho. Todos os algoritmos, com exceção da CNN, utilizaram o conjunto de dados contendo os 28 atributos extraídos das imagens. CNN utilizou as imagens diretamente.

Todo o processo computacional, exceto CNN, foi feito na linguagem R (versão 3.2.2) (R Core Team, 2015), devido a sua confiabilidade e precisão ao lidar com funções estatísticas (Almiron et al., 2009). Mais especificamente, utilizou-se os seguintes pacotes para os respectivos algoritmos:

- PCA: `stats` (e sua função `prcomp`)
- SSF, SSB, SSFB: `stats` (e sua função `step`)
- LDA: `MASS` (e sua função `lda`)
- GMM: `mclust` (e sua função `MclustDA`)
- SVM: `e1071` (e sua função `svm`)
- KNN: `class` (e sua função `knn`)
- ANN: `nnet` (e sua função `nnet`)
- DFNN: `h2o` (e sua função `h2o.deeplearning`)

Para CNN utilizou-se a linguagem Python (Rossum, 1995), que, apesar de ser de uso geral, possui diversas bibliotecas de ML, além de ser bastante amigável. Utilizou-se a biblioteca Keras (Chollet et al., 2015), que funciona como uma interface para a biblioteca Tensorflow (Abadi et al., 2015), as quais são as ferramentas mais modernas para DL até o momento.

	Desvio Padrão	Proporção de Variância	Proporção Cumulativa
PC01	3.3624	0.4038	0.4038
PC02	2.6526	0.2513	0.6551
PC03	1.9894	0.1413	0.7964
PC04	1.52620	0.08319	0.87961
PC05	1.00034	0.03574	0.91535
PC06	0.76172	0.02072	0.93607
PC07	0.71461	0.01824	0.95431
PC08	0.51650	0.00953	0.96384
PC09	0.44830	0.00718	0.97101
PC10	0.4232	0.0064	0.9774
PC11	0.40827	0.00595	0.98336
PC12	0.34610	0.00428	0.98764
PC13	0.30433	0.00331	0.99095
PC14	0.24969	0.00223	0.99318
PC15	0.22595	0.00182	0.99500
PC16	0.18113	0.00117	0.99617
PC17	0.1586	0.0009	0.9971
PC18	0.14108	0.00071	0.99778
PC19	0.12273	0.00054	0.99832
PC20	0.11506	0.00047	0.99879
PC21	0.10195	0.00037	0.99916
PC22	0.08887	0.00028	0.99944
PC23	0.08317	0.00025	0.99969
PC24	0.06644	0.00016	0.99985
PC25	0.05915	0.00012	0.99997
PC26	0.02603	0.00002	1.00000
PC27	0.008391	0.000000	1.00000
PC28	3.371e-15	0.000e+00	1.000e+00

Tabela 6.1: Resultado do PCA

Todos os algoritmos de classificação utilizados aqui foram validados através do método validação cruzada (*cross-validation*), com 10-*folds*. Para toda computação em R, o *seed* foi 321 e em Python, utilizou-se 7.

Para gerar os resultados aqui obtidos, utilizou-se um computador com um processador i5, 8 GB de memória RAM, sem GPU.

Os resultados aqui apresentados estão divididos em duas partes: pré-processamento e classificação.

6.1 Pré-processamento

Ao utilizar PCA, obtemos os dados exibidos na tabela 6.1, que são o desvio padrão, a proporção de variância de cada componente individualmente e a proporção cumulativa da variância do conjunto de dados. Notamos que as 13 primeiras componentes representam 99.09 % da variância dos dados. Individualmente, vemos que a partir da componente 8 é expressado menos de 0.01 % de variância em cada componente e a partir da componente 17, a expressão é de menos de 0.001 %. Em outras palavras, a variância presente a partir da componente 8 é quase insignificante.

Na figura 6.1 temos a proporção de variância para PCA e LDA, respectivamente. Apenas 15 componentes principais foram plotadas por uma questão de clareza, o que não impacta na comparação, devido a falta de significância das componentes seguintes.

Nota-se que LDA precisa de apenas 5 discriminantes lineares para explicar os dados. Tanto em PCA quanto em LDA, a primeira dimensão escolhida representa cerca de 40 % da variância total dos dados.

A diminuição obtida pelos métodos PCA e LDA sugerem a presença de atributos insignificantes no conjunto de dados. No entanto, como estas técnicas transformam os dados originais em um novo espaço, ainda seria necessário extrair todos os atributos das imagens.

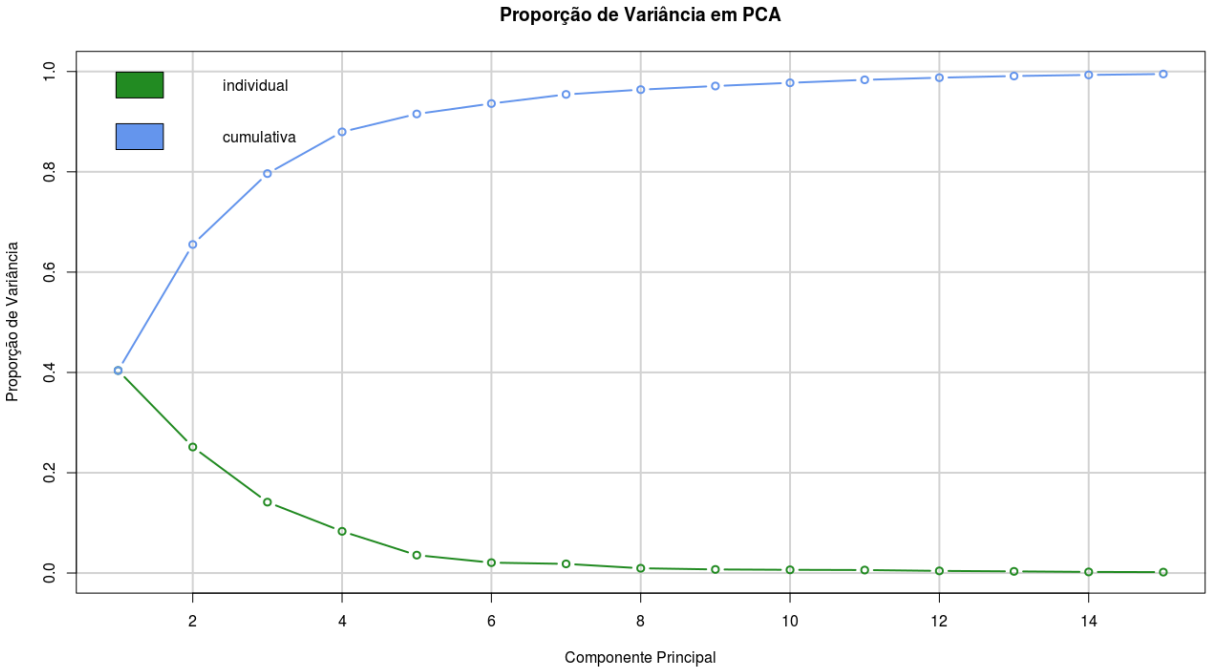
Com as seleções *stepwise*, ao utilizar SSF, conseguimos um subconjunto com 11 variáveis. Para a seleção SSB, conseguiu-se um subconjunto com 18 variáveis. E para a seleção SSFB, um subconjunto com 12 variáveis. Podemos notar que a seleção SSF nos fornece o menor subconjunto dentre as três.

A tabela 6.2 mostra quais atributos foram escolhidos utilizando cada seleção *stepwise*. O símbolo **X** representa que o atributo se encontra no subconjunto. Nota-se que alguns atributos, como média e desvio padrão da PSD não foram escolhidos por nenhuma técnica, enquanto outros foram escolhidos por todas as técnicas. Isso pode ser uma indicação de significância dos atributos, isto é, se tais atributos são irredundantes ou irrelevantes.

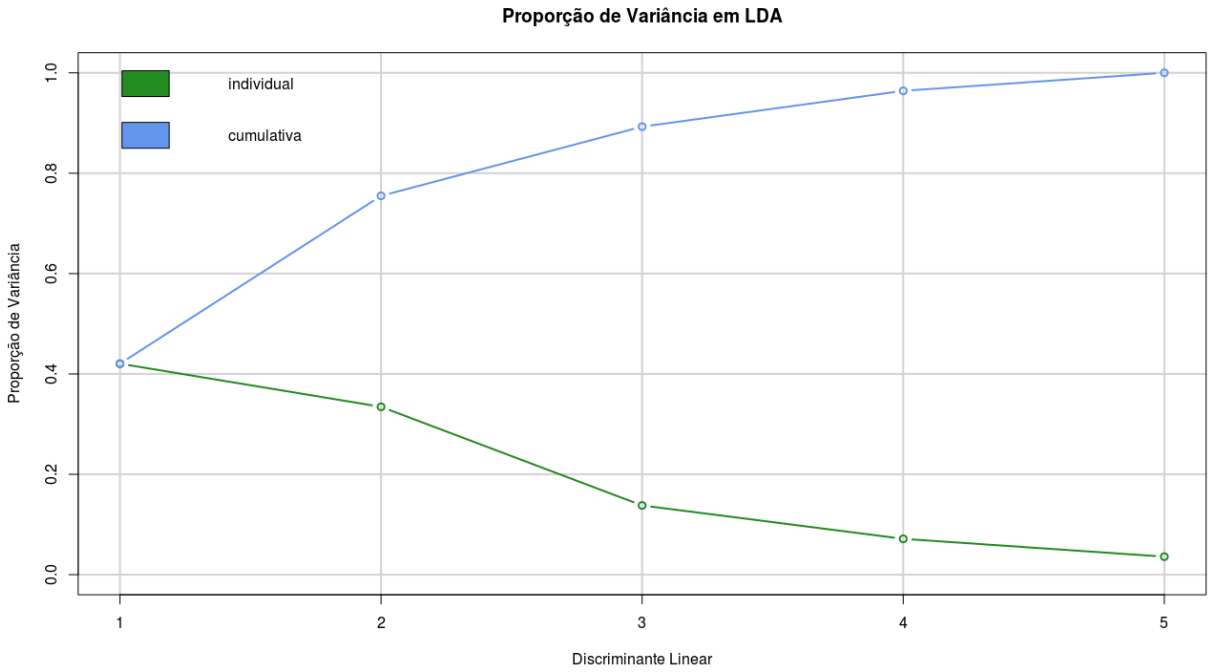
A tabela 6.3 apresenta o número de dimensões original e o quantas dimensões restaram com a utilização de cada técnica. Vemos que LDA apresenta a melhor redução de dimensionalidade (de 28 atributos para 5), seguida por SSF, SSFB, PCA e SSB.

6.2 Classificação

O conjunto de dados de atributos original e os subconjuntos obtidos com as técnicas de redução de dimensionalidade de seleção de atributos são utilizados como entrada dos algoritmos de classificação, com exceção da CNN, que utiliza as próprias imagens. Assim, podemos comparar o quão boa é a redução de dimensões, ou se esta é mesmo necessária no caso aqui estudado. É importante saber que cada algoritmo, devido as suas características intrínsecas, se comporta de maneira diferente. Dessa forma, cada classificador pode se comportar de



(a) PCA



(b) LDA

Figura 6.1: Proporção de variância de PCA e LDA

Atributos	SSF	SSB	SSFB
Média	—	—	—
Mediana	X	X	X
Desvio padrão	—	X	—
Assimetria	—	X	—
Curtose	—	X	—
Média de PSD	X	X	X
Mediana de PSD	X	X	X
Desvio padrão de PSD	—	—	—
Dimensão fractal	X	X	X
Laws' wave measures	—	X	X
***Laws' wave measures* invariante à rotação**	—	X	—
Laws' ripple measures	—	—	—
***Laws' ripple measures* invariante à rotação**	—	—	—
Laws level measures	X	—	X
Energia de Haralick	—	X	—
Momento da diferença de Haralick	—	X	—
Correlação de Haralick	X	X	X
Variância de Haralick	—	X	X
Momento da diferença inversa de Haralick	X	X	X
Soma dos quadrados: média de Haralick	—	—	—
Soma dos quadrados: variância de Haralick	—	—	—
Soma dos quadrados: entropia de Haralick	X	X	X
Entropia de Haralick	—	X	—
Momento da variância de Haralick	X	—	X
Momento da entropia de Haralick	X	X	X
Medida de informação de correlação 1 de Haralick	—	—	—
Medida de informação de correlação 2 de Haralick	—	—	—
Máximo coeficiente de correlação de Haralick	X	X	—

Tabela 6.2: Comparação entre os subconjuntos obtidos em cada seleção stepwise

	Original	PCA	SSF	SSB	SSFB	LDA
Dimensões	28	13	11	18	12	**5**

Tabela 6.3: Comparação das dimensões resultantes no pré-processamento

uma maneira diferente com a mesma entrada. Não existe uma solução universal para todos os problemas, sendo necessário testar vários métodos para se chegar ao resultado desejado.

Se os algoritmos apresentam um comportamento não determinístico, então seus parâmetros também não possuem uma solução universal. Assim, é necessário testar o algoritmo com diferentes valores em seus parâmetros para encontrar a melhor solução. Note que essa solução não é necessariamente a solução ótima. Todos os valores para os parâmetros aqui obtidos foram conseguidos de forma empírica, sem um número fixo de testes.

Para o kNN, utilizamos a mesma configuração que Pereyra et al. (2014), $k = 5$. Para GMM, que foi anteriormente utilizado em Almeida et al. (2015b,a) em sua forma não-supervisionada, preferimos utilizar a versão supervisionada, por ter apresentado melhores resultados.

Para as outras técnicas de classificação, a configuração que apresentou o melhor resultado foi a seguinte:

- SVM: um *kernel* de polinômio não-homogêneo $(\gamma u'v + b)^d$, com $d = 2$, $\gamma = 0,5$, $b = 3$ e a penalidade de violação $C = 1$;
- ANN: função de ativação *softmax*, função de treinamento BFGS quasi-Newton, uma taxa de decaimento de 10^{-5} e uma camada oculta com 8 nós para o conjunto de dados original, PCA e SSB; 11 para LDA; 15 para SSF e 9 para SSFB;
- DFNN: 300 épocas, penalidades Ridge e Lasso com valores de limitação 10^{-5}, três camadas ocultas com 230 nós cada, função de treinamento ADADELTA (Zeiler, 2012), com taxa de aprendizado 1 e hiperparâmetros $\rho = 0.99$ e $\epsilon = 10^{-8}$ e função de ativação tangente hiperbólica;
- CNN: a figura 6.3 mostra a arquitetura aqui utilizada. Cada camada convolucional tem seu filtro do tamanho 3×3, $stride = 1$ e zero-padding; as camadas de *maxPooling* possuem um filtro do tamanho 2×2 e $stride = 2$; a função de ativação para a saída é uma *softmax*; e a função de treinamento é a ADADELTA, com taxa de aprendizado 1 e hiperparâmetros $\rho = 0.99$ e $\epsilon = 10^{-8}$; A camada totalmente conectada utiliza *Dropout* com $P = 0,3$.

Para CNN utilizou-se *data augmentation*, pois, com apenas 3252 observações, é possível que haja uma memorização das entradas, o que deseja ser evitado. Assim, cria-se mais observações sintéticas, através de transformações nas imagens originais. Neste trabalho foi utilizado *shear range* (uma aplicação aleatória das transformações shearing (Tan et al., 2005)), zoom aleatório e giros horizontais. Alguns exemplos dessas observações sintéticas obtidas pode ser visto na figura 6.2.

Para avaliação de desempenho desses classificadores, como nossos dados são praticamente uniformemente distribuído, os valores de medidas macro e micro-averaged são

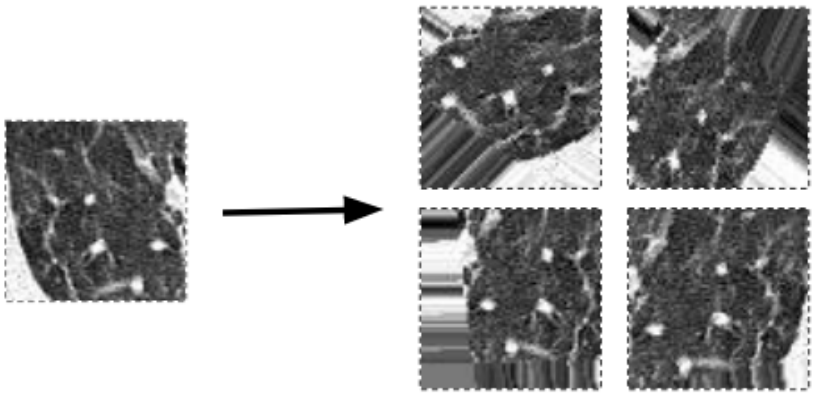


Figura 6.2: Exemplos de observações obtidas com *data augmentation*

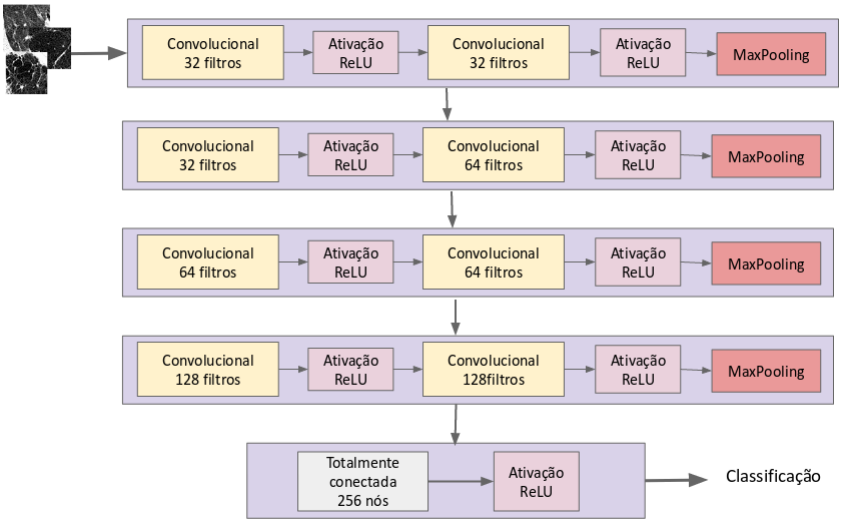


Figura 6.3: Arquitetura da CNN utilizada

Classificador	Subconjunto	Tempo (s)	CI (s)
kNN	LDA	0.009	0.0003
ANN	SSB	8.83	0.0003
SVM	SSB	0.59	0.0088
GMM	SSF	1.42	0.0803
DFNN	Original	1375.45	0.0943
CNN	Imagens	46048.60	2224.67

Tabela 6.4: Tempo para treinamento

Classificador	Subconjunto	Macro Especificidade	Macro Sensibilidade	Macro F1	Acurácia
ANN	SSB	96.86 %	84.48 %	84.49 %	84.40 %
GMM	SSF	96.87 %	84.61 %	84.66 %	84.46 %
kNN	LDA	96.74 %	83.87 %	83.89 %	83.82 %
SVM	SSB	97.31 %	86.73 %	86.76 %	86.64 %
DFNN	Original	99.92 %	99.62 %	99.60 %	99.60 %
CNN	Imagens	99.59 %	99.40 %	99.18 %	99.24 %

Tabela 6.5: Melhores classificações obtidas

muito semelhantes. Dessa forma, decidimos mostrar apenas as medidas macro-averaged, a fim de evitar informação redundante. Os resultados obtidos podem ser visto na tabela 6.5.

SVM apresenta o terceiro melhor resultado entre os métodos avaliados, com uma acurácia de 86.64 %. Porém, esses resultados são superados tanto pela CNN, com 99.24 % de acurácia, quanto pela DFNN, com 99.60 %. Note que em todas as métricas utilizadas, DFNN e CNN apresentam melhores resultados, com DFNN atingindo taxas sensivelmente mais altas que CNN.

É válido ressaltar que os resultados apresentados para kNN e GMM superam aqueles apresentados anteriormente por Pereyra et al. (2014) (kNN com 82.62 % de acurácia) e Almeida et al. (2015b) (GMM com 63 % de acurácia).

O tempo utilizado pelos classificadores apresentados para treino pode ser visto na tabela 6.4. Enquanto kNN, ANN, SVM e GMM levam menos de 10 segundos para treinar com os dados, DFNN leva cerca de três horas. O mais devagar, porém, é CNN, com cerca de 12 horas, 3 vezes mais que DFNN. Isso pode estar relacionado ao tempo de extração de atributos, uma vez que CNN recebe diretamente as imagens como entrada. O tempo para predição é desprezível quando comparado ao de treino e muito semelhante para todas as técnicas.

7

Considerações Finais

Este trabalho avaliou técnicas de classificação juntamente com técnicas de pré-processamento, aplicadas a seis padrões pulmonares relativos a DPDs.

Os dados aqui utilizado derivam do trabalho de Pereyra et al. (2014), sendo estes os responsáveis pela extração das ROIs e dos atributos. Infelizmente não é possível a reprodução dessas bases de dados pela falta de indexação das imagens originais.

Há muitas técnicas de pré-processamento, como agregação, limpeza, entre outras. Aqui apenas reduzimos a dimensionalidade, a fim de evitar uma possível Maldição da Dimensionalidade. Utilizamos LDA, PCA, SSF, SSB e SSFB. Note que há várias outras técnicas possíveis para essa redução. Com LDA obtivemos uma redução de 28 para 5 dimensões. Porém LDA transforma os dados para um novo espaço. Assim, seria necessário extrair das imagens todas as dimensões originais da mesma forma. Foge do escopo deste trabalho avaliar se tal transformação vale a pena. O segundo melhor resultado é obtido por SSF, de 28 dimensões para 11. Utilizando essa técnica os dados não são transformados. Porém, não há garantias de que esse seja o subconjunto ótimo.

Nos três subconjuntos obtidos através das técnicas *stepwise*, há alguns atributos que são adicionados aos três e há alguns que não são adicionados a nenhum subconjunto. Isso pode ser uma indicação de significância desses atributos – os descartados podem ser redundantes ou irrelevantes. Para se ter certeza seria necessário uma análise mais profunda desses atributos, o que não está dentro do escopo deste trabalho. Trabalhos futuros envolvem o uso de outras técnicas para seleção de atributos.

Para classificação, algumas outras técnicas foram utilizadas, como Regressão Logística, porém não apresentaram resultados satisfatórios e, por isso, foram excluídas deste trabalho. CNN e DFNN apresentam resultados semelhantes, acima de 99 % de acurácia, mais de 10 % acima das outras técnicas utilizadas, que são kNN, SVM, GMM e ANN. Porém o tempo para o treinamento é muito diferente: kNN, SVM, GMM e ANN levam menos de 10 segundos, enquanto CNN e DFNN levam horas, com CNN precisando cerca de três vezes mais tempo

do que a DFNN. Entretanto, CNN traz a vantagem de receber diretamente as imagens como entrada, enquanto para as outras técnicas se necessita que os atributos sejam extraídos anteriormente.

Como a base de dados não é grande, contendo apenas 3252 observações, é possível que DFNN esteja memorizando essas observações. Com o que foi feito até o momento, não é possível saber se isso é uma realidade. Trabalhos futuros pretendem investigar essa possibilidade. Com CNN há mais confiança no resultado devido a criação das observações sintéticas através de *data augmentation*.

Os resultados obtidos até o momento podem contribuir para a construção de um CAD para essas DPDs estudadas.

Ao longo deste trabalho gerou-se as seguintes publicações:

- SILVA, I. C. P.; RAMOS, H. S. ; ALMEIDA, E. S. . Classificação de Doenças Intersticiais Pulmonares Difusas através de Tomografia Computadorizada de Alta-Resolução. In: CSBC 2016 - Workshop de Informática Médica (WIM), 2016. Anais do CSBC 2016, 2016.

- SILVA, I. C. P.; RAMOS, H. S. ; ALMEIDA, E. S. . Classificação de Imagens Pulmonares por Tomografias Computadorizadas de Alta-Resolução. In: XVI Escola Regional de Computação Bahia Alagoas Sergipe (ERBASE), 2016, Maceió. Anais do Workshop de Trabalhos de Iniciação Científica e Graduação (WTICG), 2016. v. 1. p. 181-190.

E, no momento, há uma outra publicação aceita no 22º Congresso Ibero-americano em Reconhecimento de Padrões (*22nd Iberoamerican Congress On Pattern Recognition* - CIARP) 2017.

Por se tratar de um Trabalho de Conclusão de Curso, é importante ressaltar o aprendizado obtido, através do estudo dos assuntos expostos e suas aplicações, ajudando assim na criação de uma boa base na área de Inteligência Artificial.

Referências bibliográficas

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

P. Adriaans and D. Zantinge. *Data Mining*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

E. Almeida, R. Rangayyan, and P. Azevedo-Marques. Fuzzy membership functions for analysis of high-resolution CT images of diffuse pulmonary diseases. In *Engineering in Medicine and Biology Society EMBC, 37th Annual International Conference of the IEEE*, pages 719–722, 2015a.

E. Almeida, R. Rangayyan, and P. Azevedo-Marques. Gaussian mixture modeling for statistical analysis of features of high-resolution CT images of diffuse pulmonary diseases. In *Medical Measurements and Applications (MeMeA), 2015 IEEE International Symposium on*, pages 1–5, 2015b.

M. Almiron, E. Almeida, and M. Miranda. The reliability of statistical functions in four software packages freely used in numerical computation. In *Brazilian Journal of Probability and Statistics*, pages 107–119, 2009.

S. Angel, J. Parent, D. Civco, A. Blei, and D. Potere. The dimensions of global urban expansion: Estimates and projections for all countries, 2000–2050. *Progress in Planning*, 75(2):53 – 107, 2011.

S. Banik, R. M. Rangayyan, and J. Desautels. Detection of architectural distortion in prior mammograms. *IEEE Transactions on Medical Imaging*, 30(2):279–294, Feb 2011.

I. Bankman. *Handbook of Medical Imaging: Processing and Analysis Management*. Biomedical Engineering. Elsevier Science, 2000.

S. Basu, J. Chakraborty, A. Bag, and M. Aftabuddin. A review on emotion recognition using speech. In *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 109–114, March 2017.

S. Brindha, K. Prabha, and S. Sukumaran. A survey on classification techniques for text mining. In *2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 01, pages 1–5, Jan 2016.

Z. Cai, D. Xu, Q. Zhang, J. Zhang, S. Ngai, and J. Shao. Classification of lung cancer using ensemble-based feature selection and machine learning methods. *Molecular BioSystems*, 11:791–800, 2015.

A. Candel, V. Parmar, E. LeDell, and A. Arora. *Deep Learning with H2O*. H2O.ai, Inc., 5th edition, 2017.

F. Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

B. Elicker, C. Pereira, R. Webb, and K. Leslie. Padrões tomográficos das doenças intersticiais pulmonares difusas com correlação clínica e patológica. In *Jornal Brasileiro de Pneumologia, Volume 34*, number 9, pages 715–744, Sep 2008.

C. Fraley and A. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97:611–631, 2000.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS'10), Proceedings of the*. Society for Artificial Intelligence and Statistics, 2010.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

R. Haralick, K. Shanmugam, and I. Dinstein. Textural features for image classification. *Systems, Man, and Cybernetics, IEEE Transactions on*, (6):610–621, 1973.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2003.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR), The IEEE Conference on*, pages 770–778, 2016.

D. Hubel and T. Wiesel. Shape and arrangement of columns in cat's striate cortex. *The Journal of physiology*, 165(3):559–568, 1963.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer, 2013.

Andrej Karpathy. Stanford University CS231n: Convolutional Neural Networks for Visual Recognition. 2015.

K. Laws. Rapid texture identification. In *Image Processing for Missile Guidance, Proc. SPIE Conf.*, volume 238, pages 376–380, 1980.

G. Litjens, T. Kooi, B. Bejnordi, A. Setio, F. Ciompi, M. Ghafoorian, J Laak, B. Ginneken, and C. Sánchez. A survey on deep learning in medical image analysis. *CoRR*, abs/1702.05747, 2017.

Huan Liu and Hiroshi Motoda. *Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, 2007.

L. Pereyra, R. Rangayyan, M. Ponciano-Silva, and P. Azevedo-Marques. Fractal analysis for computer-aided diagnosis of diffuse pulmonary diseases in HRCT images. In *Medical Measurements and Applications (MeMeA), 2014 IEEE International Symposium on*, pages 1–6, 2014.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL https://www.R-project.org.

R.M. Rangayyan. *Biomedical Image Analysis*. CRC Press, 2004.

M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio. A network of deep neural networks for distant speech recognition. *CoRR*, abs/1703.08002, 2017.

B. Ripley and N. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA, 1st edition, 1995.

G. Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

A. Rubin, A. Santana, A. Costa, B. Baldi, C. Pereira, et al. Diretrizes de doenças pulmonares intersticiais da Sociedade Brasileira de Pneumologia e Tisiologia. *The Brazilian Journal of Pulmonology*, 38(suppl. 2):S1–S133, 2012.

C Salvatore, A Cerasa, I Castiglioni, F Gallivanone, A Augimeri, M Lopez, G Arabia, M Morelli, MC Gilardi, and A Quattrone. Machine learning on brain MRI data for differential diagnosis of Parkinson's disease and Progressive Supranuclear Palsy. *Neuroscience Methods, Journal of*, 222:230–237, 2014.

M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manage.*, 45(4):427–437, 2009.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, January 2014. ISSN 1532-4435.

P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

Department of Economic United Nations and Population Division Social Affairs. *World Population Prospects: The 2017 Revision, Key Findings and Advance Tables*. Working Paper No. ESA/P/WP/248, 2017.

D. van Dyk and X. Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

S. Wong, A. Gatt, V. Stamatescu, and M. McDonnell. Understanding data augmentation for classification: when to warp? *CoRR*, abs/1609.08764, 2016.

L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

Mohammed J. Zaki and Jr. Wagner Meira. *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press, May 2014.

M. Zeiler. ADADELTA: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

B. Zheng, S. Yoon, and S. S. Lam. Breast cancer diagnosis based on feature extraction using a hybrid of k-means and support vector machine algorithms. *Expert Syst. Appl.*, 41(4): 1476–1482, 2014.