Seleção de Instâncias em Espaço Métrico de *Word Embedding*

Eliton Luiz Scardin Perin

SERVIÇO DE PÓS-GRADUAÇÃO DA FACOM-UFMS

Data de Depósito:

Assinatura:_____

Eliton Luiz Scardin Perin

Orientador: *Prof. Dr. Edson Takashi Matsubara*
Coorientador: *Prof. Dr. Eraldo Luís Rezende Fernandes*

UFMS - Campo Grande
março/2018

Aos meus pais,
Erondina e Dilceu,

Aos meus irmãos,
Alan e Diesey,

À namorada,
Luiza,

Aos orientadores
Edson Takashi Matsubara e
Eraldo Luís Rezende Fernandes

Agradecimentos

Agradeço à Deus pelas oportunidades que me fizeram chegar até aqui e poder fazer parte desse trabalho. Agradeço a minha família à contribuição que deram na minha vida, principalmente minha mãe em sabedoria, e também pela compreensão e esforço em me auxiliar e manter estudando. Agradeço a minha namorada por me dar forças e tranquilizar nos momentos complicados e compreender minha ausência. Agradeço ao meu tio Azyr e o primo Alexandre pela hospitalidade, nos dias que precisei de ajuda. Agradeço muito aos meus orientadores pela paciência e dedicação, os professores doutores Edson Takashi Matsubara e Eraldo Luís Rezende Fernandes. Agradeço também aos professores doutores Solange de Oliveira Rezende, Ricardo Marcondes Marcacine e Bruno Magalhães Nogueira por participar da banca de defesa.

Agradeço a todos do Laboratório de Inteligência Artificial (LIA) da UFMS que quase todos os dias acompanhava o meu trabalho e auxiliava sempre que precisava seja em alguma dúvida ou palavra de apoio. Ao pessoal do PET (Programa de Educação Tutorial) da UFMS que também contribuiu para o desenvolvimento deste trabalho, em especial ao Yan Uehara.

Agradeço também a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), pelo apoio financeiro durante o projeto de mestrado.

Agradeço a todos que de alguma forma direta ou indireta contribuíram para meu trabalho.

Resumo

Classificação de textos é uma tarefa difícil para ser processada automaticamente por uma computador. No entanto, com o avanço em pesquisas com redes neurais artificiais e Processamento de Linguagem Natural tem apresentado resultados surpreendentes nesta tarefa e outras da mesma área.

Dados textuais possuem características de dados não estruturados. Estes tipos de dados necessitam de representações computacionais. As redes neurais artificias possibilitaram o reconhecimento de padrões e extração de características de um grande quantidade de dados.

Uma extração de característica impressionante adquirida com uso de redes neurais são os vetores de representação de palavra, também conhecidos por *word embeddings*. É possível realizar operações matemáticas com os vetores aprendidos por essa rede neural, disponibilizada pela ferramenta do WORD2VEC.

Outro algoritmo que possui bons resultados para classificação de dados textuais é o k-*Nearest Neighbors* (KNN). O algoritmo do KNN é dependente de uma métrica para comparar os dados para classificação. A métrica ainda pode ser desenvolvida para melhorar a representação dos dados textuais, usando algoritmos de aprendizado de métrica de distância.

Juntamente com os *word embeddings*, o KNN e aprendizado de métrica de distância apresentou resultados do estado da arte para a tarefa de classificação de textos.

Algumas desvantagens do método do KNN é que o processamento de dados cresce com a muito com quantidade de dados que lhe é fornecido. Técnicas como Seleção de Instâncias são úteis nesta tarefa como etapa de pré-processamento, eliminando dados que sejam supérfluos.

Este trabalho tem o propósito analisar os ganhos obtidos ao utilizar as combinações de algoritmos de aprendizado de métrica de distância, representação de dados textuais e algoritmos de Seleção de Instâncias.

Os resultados obtidos por esta pesquisa demonstram que o uso de Seleção de Instâncias utilizando os *word embeddings* são eficientes e aumentam a acurácia do classificador KNN em média 11% comparados sem o uso dos *word embeddings* em três *datasets*. Outro ganho é na quantidade de exemplos necessários para realizar o treino do modelo, diminuindo 30% das instâncias para método de Seleção de Instâncias CNN.

Abstract

Classification of texts is a difficult task to be processed automatically by computer. However, with the advancement in research with artificial neural networks and Natural Language Processing has presented surprising results in this task and others in the same area.

Textual data has unstructured data characteristics. These types of data require computational representations. Artificial neural networks enabled pattern recognition and feature extraction of a large amount of data.

An impressive feature extraction acquired with the use of neural networks are the word representation vectors, also known as *word embedding*. It is possible to perform mathematical operations with the vectors learned by this neural network, made available by the WORD2VEC tool.

Another algorithm that has good results for textual data classification is k-*Nearest Neighbors* (KNN). The KNN algorithm is dependent on a metric to compare the data for classification. The metric can still be developed to improve the representation of textual data, using distance metric learning algorithms.

Along with the *word embeddings*, the KNN and distance metric learning presented state-of-the-art results for the task of classification of texts.

Some disadvantages of the KNN method is that the data processing grows with much data quantity provided to it. Techniques such as Instances Selection are useful in this task as a preprocessing step, eliminating data that are superfluous.

This work has the purpose of analyzing the gains obtained by using the combinations of distance metric learning algorithms, textual data representation and Instances Selection algorithms.

The results obtained by this research demonstrate that the use of Instances Selection using the *word embeddings* are efficient and increase the accuracy of the classifier KNN on average 11 % compared without the use of *word embeddings* in three *datasets*. Another gain is the number of examples needed to perform the model training, decreasing 30 % of instances to Instances Selection CNN method.

Sumário

Sumário xiv
Lista de Figuras xvi
Lista de Tabelas xvii
Lista de Abreviaturas xix
Lista de Notações xxiii
Lista de Algoritmos xxv

1 Introdução **1**
1.1 Histórico 1
1.2 Objetivo 3
1.3 Proposta 3

2 Fundamentos e Conceitos **5**
2.1 Processamento de Linguagem Natural 5
2.2 Aprendizado de Máquina 7
2.2.1 k-*Nearest Neighbors* 7
2.2.2 Definições de Aprendizado de Métrica 8
2.2.3 Aprendizado de Métrica de Distância 9
2.2.4 Redes Neurais 11
2.3 Transformação e Representação de Dados 26
2.3.1 *Bag-of-Words* 26
2.3.2 Word2vec 28
2.4 Métricas para Avaliação dos Classificadores 31
2.4.1 Precision 31
2.4.2 Accuracy 32
2.4.3 Recall 32
2.4.4 F1-Score 32

3 Seleção de Instâncias **35**
3.1 *Condensed Nearest Neighbor* 36
3.2 *Edited Nearest Neighbor* 37
3.3 *Random Mutation Hill Climbing* 37
3.4 *Silhouette* 39

4 Algoritmos de Otimização com Aprendizado de Métrica **43**
4.1 *Earth Mover's Distance* 43
4.2 *Word Mover's Distance* 45
4.3 *Supervised Word Mover's Distance* 49

5 Avaliação Experimental **55**
5.1 Critérios de Avaliação 56
5.2 Gráficos de Avaliação Multi-critério 57
5.3 Descrição dos *Datasets* Textuais Utilizados 58
5.4 Resultados 59
5.4.1 *Word Embedding* Melhora Métodos de Seleção de Instâncias? 60
5.4.2 Seleção de Instâncias Somente com *Word Embeddings* vs com Aprendizado de Métrica e *Word Embeddings* 66
5.4.3 Desempenho em *Accuracy* com Seleção de Instâncias . . . 70

6 Conclusões **75**
6.1 Contribuições 76
6.2 Trabalhos Futuros 76

Referências **83**

Lista de Figuras

1.1 Exemplo do funcionamento do WMD. Figura original do artigo (Kusner et al., 2015) . . . 2
1.2 Transformação do espaço métrico utilizando S-WMD com datasets benchmark de textos. Figura original do artigo (Huang et al., 2016) . . . 4

2.1 Etapas de treino e teste para um kNN sem Aprendizado de Métrica usando a distância Euclideana. . . . 8
2.2 Representação simplificada de um neurônio extraído de da Silva et al. (2010). . . . 11
2.3 Modelo matemático de um neurônio. . . . 12
2.4 Função logística para $w = 1$. . . . 16
2.5 Representação de uma rede neural. . . . 21
2.6 Representação de uma Rede Neural Artificial PMC. . . . 23
2.7 Diagrama das topologias de redes neurais. . . . 29
2.8 Relação entre as representações vetoriais com *word embedding* produzidos com Word2Vec (Mikolov et al., 2013d). . . . 30

3.1 Representação gráfica do processo de Seleção de Instâncias. . . . 35
3.2 Representação gráfica de 3 agrupamentos para exemplificar o valor de *Silhouette* . . . 40
3.3 Representação gráfica do processo de construção da distância usando a *Silhouette* para seleção de instância do Silhouette Selection. . . . 41

4.1 Exemplo de grafo bipartido com três fornecedores e dois consumidores. Extraída de Rubner et al. (1998). . . . 44
4.2 Caracterização da distância WMD entre os documentos $doc^{(1)}$ e $doc^{(2)}$. . . . 46
4.3 Representação do espaço de métrica WCD e suas distâncias entre a média dos seus vetores de palavras. . . . 48
4.4 Transformações sobre os dados para produção das distância do método S-WMD. . . . 49
4.5 Transformação do WCD em S-WCD pela matriz A. . . . 52
4.6 Representação gráfica do processo de construção da distância S-WMD. . . . 52

5.1 Configuração de treino e teste para a representação BOW sobre o kNN usando distâncias Euclideanas. . . . 56

5.2 Configuração de treino e teste para a representação WCD sobre o kNN usando distâncias Euclideanas. 56
5.3 Configuração de treino e teste para a distância S-WMD sobre o kNN usando distâncias Euclideanas. 57
5.4 Configuração de treino e teste para a representação S-WCD sobre o kNN usando distâncias Euclidianas. 57
5.5 Exemplo de gráfico de avaliação multi-critério utilizado neste trabalho. Eixo y, quanto maior melhor; eixo y, quanto menor melhor 58
5.6 Desempenho vs número médio de exemplos selecionados no *dataset* Reuters. 65
5.7 Desempenho vs número médio de exemplos selecionados no *dataset* BBC Sport. 66
5.8 Desempenho vs número médio de exemplos selecionados no *dataset* Twitter. 66
5.9 Desempenho vs tempo de seleção mais tempo de teste no *dataset* Reuters. 67
5.10 Desempenho vs tempo de seleção mais tempo de teste no *dataset* BBC Sport. 67
5.11 Desempenho vs tempo de seleção mais tempo de teste no *dataset* Twitter. 68
5.12 Desempenho vs número médio de exemplos selecionados no *dataset* Reuters. 70
5.13 Desempenho vs número médio de exemplos selecionados no *dataset* BBC Sport. 70
5.14 Desempenho vs número médio de exemplos selecionados no *dataset* Twitter. 71

Lista de Tabelas

2.1 Tabela de equivalência Neurônio Biológico e Neurônio Artificial. Dados extraídos de da Silva et al. (2010). 13
2.2 Tipos de Lanches 14
2.3 Tabela de distribuição de termo frequência (TF). 27
2.4 Matriz de confusão 31
2.5 Tabela da matriz de confusão dos resultados para um classificador binário Cl_A sobre um T_{teste} desbalanceado $dataset_d$. 31
2.6 Tabela da matriz de confusão para um classificador binário Cl_B sobre um T_{teste} balanceado $dataset_b$. 31

5.1 Descrição dos *datasets*. 58
5.2 Distribuição dos exemplos por classes dos *datasets*: BBC SPORT, TWITTER e REUTERS. 59
5.3 Capacidade dos processadores e memórias utilizados nos experimentos. 59
5.4 Valores de k do hiper-parâmetro do KNN selecionados para o *dataset* REUTERS. 60
5.5 Valores de k do hiper-parâmetro do KNN selecionados para o *dataset* BBC SPORT. 61
5.6 Valores de k do hiper-parâmetro do KNN selecionados para o *dataset* TWITTER. 61
5.7 Tabela com os resultados de *f1-score* para o *dataset* REUTERS. . . 62
5.8 Tabela com os resultados de *f1-score* para o *dataset* BBC SPORT. 63
5.9 Tabela com os resultados de *f1-score* para o *dataset* TWITTER. . . 63
5.10 Resultados de *accuracy*, *precision*, *recall* e *f1-score* para as representações BOW e WCD nos *datasets* REUTERS, BBC SPORT e TWITTER. 64
5.11 Resultados de *accuracy*, *precision*, *recall* e *f1-score* para as representações WCD e S-WCD nos *datasets* REUTERS, BBC SPORT e TWITTER. 69
5.12 Número de exemplos selecionados pelo método Seleção de Instâncias e pela representação para cada *dataset* avaliado. 73
5.13 Desempenho de *accuracy* dos métodos de Seleção de Instâncias para todos os *datasets* utilizados 74

Lista de Abreviaturas

AM Aprendizado de Máquina

AVNM *A Voting based Novel Mathematical Rule*

BOW *Bag-of-Words*

DWKNN *New Distance-weightes* k-*Nearest Neighbors*

IA INTELIGÊNCIA ARTIFICIAL

ITML *Information Theoretic Metric Learning*

KL *Kullback-Leibler*

KNN k-*Nearest Neighbors*

LMNN *Large Margin Nearest Neighbor*

LOO *Leave-one-out*

NCA *Neighborhood Component Analysis*

MT *Machine Translate*

PLN Processamento de Linguagem Natural

S-WCD *Supervised Word Centroids Distance*

S-WMD *Supervised Word Mover's Distance*

WCD *Word Centroid Distances*

WMD *Word Mover's Distance*

Lista de Notações

d dimensões

doc vetor que representa um documento no formato nBOW

doc^a vetor que representa o documento a no formato nBOW

doc_i posição i do vetor que representa o documento no formato nBOW

doc_i^a posição i do vetor que representa o documento a no formato nBOW

D Dimensões maiores que d

emb_i vetor de palavra, *word embedding* da palavra i

X conjunto dos exemplos

Y conjunto dos rótulos ou classe de X

x um exemplo do conjunto X

$\mathcal{D}$ uma métrica de distância

$x^{(i)}$ i-ésimo exemplo do conjunto X

$y^{(i)}$ i-ésimo rótulo ou classe do $x^{(i)}$ do conjunto X

x_i i-ésimo atributo do exemplo

$\mathbb{R}$ números reais

EM espaço métrico, um par $(X, \mathcal{D})$

A matriz *Mahalanobis*

M matriz *Mahalanobis*

n número de exemplos

m número de atributos

$p_{i,j}$ probabilidade de cada exemplo selecionar seu vizinho

p_i probabilidade de todo exemplo da mesma classe de i selecionar seu vizinho

T_{treino} conjunto de exemplos de treinamento

T_{teste} conjunto de exemplos de teste

$f(A)$ função de cálculo da matriz A

$g(A)$ função de cálculo da matriz A

$h(x)$ função de hipótese

w matriz ou vetor de pesos da RNA

w^i pesos da i-ésima camada da RNA

$w_{r,i}$ i-ésimo peso do vetor de pesos no neurônio r da RNA

w_i i-ésimo peso do vetor de pesos da RNA

r índice para número de neurônios

u_r potencial de ativação do neurônio r

u_r^i potencial de ativação do neurônio r da camada i

b_r bias do neurônio r

$g(\cdot)$ função de ativação

s é o sinal de saída do neurônio

α é a taxa de aprendizagem

Pr é uma função de probabilidade

N_{cl} quantidade de classes

c índice para classes

I^i vetor intermediário i da RNA

E taxa de erro

V conjunto do das palavras de um corpus ou *dataset*, vocabulário

ρ uma palavra

$w(t)$ função de predição da palavra t

Q_i complexidade do algoritmo i

d_{emb} dimensão do vetor de palavra ou *word embedding*

ctx é o tamanho do contexto (quantidade de palavras ao redor)

VP quantidade de verdadeiro positivo

VN quantidade de verdadeiro negativo

TVP taxa de verdadeiro positivo ou *recall*

TVN taxa de verdadeiro negativo

FN quantidade de falso negativo

FP quantidade de falso positivo

Cl_A classificador A

Cl_B classificador B

S conjunto de exemplos selecionados

$sil(x^{(i)})$ função de *Silhouette* para o exemplo $x^{(i)}$

In lista de índices

str *string* binária do método RMHC

p_r proporção de instâncias selecionadas

$c_{i,j}$ custo de i até j

$\mathcal{C}_i$ cluster i

$\mathcal{C}_j$ cluster j

v_i coordenadas do *cluster* $\mathcal{C}_i$

q_j coordenadas do *cluster* $\mathcal{C}_j$

oc_i é a ocorrência de uma palavra em um documento

T matriz de transporte

λ parâmetro de regularização

$\mathcal{N}_a$ conjunto de documentos mais próximos de um documento doc^a na distância WCD

Lista de Algoritmos

1 Algoritmo de Gradiente Descendente aplicado busca da convergência de hipótese da regressão linear, através da atualização dos pesos w_i da hipótese. 15
2 Algoritmo de Gradiente Descendente aplicado a busca da convergência da hipótese da regressão logística, através da atualização dos pesos w_i da hipótese. 17
3 Algoritmo de Gradiente Descendente aplicado busca da convergência de hipótese da regressão logística, através da atualização dos pesos w_i da hipótese. 18
4 Implementação da fase de treinamento da regressão softmax sobre o *dataset* MNIST. 20
5 Algoritmo CNN. 37
6 Algoritmo ENN. 38
7 Algoritmo RMHC. 39
8 Pseudo algoritmo de seleção de instâncias SILHOUETTE SELECTION. 42
9 Algoritmo de treinamento do S-WMD. 51

CAPÍTULO

1

Introdução

1.1 Histórico

O tratamento de imensos volumes de dados, também conhecido como *big data*, que inclui dados estruturados e não estruturados, dados confiáveis ou não e velocidade de transmissão de dados tem trazido resultados impactantes em diversos domínios de aplicação na academia e também nas corporações (John Walker, 2014). Conhecidos também como os 4 Vs do *big data* (Volume, Velocidade, Variedade, Veracidade), um dos seus grandes desafios está na extração de conhecimento de dados não estruturados como imagens e textos. O reconhecimento de padrões em dados não estruturados por diversos anos foi restrito às técnicas de extração de características manuais. Grandes áreas de pesquisa em Ciência da Computação, como visão computacional, tiveram por diversos anos a proposta de desenvolver novas técnicas de extração de características para imagens.

Com o uso de GPUs e melhor conhecimento dos critérios de convergência de redes neurais, em 2012 foi desenvolvida uma rede neural que chamou a atenção da comunidade de visão computacional no *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) de 2012[1] (Russakovsky et al., 2015). No ILSVRC normalmente participam os grupos de pesquisa que na época detinham o estado da arte em visão computacional mais renomados do mundo. Naquele ano participava uma equipe pouco conhecida denominada `SuperVision`. A equipe era composta basicamente por dois alunos de doutorado, Alex Krizhevsky e Ilya Sutskever, e o seu orientador Geoffrey Hinton, um pesquisador renomado de redes neurais. A equipe `SuperVision` ganhou a competição com desempenho quase duas vezes melhor (erro de 0.15 vs 0.26) que o segundo colocado. Do segundo colocado em diante combinavam diversos algoritmos de extração de características de visão computacional. A ideia da rede neural utilizada pela `SuperVision` não era nova, no trabalho de Hinton e Salakhutdinov (2006) já havia apresentado uma rede muito similar, mas com diversas limitações e dificuldades quanto a inicialização da rede neural.

Intrigado pelo resultado de Krizhevsky et al. (2012), a comunidade científi-

[1] http://www.image-net.org/challenges/LSVRC/2012/results.html

fica de Aprendizado de Máquina (AM) voltou-se em grande parte para o uso desta rede neural. A rede não utilizava técnicas convencionais de extração de características, a rede neural profunda conseguiu aprender de modo não supervisionado filtros que faziam o papel da extração de características. Assim a rede passou a ser utilizada obtendo o estado da arte em diversos domínios como séries temporais, reconhecimento de voz, e texto (Goodfellow et al., 2016).

No domínio de texto, com o avanço das redes neurais profundas, tornou-se possível treinar modelos mais complexos. Uma ideia já bem conhecida pelos pesquisadores de Processamento de Linguagem Natural (PLN) é o aprendizado de vetores que representam palavras nos textos (Elman, 1990; Rumelhart et al., 1986). Assim, no trabalho de Mikolov et al. (2013a) propõem construir vetores utilizando dois modelos: o CONTINUOUS BAG-OF-WORDS (CBOW) e o SKIP-GRAM. Em 2013, essas representações ganharam bastante atenção da comunidade por ir além da representação das palavras utilizadas tradicionalmente. A representação permitia fazer operações algébricas sobre os vetores das palavras. Um exemplo clássico da área é a operação dos vetores: vetor("Rei") - vetor("Homem") + vetor("mulher"). Surpreendentemente o vetor resultante é muito próximo do vetor que representa a palavra "rainha". Esta abordagem atualmente é conhecida como *word embedding*, e esses vetores foram obtidos pela ferramenta do WORD2VEC.

Dois anos mais tarde em 2015 foi proposto um trabalho que utiliza a representação de WORD2VEC para calcular a distância entre documentos textuais (Kusner et al., 2015) denominado de *Word Mover's Distance* (WMD). Esta distância faz uso da proximidade entre as palavras no espaço de representação de WORD2VEC. Na Figura 1.1 está ilustrado um exemplo extraído do artigo original no qual são comparadas as sentenças: "*Obama speaks to the media in Illinois*" e "*The president greets the press in Chicago*". Observe que a sentença tem um significado muito similar, mas em termos de palavras que as compõem são muito diferentes. Como ilustrado na figura, no espaço de WORD2VEC as palavras "*Obama*" e "*President*", "*Speaks*" e "*greets*", "*media*" e "*press*" e "*Illinois*" e "*Chicago*" são bem próximas, o que tornaria a distância entre as duas sentenças também muito próximas.

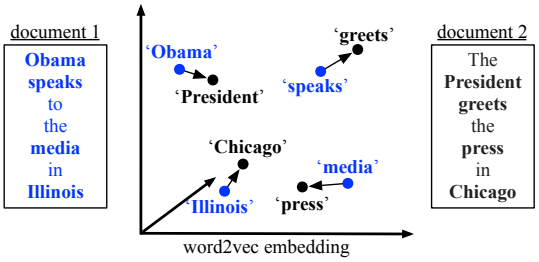


Figura 1.1: Exemplo do funcionamento do WMD. Figura original do artigo (Kusner et al., 2015)

No ano seguinte em 2016 no NIPS foi proposta uma melhoria no WMD utilizando Aprendizado de Métrica denominado de *Supervised Word Mover's Distance* (S-WMD)(Huang et al., 2016). Atualmente é um dos algoritmos estados da arte em classificação de textos (Huang et al., 2016).

Paralelamente estava sendo desenvolvido um trabalho de mestrado no Laboratório de INTELIGÊNCIA ARTIFICIAL que combinava Aprendizado de Métrica

e Seleção de Instâncias (Max, 2016). O trabalho apresentou resultados que melhoravam a Seleção de Instâncias com o uso de Aprendizado de Métrica. Isto se mostrou promissor e pondo em dúvida o que ainda pode ser melhorado com outros métodos de Aprendizado de Métrica para Seleção de Instâncias, como o proporcionado pelo S-WMD. O presente trabalho propõe uma extensão do mesmo explorando o potencial reportado na literatura do S-WMD. Para acrescentar a este trabalho as representações utilizadas nos trabalhos de (Huang et al., 2016) foi necessário comparar estes métodos com as representações. Então, a Seleção de Instâncias é realizada após a transformação dos exemplos em um espaço métrico de representação dos dados textuais.

1.2 Objetivo

O objetivo deste trabalho é avaliar a combinação de S-WMD e Seleção de Instâncias em representações para dados textuais. Assim, o trabalho avalia simultaneamente três técnicas: Seleção de Instâncias, Aprendizado de Métrica e WMD. O desempenho é dividido em três aspectos: o tempo de construção do modelo e classificação, a seleção da menor quantidade de número de exemplos e melhor avaliação de desempenho na classificação. Por último é feito um resumo da melhor combinação entre estes três aspectos.

1.3 Proposta

Os resultados de S-WMD e Max (2016) são promissores e a pergunta que este trabalho procura responder é: a combinação de Seleção de Instâncias, Aprendizado de Métrica e WMD por meio da junção de S-WMD e Seleção de Instâncias traz ganhos?

Ao aplicar transformação linear sobre o espaço de exemplos utilizando Aprendizado de Métrica, os exemplos ficam melhor organizados. Na Figura 1.2 está ilustrado o espaço métrico utilizando t-SNE (Maaten e Hinton, 2008) com WMD e S-WMD. Pode-se observar que as classes ficam melhor separadas, o que pode beneficiar não somente a melhoria do desempenho de classificadores, mas também de métodos de Seleção de Instâncias. Assim, a proposta deste trabalho é avaliar os resultados verificando se o ganho também ocorre para a Seleção de Instâncias ao utilizar a transformação linear proporcionada pelo método.

Esta dissertação está organizada em 6 capítulos. No Capítulo 2 está apresentado os fundamentos e conceitos usados no projeto de mestrado, as dificuldades no Processamento de Linguagem Natural, os algoritmos de Aprendizado de Máquina utilizados como o k-*Nearest Neighbors* e as redes neurais artificiais, que realizam a classificação e extração de *features* dos dados. Também serão discutidas algumas representações textuais e de dados mais conhecidas como o *Bag-of-Words* (BOW) e suas variações, além de técnicas utilizando *deep learning*. No Capítulo 3 são detalhados os algoritmos de Seleção de Instâncias utilizadas neste trabalho. No Capítulo 4 é apresentado o problema do *Earth Mover's Distance* (EMD) e sobre as métricas de distância sem supervisão do WMD e com supervisão do S-WMD. No Capítulo 5 são apresentados os resultados dos experimentos realizados utilizando as representações textuais, técnicas de Seleção de Instâncias e Aprendizado de Métrica de Distância. Por fim, no Capítulo 6 é apresentada as conclusões e trabalhos futuros.

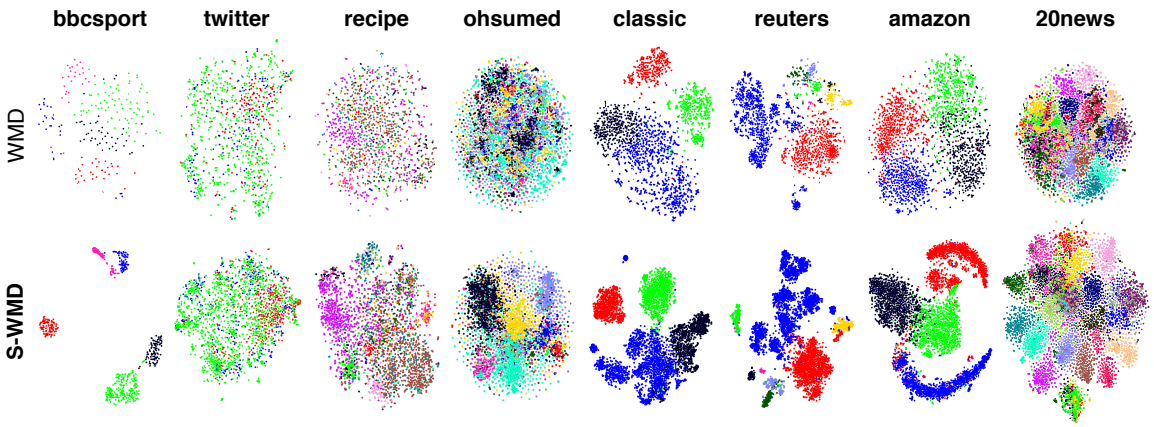


Figura 1.2: Transformação do espaço métrico utilizando S-WMD com datasets benchmark de textos. Figura original do artigo (Huang et al., 2016)

CAPÍTULO

2

Fundamentos e Conceitos

Neste capítulo é levantado os fundamentos e conceitos dos assuntos tratados acerca deste projeto de mestrado para melhor leitura dos capítulos seguintes. Na Seção 2.1 está introduzida sucintamente a origem, definição e os problemas encontrados no Processamento de Linguagem Natural. Na Seção 2.2 são apresentados algoritmos de Aprendizado de Máquina utilizados para as tarefas de classificação e representação textual neste trabalho como o k-*Nearest Neighbors* e as redes neurais. A Seção 2.2.2 apresenta definições e métodos sobre Aprendizado de Métrica de Distância. A Seção 2.3 apresenta o BOW e os *word embeddings*, como são criados e suas propriedades e limitações.

2.1 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) é uma abordagem computacional para analisar textos que é baseada em conjuntos de teorias e tecnologias (Liddy, 2001). Ainda não há uma definição que chegue a um senso comum sobre PLN, entretanto Liddy (2001) fornece uma para apoio desta dissertação:

> "Processamento de Linguagem Natural é uma variedade de técnicas computacionais movida teoricamente para analisar e representar naturais ocorrentes em textos de até um ou mais níveis das analises linguísticas para o propósito de alcançar o processamento de linguagem como um humano para uma variedade de tarefas e aplicações."

O trabalho de processar a linguagem natural é difícil. Para o ser humano aprender uma nova linguagem que não seja a sua nativa requer um grande esforço mesmo com grande capacidade de construções lógicas. Uma máquina encontra diversas dificuldades para processar as informações contidas na linguagem. Entre os problemas mais difíceis em PLN estão (Rocha, 2007):

- Problema na compreensão da linguagem dada como entrada para o algoritmo que dependem do contexto como por exemplo: palavras técnicas, tema do texto e diálogo.
- Ligação entre áreas do conhecimento como Computação, Linguística, Psicologia e Fonética.

Geralmente os problemas da PLN são associados aos algoritmos de INTELIGÊNCIA ARTIFICIAL (IA). Os principais problemas no processamento de linguagem natural formas textuais envolvem ambiguidades. Por vezes são construções simples, porém precisam de um entendimento global do assunto, por exemplo (Vieira e Lima, 2001):

> *O homem viu o menino com o telescópio.*
>
> *Ele entrou na sala de muletas.*

Pode-se retirar interpretações diferentes das frases: o homem que usou o telescópio para ver o menino ou o menino que estava com o telescópio, ou na outra frase, ele entrou na sala usando muletas ou adentrou na sala das muletas. Essas ambiguidades são produzidas não por causa dos significados das palavras, mas pela construção, estrutura que a frase foi feita.

As pesquisas sobre Processamento de Linguagem Natural começaram nas décadas de 1940. O foco era na criação de máquinas de tradução (MT). As MT's eram usadas para quebrar códigos inimigos na Segunda Guerra Mundial. As máquinas de tradução usavam dicionários para realizar a tradução dos documentos, o que as tornavam falhas quando os documentos continham ambiguidades (Liddy, 2001).

Depois desse período, houve a divisão em duas vertentes da PLN, uma passou a considerar teoria de informação estatística e a outra seguiu o campo da linguística (Liddy, 2001), renovado por Chomsky ao publicar Estruturas Sintáticas (Chomsky, 1957) que mais tarde foi transformada em gramática gerativa (Trombley, 2014).

A área de PLN pode ser dividida em 5 níveis (da Silva Brito, 2000):

- nível morfológica: estuda a função das palavras na sua sentença.
- nível lexical: estabelece a correspondência da palavra com o seu significado sua informação.
- nível sintático: refere-se a posição das palavras na sentença.
- nível semântico: faz a ligação entre as estruturas encontradas pelo nível sintático com a realidade.
- nível pragmático: trabalha em um contexto mais amplo para troca de informações.

A computação dos dois primeiros níveis é de possível resolução, mas os demais são o que oferecem as maiores dificuldades e se utilizam vários tipos de tratamento para tentar resolvê-los.

Neste projeto, a vertente de trabalho foca-se à teoria de informação estatística, trabalhando com os vetores de palavras para resolução dos níveis sintático e semântico e no nível pragmático comparando documentos, como será visto nos algoritmos das próximas seções que se apoiam em modelos matemáticos.

2.2 Aprendizado de Máquina

Nesta seção é apresentada dois algoritmos de Aprendizado de Máquina: o k-*Nearest Neighbors* (KNN) na Seção 2.2.1 e as redes neurais na Seção 2.2.4. O KNN é usado na tarefa de classificação de documentos deste projeto de mestrado. O Aprendizado de Métrica de Distância é discutido na Seção 2.2.3 e produz uma função de distância para o KNN. As redes neurais artificiais são explicadas desde a sua intuição e abstração, passando pelos conceitos de regressão linear, logística, PERCEPTRON até PERCEPTRONS multi-camadas. Neste trabalho as redes neurais contribuem para a explicação dos algoritmos de *deep learning* e são descritos na Seção 2.2.4.

2.2.1 k-*Nearest Neighbors*

O algoritmo k-*Nearest Neighbors* foi proposto por Cover e Hart (1967) e é bem conhecido na literatura. Este algoritmo usa um modelo não-paramétrico de classificação que estima um grupo ou classe para cada exemplo (Russell et al., 2003). Assim, não depende, não é caracterizado por um conjunto limitado de parâmetros para classificar, mas possui meta-parâmetros como k e a métrica de distância a ser utilizada.

O KNN considera os k vizinhos mais próximos para estimar a classe de um novo exemplo. A classe escolhida para um exemplo dado acontece através de voto unitário de cada um dos k's exemplos rotulados mais similares (Wu et al., 2008). Muitas vezes se escolhe um número ímpar para k, para que não ocorra empate (Russell et al., 2003).

O algoritmo é fortemente dependente de uma Métrica de Distância para definir os vizinhos mais próximos (Theodoridis e Koutroumbas, 2003; Mitchell, 1997). Uma possível Métrica de Distância é a *Minkowiski* ou com a norma $\mathcal{D}^d$:

$$\mathcal{D}^d(x^{(i)}, x^{(j)}) = (\sum_k |x_k^{(i)} - x_k^{(j)}|^d)^{1/d}. \tag{2.1}$$

sendo $x^{(i)}$ e $x^{(j)}$ são dois exemplos. Para $d = 2$, temos a métrica de distância Euclideana, com $d = 1$, a métrica é de Manhattan (Russell et al. (2003)). O espaço considerado é os atributos dos exemplos, daí se aplica uma das métricas para definir as distâncias e exemplos usados para eleição da classe para o novo exemplo (Theodoridis e Koutroumbas (2003)). Para incluir um novo exemplo, pode ser necessário compará-lo com todos os exemplos já contidos no classificador.

O KNN é um algoritmo de fácil compreensão e implementação. Em muitos casos é considerado de rápido treinamento e possui um bom desempenho. Ele é bem adequado para problemas que envolvem várias classes (Kuramochi e Karypis, 2005; Wu et al., 2008). Na Figura 2.1 é exibido um esquema de como funciona os dados para serem classificados por um k-*Nearest Neighbors* sem transformação no espaço dos exemplos de treinamento. A classificação *Leave-one-out* ocorre pegando cada exemplo de teste e computando a sua distância com todos os elementos do treino, desse modo para cada exemplo de teste tem uma lista de distâncias. Esta lista é ordenada e a partir dela é feita a votação pela classe majoritária até o k-ésimo elemento mais próximo.

O KNN não se adapta muito bem em exemplos com altas dimensões, ou seja, grande quantidade de atributos (Friedman, 1997). Em conjuntos com

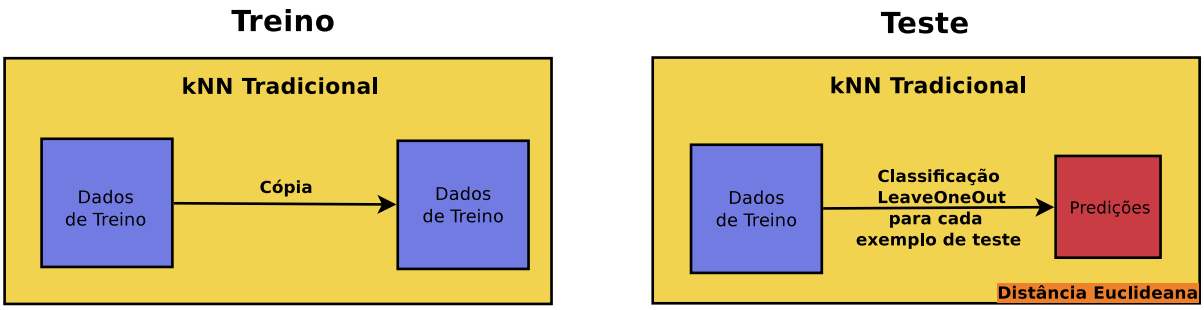


Figura 2.1: Etapas de treino e teste para um kNN sem Aprendizado de Métrica usando a distância Euclideana.

muitos atributos, a distância entre todos os exemplos fica muito parecida, requerendo algum método de redução de dimensão. O algoritmo pode necessitar que seus exemplos sejam normalizados para uma melhor classificação, caso a métrica não faça essa regularização (Ma et al., 2014).

O treinamento é rápido e alguns pesquisadores consideram como inexistente. Em contrapartida a classificação pode ser lenta se o conjunto de treinamento for grande. Outro problema acontece quando se usa um k muito pequeno, o algoritmo pode ficar sensível a ruídos. Em complemento, com um k muito grande, pode dar preferências para outras classes (Wu et al., 2008).

Novas métricas e propostas têm como objetivo romper essas barreiras do kNN. Uma abordagem do problema da sensibilidade é resolvida pelo *New Distance-weightes* k-*Nearest Neighbors* (DWKNN) e demonstra bons resultados em 12 *datasets* (Gou et al., 2012). O algoritmo *A Voting based Novel Mathematical Rule* (AVNM) também usa um modelo não paramétrico como o k-*Nearest Neighbors* para obter o estado da arte para classificação de imagens (Vidyarthi e Mittal, 2016), faz votação ponderada com redução no espaço dos exemplos.

2.2.2 Definições de Aprendizado de Métrica

Métrica é uma função que mapeia dois elementos de um conjunto X para um valor $\mathbb{R}$. Dizemos que $\mathcal{D} : X \times X \to \mathbb{R}_0^+$, sobre X, sendo $\forall x^{(i)}, x^{(j)}, x^{(k)} \in X$, deve seguir 4 propriedades (Weinberger e Saul, 2009):

1. $\mathcal{D}(x^{(i)}, x^{(k)}) + \mathcal{D}(x^{(i)}, x^{(k)}) \leq \mathcal{D}(x^{(i)}, x^{(k)})$
2. $\mathcal{D}(x^{(i)}, x^{(j)}) \geq 0$
3. $\mathcal{D}(x^{(i)}, x^{(j)}) = \mathcal{D}(x^{(j)}, x^{(i)})$
4. $\mathcal{D}(x^{(i)}, x^{(j)}) = 0 \iff x^{(i)} = x^{(j)}$

Dessa forma $\mathcal{D}$ é uma métrica sobre X. A primeira propriedade de uma métrica apresenta a desigualdade triangular; a segunda diz respeito a não negatividade; a terceira e quarta são a simetria e distinguibilidade, respectivamente.

Pseudo-métrica é quando uma métrica possui somente as 3 primeiras propriedades (Weinberger e Saul, 2009).

Distância é a imagem de cada par $\mathcal{D}(x^{(i)}, x^{(j)})$, assim é dito que o valor real resultante da função é a distância entre $x^{(i)}$ e $x^{(j)}$ (Lima, 1983).

Espaço Métrico (EM) é um par $(X, \mathcal{D})$, onde X é um conjunto e $\mathcal{D}$ é uma métrica em X (Lima, 1983).

2.2.3 Aprendizado de Métrica de Distância

Antes de serem calculadas as distâncias entre os elementos do conjunto X, pode-se realizar algumas modificações nos elementos. Transformando linearmente dois elementos de X pela matriz A e em seguida obter a distância Euclideana:

$$x \to Ax \tag{2.2}$$

$$\mathcal{D}_A(x^{(i)}, x^{(j)}) = \left\| A(x^{(i)} - x^{(j)}) \right\|_2^2 \tag{2.3}$$

Outro modo comum de se expressar a distância ao quadrado de A é em termos de M:

$$M = A^T A. \tag{2.4}$$

Qualquer matriz M como na Equação 2.4 possui a garantia que seus valores reais para a matriz A é semi-definida positiva. Reformulando a distância tem-se:

$$\mathcal{D}_M(x^{(i)}, x^{(j)}) = (x^{(i)} - x^{(j)})^T M (x^{(i)} - x^{(j)}) \tag{2.5}$$

A métrica *Mahalanobis* pode ser parametrizada nos termos da matriz A ou M. Isso é usado por muitos pesquisadores como proposta de cálculo de distância para o KNN. Note que quando a matriz é identidade a métrica se torna Euclideana.

Análise de Componentes Vizinhos - NCA

O *Neighborhood Component Analysis* (NCA) é um método de aprender a medida de distância *Mahalanobis* útil em um método de classificação KNN (Goldberger et al., 2005). Segundo os autores é de simples implementação e possui um método efetivo para classificação. A abordagem se aproveita das fronteiras de decisão não lineares do conjunto de dados.

Seja um conjunto de dados rotulados com n exemplos de entrada como vetores $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$ e correspondente a eles suas classes $y^{(1)}, \ldots, y^{(n)}$. O método quer encontrar uma métrica de distância que melhore a performance do KNN. Como não se é conhecida a distribuição das classes, o que se tenta aprimorar é a classificação para cada exemplo, com uma validação cruzada *Leave-one-out* (LOO).

A métrica é restrita ao Aprendizado de Métrica de Distância *Mahalanobis*, que pode ser representada por matrizes simétricas semi-definidas positiva. A métrica é estimada pelas suas raízes quadradas inversas, pelo aprendizado de transformação linear de um espaço de entrada em um espaço transformado, onde o KNN realize uma boa classificação. Seja A a matriz transformação, então a métrica objetiva é:

$$M = A^\top A, \text{ tal que } \mathcal{D}_M(x^{(i)}, x^{(j)}) = (x^{(i)} - x^{(j)})^T M (x^{(i)} - x^{(j)}) \tag{2.6}$$

$$= (Ax^{(i)} - Ax^{(j)})^\top (Ax^{(i)} - Ax^{(j)}). \tag{2.7}$$

Qualquer pequena modificação sobre A pode tornar mais próximo ou afastar a vizinhanças entre os exemplos. A métrica usa uma função de custo diferenciável estocástica para atribuição de vizinhos no espaço transformado. Dessa forma, cada exemplo i seleciona outro exemplo j como seu vizinho com probabilidade p_{ij}, tomando a classe rotulada do exemplo selecionado. Uma normalização SOFTMAX, usando o exponencial $\exp$, é realizada sobre a distância Euclideana e define o valor de p_{ij}:

$$p_{ij} = \frac{\exp(-||Ax^{(i)} - Ax^{(j)}||^2)}{\sum_{k \neq i} \exp(-||Ax^{(i)} - Ax^{(k)}||^2)}, p_{ii} = 0 \tag{2.8}$$

Pode ser computada a probabilidade p_i para o exemplo i que será corretamente classificado, e é escolhida a mesma classe para o exemplo i de acordo com $Y_i = \{j | y^{(i)} = y^{(j)}\}$:

$$p_i = \sum_{j \in Y_i} p_{ij} \tag{2.9}$$

O objetivo é maximizar o número de exemplos corretamente classificados sobre:

$$f(A) = \sum_i \sum_{j \in Y_i} p_{ij} = \sum_i p_i \tag{2.10}$$

Outro modo de maximizar o número de exemplos classificados é usando a Divergência *Kullback-Leibler*, ou somente, Divergência KL:

$$g(A) = \sum_i \log \left(\sum_{j \in Y_i} p_{ij} \right) = \sum_i \log(p_i) \tag{2.11}$$

A expressão que representa o gradiente da Equação 2.11 é:

$$\frac{\partial g}{\partial A} = 2A \sum_i \left(\sum_k p_{ik}(x^{(i)} - x^{(k)})(x^{(i)} - x^{(k)})^\top - \frac{\sum_{j \in Y_i} p_{ij}(x^{(i)} - x^{(j)})(x^{(i)} - x^{(j)})^\top}{\sum_{j \in Y_i} p_{ij}} \right) \tag{2.12}$$

Também é possível fazer redução linear de dimensionalidade com o NCA, seja para economia computacional ou para regularização de algum algoritmo posterior de aprendizagem. A técnica de redução linear de dimensionalidade, onde se aplicam operações lineares sobre os dados originais para obter uma representação reduzida, são rápidas e relativamente imunes a *overfitting* (Goldberger et al., 2005). Uma matriz não quadrada de dimensões $d \times D$ pode restringir a matriz A do método NCA. Ao fazer, o método por fornecer outros benefícios, por exemplo, para um $d \ll D$ pode-se reduzir o tempo de computação do KNN e o custo de armazenamento, ou para um $d = 2$ ou $d = 3$ pode-se visualizar os dados rotulado em uma baixa dimensão. O algoritmo continua sendo o mesmo, realiza-se uma otimização sobre a função de custo

da Equação 2.10 usando um gradiente descendente sobre a matriz não quadrada. A dimensão da matriz A para redução deve ser configurada pelo usuário.

Uma abordagem que também tenta aprender a métrica distância *Mahalanobis* é o Vizinho Mais Próximo de Larga Margem (LMNN). O foco do LMNN é melhorar a *accuracy* de classificação do KNN (Weinberger e Saul, 2009). Para isso, o método conta com três fatores principais: sua função convexa de decaimento, seu objetivo de maximização de margem e as restrições sobre a métrica de distância para melhoria da *accuracy* do KNN.

A intuição ou idealização que levou a esse método é que cada exemplo de treinamento deve compartilhar a mesma classe com seu k vizinho e que exemplos com diferentes classes devem ser separadas. O que é desejado fazer é isso, criar uma transformação linear que cumpra as duas idealizações do método (Weinberger e Saul, 2009).

Outro método também para o um Aprendizado de Métrica de Distância *Mahalanobis* conhecido na literatura é o *Information Theoretic Metric Learning* (ITML). O ITML é um método rápido e escalável. Ele não requer uma computação de autovetores e autovalores (Davis et al., 2007).

2.2.4 Redes Neurais

A partir da observação do funcionamento de um neurônio, em Rosenblatt (1958) foi proposto um modelo matemático para o armazenamento de informações e que simultaneamente também poderia ser estimulado para ativar respostas. Esta representação foi denominada de PERCEPTRON e buscava simular comportamento de um neurônio biológico. Na Figura 2.2 é ilustrado um neurônio biológico que pode ser separado em três partes: (1) os dendritos, (2) o corpo celular (ou ainda, soma) e o (3) axônio (da Silva et al. (2010)).

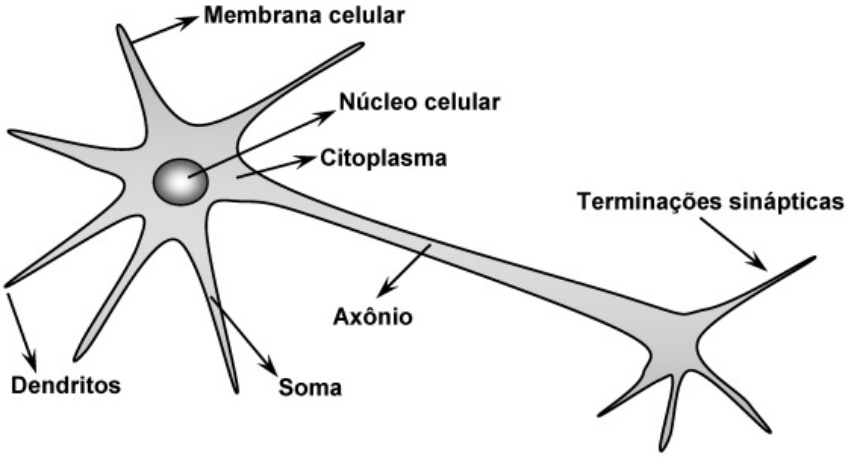


Figura 2.2: Representação simplificada de um neurônio extraído de da Silva et al. (2010).

Nos dendritos são captados os estímulos de outros neurônios e funcionam como conectores. O corpo celular é responsável por processar todas as informações que passam por ele e irá produzir um potencial de ativação. Com este potencial poderá impulsionar uma energia ao longo do seu axônio. O axônio irá repassar o impulso elétrico para outro neurônio ou algum tecido muscular. Na terminação de um axônio possuem ramificações conhecidas por terminações sinápticas, onde as trocas de informações são feitas.

A modelagem mais utilizada de um neurônio humano para um artificial é representado na Figura 2.3. Repare que na figura os pesos $w_{r,1}, w_{r,2}, \ldots, w_{r,m}$ tem o índice r, pois pertencem ao neurônio r, assim como o sinal s e u_r. Neste modelo os sinais (representam os estímulos que chegam aos dendritos do neurônio humano) que chegam até o neurônio r são representados pelo conjunto $x = \{x_1, x_2, \ldots, x_m\}$. Os pesos das uniões são representados pelo conjunto de pesos sinápticos $w = \{w_1, w_2, \ldots, w_m\}$. A modelagem também possui: combinador linear, limiar de ativação, mais conhecido como bias, potencial de ativação, função de ativação e sinal de saída.

O combinador linear realiza a junção dos sinais com os pesos, resultando num potencial de ativação. O limiar de ativação é uma variável que especifica o nível adequado para ser disparado para a saída do neurônio junto com o potencial de ativação (da Silva et al., 2010). Dessa forma, o potencial de ativação é o resultado da diferença do valor do combinador linear com o limiar de ativação. A função de ativação serve para regular o intervalo do valor produzido pelo potencial de ativação. Por fim, o sinal de saída é o resultado do potencial de ativação aplicado na função de ativação.

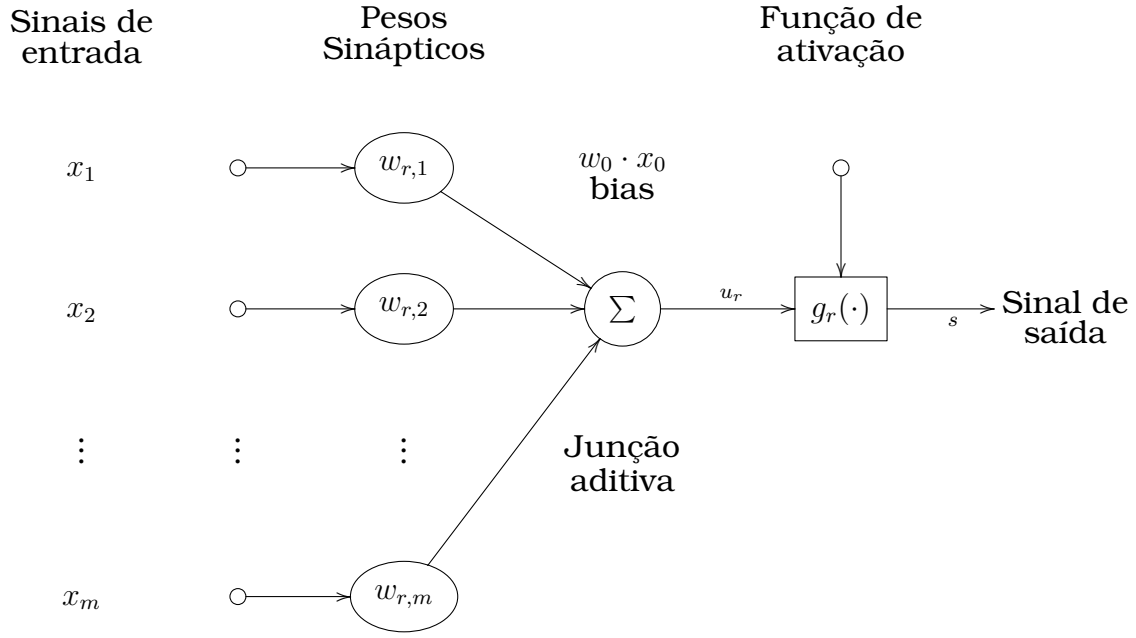


Figura 2.3: Modelo matemático de um neurônio.

As fórmulas que representam o cálculo de um sinal de saída para um neurônio r:

$$u_r = \sum_{j=1}^{m} w_{r,j} \cdot x_j - w_0 \cdot x_0 \tag{2.13}$$

$$s = g_r(u_r) \tag{2.14}$$

onde u_r é o potencial de ativação, $w_{r,j}$ e x_j são os pesos e sinais de entrada, o bias é representado por b_r em outros projetos, e seu cálculo: $b_r = w_0 \cdot x_0$, s é o sinal de saída e $g(\cdot)$ é a função de ativação.

Na Tabela 2.1 está representada a equivalência dos modelos nas três partes principais do neurônio biológico.

Neurônio Biológico	Neurônio Artificial	Explicação
Dentritos	Pesos sinápticos	Cada dentrito pode ter uma espessura específica e quanto maior a espessura, mais forte é sinal passado por ele. O peso sináptico age de maneira similar, quanto maior mais "forte" é o sinal de x_j.
Corpo Celular	Junção Aditiva	O corpo celular faz a união dos sinais depois de terem passado pelo dentrito. A junção aditiva faz a soma dos valores de x_j multiplicados pelos pesos $w_{r,j}$ que é equivalente ao que o corpo celular faz.
Axônio	Função Ativação	O axônio reduz a força do sinal que vem do corpo celular. A função de ativação atenua o valor de $u_r \in \mathbb{R}$ para um valor em um intervalo menor como 0 e 1 (função sigmóide), ou -1 e 1 (função tangente hiperbólica)

Tabela 2.1: Tabela de equivalência Neurônio Biológico e Neurônio Artificial. Dados extraídos de da Silva et al. (2010).

Apesar de sua grande utilidade na época, o PERCEPTRON não era capaz de aprender conceitos mais complexos, restringindo-se a problemas de separação linear e "problemas de brinquedo" (*toy problems*).

A combinação de diversos PERCEPTRONS somente foi possível quase 30 anos depois com o trabalho de Rumelhart et al. (1985) que mostra como fazer a propagação de erros em múltiplas camadas, produzindo assim o que hoje é conhecido como PERCEPTRON de múltiplas camadas (*Multi-Layer* PERCEPTRON- MLP) constituindo uma rede de PERCEPTRONS (neurônios) surgindo assim o conceito de redes neurais.

Com o amadurecimento do conceito de redes neurais, pode-se identificar raízes destes modelos em diversas áreas como: neurociência, física, matemática, estatística, ciência da computação e engenharia (Haykin, 2000). As redes neurais artificiais (RNA) surgiram pelo reconhecimento de que o cérebro humano processa informações de uma forma diferente de um computador digital. O cérebro humano possui a capacidade de processar informações complexas de modo paralelo, que o torna rápido em obter respostas.

No começo da década de 90 as Máquinas de Vetores de Suporte (SVM) (Cortes e Vapnik, 1995) começaram a apresentar resultados melhores que a maioria dos algoritmos desenvolvidos até então, tornando SVM o estado da arte em Aprendizado de Máquina, por vários anos. Um grande esforço da comunidade científica foi dispendido em SVM e deixando muitas vezes as redes neurais como mais um algoritmo para comparação em benchmarks.

As redes neurais voltaram a ser um tópico de importância em AM com o desenvolvimento das redes neurais profundas (*deep learning*) (LeCun et al., 2015a) que atualmente são os algoritmos estado da arte em diversas tarefas como reconhecimento de voz, reconhecimento visual de objetos, proces-

samento de linguagem natural, entre muitos outros. O que torna as redes neurais profundas interessantes são os extratores de características (*feature extractors*) que são obtidas de maneira não supervisionada. Neste trabalho algumas ideias baseadas em redes neurais profundas são utilizadas para a extração automática de características de palavras como informações sintáticas e semânticas. Deste modo a compreensão dos fundamentos de redes neurais são de grande importância para este trabalho e nas seções seguintes são apresentados conceitos fundamentais para a compreensão de redes neurais profundas.

Regressão Logística

Uma das maneiras de melhor entender o funcionamento de um PERCEPTRON é o estudo da Regressão Logística. A Regressão Logística é um modelo estatístico que pode ser reduzido a um PERCEPTRON. Para facilitar a compreensão da Regressão Logística, esta seção inicia a explicação de regressores lineares que depois são modificados para Regressão Logística, sendo esta um tipo de regressão linear.

A regressão linear estima funções lineares que melhor se ajustam a um conjunto de pontos (Friedman et al., 2001). A função linear é expressa pela combinação linear dos valores dos atributos e um conjuntos de pesos.

$$h(x) = w_0 \cdot x_0 + w_1 \cdot x_1 + ... + w_m \cdot x_m \tag{2.15}$$

$$h(x) = w^T \cdot x \tag{2.16}$$

A regressão linear procura encontrar os pesos w_j de modo a minimizar uma função custo:

$$(h(x) - y)^2 \tag{2.17}$$

onde y é o valor desejado como conceito classe.

Para exemplificar, considere os lanches apresentados na Tabela 2.2 que possuem como atributos: quantidade de pão, queijo, presunto, orégano e o atributo classe valor do lanche.

	Pão (g)	Queijo (g)	Presunto (g)	Orégano (g)	Valor (R$)
Pão de queijo	100	50	0	0	2,00
Pão de presunto	100	0	50	0	1,50
Misto	100	50	50	15	2,50

Tabela 2.2: Tipos de Lanches

O problema é encontrar uma função que dado os ingredientes seja possível estimar o preço de um novo lanche baseado nas observações da Tabela 2.2. A função que estima o valor do lanche pode ser uma combinação linear de pesos e a quantidade de ingredientes conforme descrito na Equação 2.18.

$$h(x_1, x_2, x_3, x_4) = w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + w_3 \cdot x_3 + \cdot w_4 \cdot x_4 \tag{2.18}$$

As variáveis $w_1, \ldots, w_4$ descrevem o preço por grama do respectivo produto. As variáveis $x_1, \ldots, x_4$ descrevem a quantidade do produto usado para o lanche e w_0 representa o lucro obtido sobre lanche. Generalizando para m atributos

com $x = \{x_0, x_1, \ldots, x_m\}$ e $w = \{w_0, w_1, \ldots, w_m\}$ obtém-se as Equações 2.15 e 2.16. Note que por simplicidade $x_0 = 1$ e w_0 representa o bias e simbolizado por b em outras pesquisas e na Equação 2.13.

A regressão linear pode ser estimada de maneira algébrica ou iterativa. Para a solução iterativa, uma das maneiras para encontrar os pesos w que melhor estimam o valor de y é utilizando a ideia de gradiente descendente.

Inicialmente os pesos w da Equação 2.18 são iniciados com valores aleatórios. Logo em seguida se aplica o algoritmo de gradiente descendente, que realiza sucessivas atualizações nos pesos. As atualizações são feitas com a derivada da função de custo. A função de custo é definida pela seguinte Equação 2.19:

$$J(w_1, w_2, w_3, w_4) = \frac{1}{2m} \sum_{i=1}^{m} (h(x^{(i)}) - y^{(i)})^2 \tag{2.19}$$

onde a função J representa a média da diferença dos erros quadráticos entre o valor estimado e o verdadeiro valor para todos os m exemplos.

O termo $x^{(i)}$ simboliza um exemplo do conjunto de treinamento na linha i da representação de tabela atributo valor e $y^{(i)}$ o valor do atributo classe para o exemplo $x^{(i)}$. O algoritmo de gradiente descendente, ajusta os pesos até um determinado número de iterações ou até a convergência dos pesos.

A variável α é a taxa de aprendizagem e $\frac{\partial J(w)}{\partial w_i}$, a derivada da função custo, é:

$$\frac{\partial J(w)}{\partial w_j} = \frac{1}{m} \sum_{i=1}^{m} ((h(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}) \tag{2.20}$$

Algoritmo 1: Algoritmo de Gradiente Descendente aplicado busca da convergência de hipótese da regressão linear, através da atualização dos pesos w_i da hipótese.

Entrada: w, o conjunto de pesos da hipótese; α, o parâmetro de aprendizado; ϵ número de iterações; o conjunto de treinamento T_{treino}.
Saída: w, pesos da hipótese.

```
it = 0;
enquanto it < ε faça
    para todo (x^(i), y^(i)) ∈ T_treino faça
        h(x^(i)) = w^T · x^(i);
        para todo j ∈ {1, ..., m} faça
            w_j ← w_j − α · ((h(x^(i)) − y^(i)) · x_j^(i));
        fim
    fim
    it = it + 1;
fim
```

A **Regressão Logística** é um processo de ajuste de pesos do modelo de classificação linear para minimizar a perda em um conjunto de dados. A Regressão Logística também usa o cálculo da descida pelo gradiente para encontrar um valor ótimo para os pesos. Ela se baseia na função logística (Russell et al., 2003).

A Função 2.21 define a função logística e seu gráfico está exibido na Figura 2.4, em ambas é atribuído a w o valor 1. O valor de w define diretamente a inclinação da função logística sobre o eixo da abscisa, com o ponto de inflexão no eixo da ordenada em 0.5.

$$g(w^T \cdot x) = \frac{1}{1 + e^{-w^T \cdot x}} \tag{2.21}$$

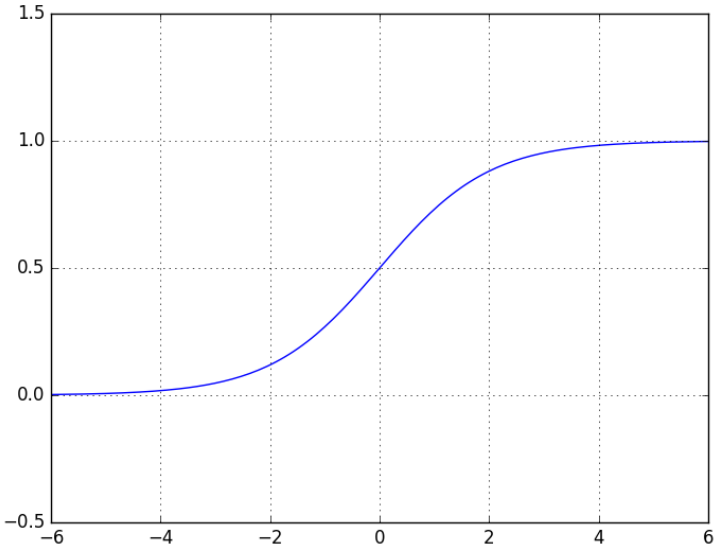


Figura 2.4: Função logística para $w = 1$.

Para demonstrar seu funcionamento para indução da hipótese, o cálculo para um único exemplo do conjunto de treinamento é realizado da seguinte forma (Russell et al., 2003).

$$\frac{\partial}{\partial w_j} \mathbf{J}\ (w) = \frac{\partial}{\partial w_j}(y - g(w \cdot x))^2 \tag{2.22}$$

$$= 2(y - g(w \cdot x)) \times \frac{\partial}{\partial w_j}(y - g(w \cdot x))) \tag{2.23}$$

$$= 2(y - g(w \cdot x)) \times g'(w \cdot x) \times \frac{\partial}{\partial w_j}(w \cdot x)) \tag{2.24}$$

$$= 2(y - g(w \cdot x)) \times g'(w \cdot x) \times x_j^{(i)} \tag{2.25}$$

onde g' é a derivada da função logística:

$$g'(w \cdot x) = g(w \cdot x)(1 - g(w \cdot x)) = g(1 - g(w \cdot x)) \tag{2.26}$$

finalmente, cada peso é minimizado pelo algoritmo de gradiente descendente como é exibido no Algoritmo 2.

Outro modo de fazer um treinamento com a Regressão Logística é através do seu conjunto de treinamento: $T_{treino} = \{(x^{(0)}, y^{(0)}), \cdots, (x^{(n-1)}, y^{(n-1)})\}$ rotulada de n exemplos com atributos das entradas $x^{(i)} \in \mathbb{R}^m$, onde m é quantidade de atributos. Os rótulos são $y^{(i)} \in \{0, 1\}$. Os parâmetros do modelo w são

Algoritmo 2: Algoritmo de Gradiente Descendente aplicado a busca da convergência da hipótese da regressão logística, através da atualização dos pesos w_i da hipótese.

Entrada: w, o conjunto de pesos da hipótese; α, o parâmetro de aprendizado; ϵ número de iterações; o conjunto de treinamento T_{treino}.

Saída: w, pesos da hipótese.

$it = 0$;
enquanto $it < \epsilon$ **faça**
  para todo $(x^{(i)}, y^{(i)}) \in T_{treino}$ **faça**
    $g(w^T \cdot x^{(i)}) = \frac{1}{1+e^{-w^T \cdot x^{(i)}}}$;
    para todo $j \in \{1, ..., m\}$ **faça**
      $w_j \leftarrow w_j - \alpha(y - g(w^T \cdot x^{(i)})) \cdot g(1 - g(w^T \cdot x^{(i)})) \cdot x_j^{(i)}$;
    fim
  fim
  $it = it + 1$;
fim

treinados para minimizar a função de custo:

$$J(w) = \frac{1}{n}\left[\sum_{i=1}^{n} y^{(i)} \cdot \log(g(w \cdot x^{(i)})) + (1 - y^{(i)}) \cdot \log(1 - g(w \cdot x^{(i)}))\right] \tag{2.27}$$

onde $\frac{\partial J(w)}{\partial w_i}$ é:

$$\frac{\partial J(w)}{\partial w_i} = \frac{1}{n}\sum_{i=1}^{n}((g(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}) \tag{2.28}$$

essa função irá atualizar os pesos da nossa hipótese, como no Algoritmo 2. Obtendo a nova atualização para o algoritmo de gradiente descendente, apresentado no Algoritmo 3.

Regressão Softmax

A REGRESSÃO SOFTMAX é a generalização da Regressão Logística para classificação, onde podem existir mais de duas classes, ou seja, mais de duas possibilidades de encontrar a classes correta para algum exemplo.

Seja o conjunto de treinamento $T_{treino} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$ com n exemplos rotulados. Dado um exemplo, estimamos a sua probabilidade por $Pr(y = c|x)$ para cada valor de $c = 1, \ldots, N_{cl}$, onde N_{cl} é o número de classes possíveis. Assim, em uma hipótese generalista irá produzir a seguinte fórmula para N_{cl} classes:

$$h_w(x^{(i)}) = \begin{bmatrix} Pr(y^{(i)} = 1|x^{(i)}; w) \\ Pr(y^{(i)} = 2|x^{(i)}; w) \\ \vdots \\ Pr(y^{(i)} = N_{cl}|x^{(i)}; w) \end{bmatrix} = \frac{1}{\sum_{c=1}^{N_{cl}} \exp^{w_c^T x^{(i)}}} \begin{bmatrix} \exp^{w_1^T x^{(i)}} \\ \exp^{w_2^T x^{(i)}} \\ \vdots \\ \exp^{w_{N_{cl}}^T x^{(i)}} \end{bmatrix} \tag{2.29}$$

Algoritmo 3: Algoritmo de Gradiente Descendente aplicado busca da convergência de hipótese da regressão logística, através da atualização dos pesos w_i da hipótese.

Entrada: w, o conjunto de pesos da hipótese; α, o parâmetro de aprendizado; ϵ, número de iterações; o conjunto de treinamento T_{treino}.

Saída: w, um conjunto de pesos da hipótese.

```
1  it = 0;
2  enquanto it < ϵ faça
3  |  para todo (x^(i), y^(i)) ∈ T_treino faça
4  |  |  g(w^T · x^(i)) = 1/(1+e^(−w^T·x^(i)));
5  |  |  para todo j ∈ {1, ..., m} faça
6  |  |  |  w_j = w_j − α · ((g(x^(i)) − y^(i)) · x_j^(i));
7  |  |  fim
8  |  fim
9  |  it = it + 1;
10 fim
```

onde $w_1, w_2, \ldots, w_{N_{cl}} \in \mathbb{R}^{N_{cl}}$ são os parâmetros do nosso modelo. Nota-se que o termo $\frac{1}{\sum_{c=1}^{N_{cl}} \exp^{w_c^T x^{(i)}}}$ normaliza a distribuição em 2.29, então isto soma-se 1.

A função de custo para esse modelo é uma generalização da Regressão Logística semelhante a Função 2.27, a expressão ainda possui um termo que funciona como uma condição, caso $\{y^{(i)} = c\}$ então o $\log$ de $h_w(x^{(i)})$ será adicionado ao somatório mais interno, caso contrário não.

$$J(w) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{c=1}^{N_{cl}} 1\left\{y^{(i)} = c\right\} \log \frac{\exp^{w_c^T x^{(i)}}}{\sum_{l=1}^{N_{cl}} \exp^{w_l^T x^{(i)}}}\right] \tag{2.30}$$

Outra diferença está na soma de todas as possíveis c's classes da classe rotulada, veja que agora:

$$Pr(c^{(i)} = c|x^{(i)}; w) = \frac{\exp^{w_c^T}\ x^{(i)}}{\sum_{l=1}^{N_{cl}} \exp^{w_l^T}\ x^{(i)}} \tag{2.31}$$

Ainda não existe uma fórmula fechada para o calculo do mínimo de $J(w)$, então para se obter uma solução ótima dessa fórmula usa-se também um algoritmo como gradiente descendente ou L-BFGS (*Limited memory Broyden-Fletcher-Goldfarb-Shanno algorithm*, um método iterativo para resolver problemas não lineares). Na Equação 2.32 é apresentada a fórmula com o gradiente.

$$\nabla_{w_c} J(w) = -\frac{1}{n}\sum_{i=1}^{n}\left[x^{(i)}\left(1\{y^{(i)} = c\} - Pr(y^{(i)} = c|x^{(i)}; w)\right)\right] \tag{2.32}$$

Para penalizar os grandes valores dos parâmetros há um termo de decaimento na função de custo:

$$J(w) = -\frac{1}{n}\left[\sum_{i=1}^{n}\sum_{c=1}^{N_{cl}} 1\left\{y^{(i)} = c\right\} \log \frac{\exp^{w_c^T x^{(i)}}}{\sum_{l=1}^{N_{cl}} \exp^{w_l^T x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{i=1}^{N_{cl}}\sum_{c=0}^{m} w_{ic}^2 \tag{2.33}$$

e para aplicar a um algoritmo de otimização deve-se fazer uma derivada sobre essa nova função de custo:

$$\nabla_{w_c} J(w) = -\frac{1}{n}\sum_{i=1}^{n}\left[x^{(i)}(1\{y^{(i)} = c\} - Pr(y^{(i)} = c|x^{(i)}; w))\right] + \lambda\theta_c \tag{2.34}$$

A atualização de pesos para um exemplo fica:

$$w_c^{(nova)} \leftarrow w_c^{(antiga)} - \alpha\nabla_{w_c} J(w)(\text{para cada } c = 1, \ldots, N_{cl}). \tag{2.35}$$

Para exemplificar o modelo dessa Regressão, a aplicação sobre o *dataset* MNIST, com imagens de números de 0 até 9 rotulados, um *dataset* para identificação de dígitos a partir de imagens (LeCun et al. (1995, 1998)), utilizando sua estrutura neural para esse problema e as fórmulas envolvidas.

Em uma REGRESSÃO SOFTMAX o objetivo é a classificação de múltiplas classes, no caso da classificação de dígitos a partir de imagens, temos 10 classes, um para cada algorismo conhecido $c \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Primeiramente, o *dataset* é organizado para ser processado pela REGRESSÃO SOFTMAX. Logo em seguida realiza-se a passagem dos exemplos do conjunto de treinamento pelo neurônio e na sequência as atualizações dos pesos são realizadas. É exibido no Algoritmo 4 a implementação algorítmica da REGRESSÃO SOFTMAX aplicado no problema de reconhecimento de imagens.

Nas linhas 5 à 6 do Algoritmo 4, cada exemplo é passado pela matriz de pesos transposta, fazendo uma multiplicação de matrizes. O resultado disso são vetores alocados em $logits$. Nas linhas 8 à 15, é o processo de *Softmax*, transformando cada vetor de $logits$, em uma distribuição de probabilidade. Nas linhas 17 até a 25 é calculada a perda ou erro da regressão, uma função chamada soma realiza a soma do vetor que é o produto de $y^{(i)}$ e $\log(logits_i)$. Nas linhas 26, 27, 28 e 29 é calculado o termo de decaimento. Das linhas 30 à 35 é realizado cálculo do gradiente descendente somado ao termo de decaimento, logo em seguida a atualização dos pesos de w. A linha 31 possui a subtração de vetor $y^{(i)}$, numa representação conhecida por *one hot label*, pelo vetor $logits_i$, seguida da multiplicação de matriz por $x^{(i)}$ transposto.

PERCEPTRON *multi-camadas*

Para formar uma rede neural é necessário de mais neurônios, formando ligações e estabelecendo relações entre eles. Um conjunto de neurônios forma uma camada. As camadas podem ser classificadas em de entrada, escondida e de saída. A camada de entrada recebe os sinais que chegam a rede, geralmente número reais. A camada escondida, oculta, ou ainda intermediária,

Algoritmo 4: Implementação da fase de treinamento da regressão softmax sobre o *dataset* MNIST.

Entrada: o conjunto de treinamento T_{treino}; ϵ é a quantidade de épocas de treinamento; Um parâmetro λ; Um parâmetro α.

Resultado: Pesos W.

$w_{ic} = pesos\_aleatórios(), i \in \{1, \ldots, 784\}, c \in \{1, \ldots, 10\}$;
$n = |T_{treino}|$;
$epocas = 0$;
enquanto $epocas < \epsilon$ **faça**
  para todo $(x^{(i)}, y^{(i)}) \in T_{treino}$ **faça**
    $logits_i = w^T \cdot x^{(i)}$;
  fim
  para todo $l_i \in logits$ **faça**
    $total = 0$;
    para todo $k_c \in l_i$ **faça**
      $total = total + k_c$;
    fim
    para todo $k_c \in l_i$ **faça**
      $k_c = \frac{e^{k_c}}{total}$;
    fim
  fim
  $cross = 0$;
  para todo $i \in \{1, \ldots, n\}$ **faça**
    $cross = cross + soma(y^{(i)} * \log(logits_i))$;
  fim
  $somatorio = 0$;
  para todo $w_{ic} \in w$ **faça**
    $somatorio = somatorio + w_{ic} * w_{ic}$;
  fim
  $perda = -\frac{cross}{n} + \frac{\lambda * somatorio}{2}$;
  $dec_{i,c} = 0, i \in \{1, \ldots, 784\}, j \in \{1, \ldots, n\}$;
  para todo $w_{ic} \in W$ **faça**
    $dec_{ic} = \lambda * w_{ic}$;
  fim
  para todo $i \in \{1, \ldots, n\}$ **faça**
    $grad_i = x^{(i)T} \cdot (y^{(i)} - logits_i) + dec_i$;
  fim
  para todo $w_{ic} \in W$ **faça**
    $w_{ic} = w_{ic} - \alpha * grad_{ic}$;
  fim
  $epocas = epocas + 1$;
fim

elas recebem os valores da camada de entrada, cada neurônio calcula sua a soma e o seu potencial de ativação, resultando em cada respectivo sinal de saída através da função de ativação, que é propagado para a próxima camada. Uma rede pode conter uma ou mais camadas escondidas. Na Figura 2.5 é exibido um modelo de rede neural com uma camada de entrada, duas camadas escondidas. A última camada é chamada de camada de saída. Neste projeto os pesos das camadas são simbolizados por $w^{(l)}$, sendo l, o número da camada.

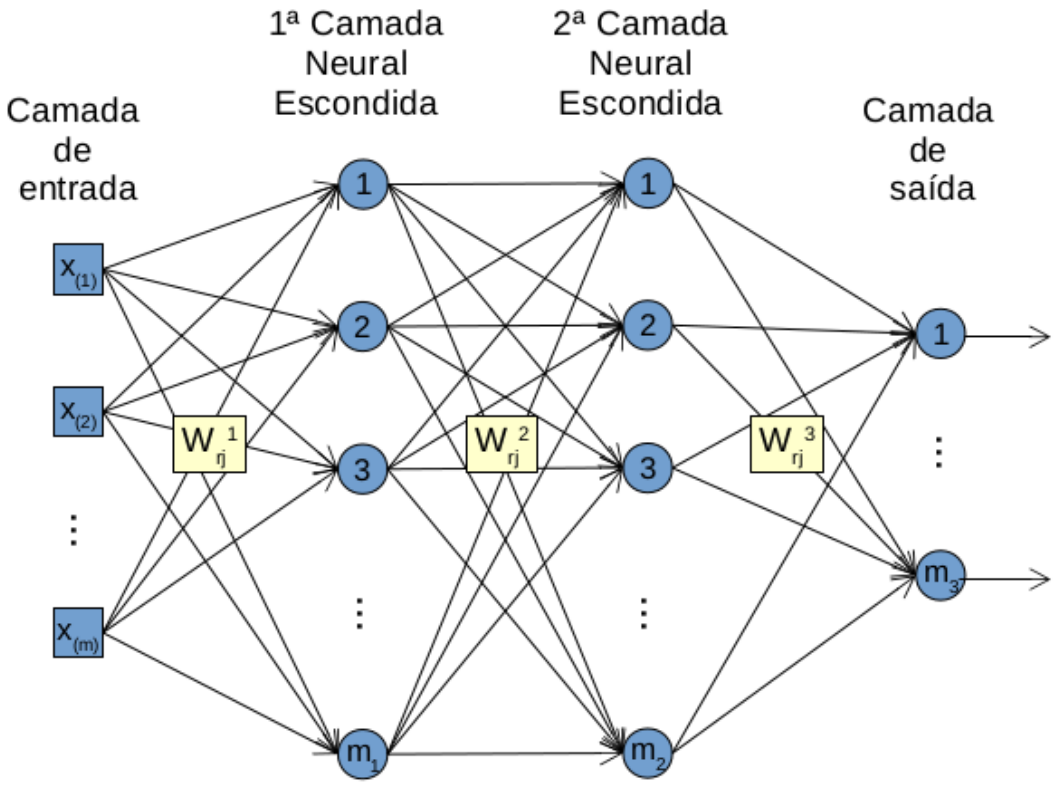


Figura 2.5: Representação de uma rede neural.

Existem diversos tipos de funções de ativação sendo algumas parcialmente diferenciáveis, como por exemplo as funções degrau (2.6a), ReLU (2.6f), bipolar (2.6b) e a rampa simétrica (2.6c); e também as totalmente diferenciáveis como as funções logística (2.6d), tangente hiperbólica (2.6e) e linear.

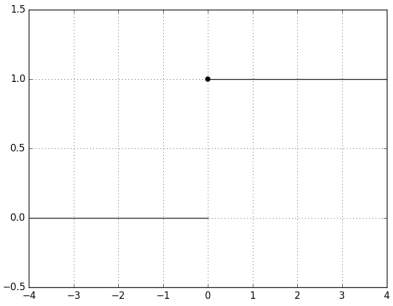
Na Figura 2.6 é exibido uma Rede Neural Artificial Perceptron de Múltiplas Camadas (PMC) e sobre ele será exemplificado o seu treinamento e o algoritmo *backpropagation* para atualização de pesos. As matrizes (2.36, 2.37, 2.38) correspondem aos pesos das respectivas camadas da Rede na figura (2.6):

$$w^1 = \begin{bmatrix} 0.2 & 0.4 & 0.5 \\ 0.3 & 0.6 & 0.7 \\ 0.4 & 0.8 & 0.3 \end{bmatrix} \tag{2.36}$$

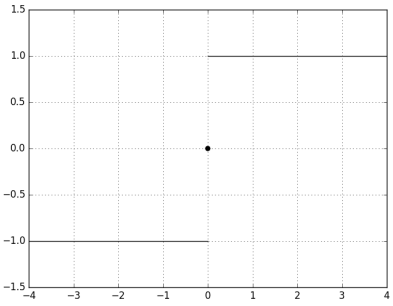
$$w^2 = \begin{bmatrix} -0.7 & 0.6 & 0.2 & 0.7 \\ -0.3 & 0.7 & 0.2 & 0.8 \end{bmatrix} \tag{2.37}$$

$$w^3 = \begin{bmatrix} 0.1 & 0.8 & 0.5 \end{bmatrix} \tag{2.38}$$

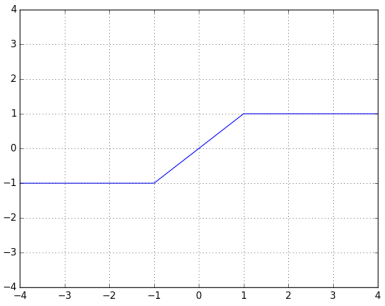
Os procedimentos aplicados sobre a Rede Neural Artificial PMC são a fase de treinamento e a fase de operação. Tanto no algoritmo de treinamento



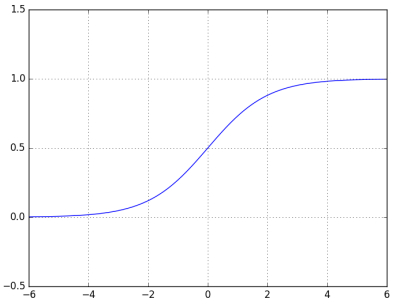
(a) Função Degrau.



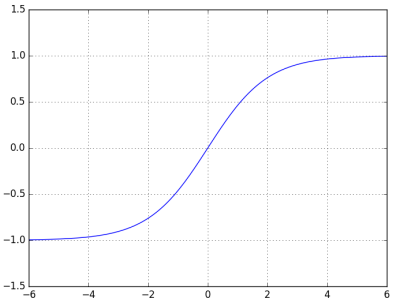
(b) Função Degrau Bipolar.



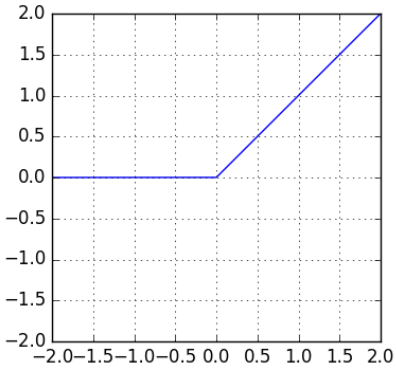
(c) Função Rampa Simétrica.



(d) Função Logística.



(e) Função Hiperbólica.



(f) Função ReLU.

quanto no de operação a rede é alimentada para frente, conhecida por *feed-forward*, os dados que chegam até a Rede pela camada de entrada e vão sendo calculadas para as camadas ocultas processarem até a camada de saída. Um exemplo $x^{(i)}$ é dado para a rede, logo a camada de entrada será $w^1 \cdot x^{(i)}$:

$$x^{(i)} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 0.3 \\ 0.7 \end{bmatrix} \tag{2.39}$$

$$u_r^1 = w^1 \cdot x^{(i)} = \begin{bmatrix} 0.2 \cdot (-1) + 0.4 \cdot 0.3 + 0.5 \cdot 0,7 \\ 0.3 \cdot (-1) + 0.6 \cdot 0.3 + 0.7 \cdot 0,7 \\ 0.4 \cdot (-1) + 0.8 \cdot 0.3 + 0.3 \cdot 0,7 \end{bmatrix} = \begin{bmatrix} 0.27 \\ 0.37 \\ 0.05 \end{bmatrix} \tag{2.40}$$

A função de ativação desta RNA é a logística: $g(x^{(i)}) = \dfrac{1}{1 + e^{-w \cdot x^{(i)}}}$. Em seguida, aplica-se a função de ativação:

$$g(u_j^1) = \begin{bmatrix} g(0.27) \\ g(0.37) \\ g(0.05) \end{bmatrix} = \begin{bmatrix} 0.567 \\ 0.591 \\ 0.512 \end{bmatrix} \tag{2.41}$$

O resultado é passado para a próxima camada com valor -1 para o bias, o

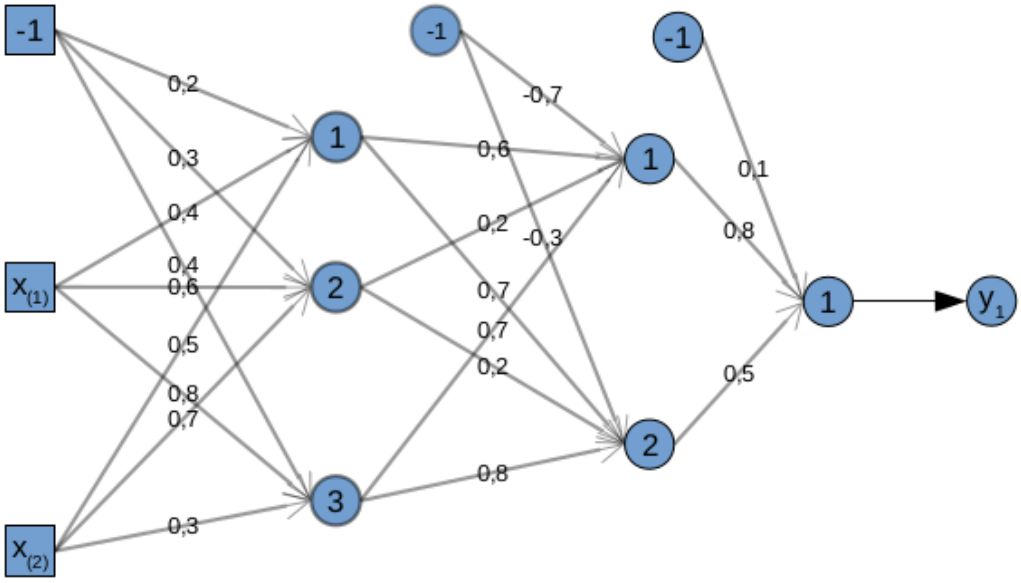


Figura 2.6: Representação de uma Rede Neural Artificial PMC.

vetor intermediário é $I^1 = g(u_r^1)$:

$$I^1 = \begin{bmatrix} -1 \\ 0.567 \\ 0.591 \\ 0.512 \end{bmatrix} \tag{2.42}$$

$$u_r^2 = w^2 \cdot I^1 = \begin{bmatrix} -0.7 \cdot (-1) + 0.6 \cdot 0.567 + 0.2 \cdot 0.591 + 0.7 \cdot 0.512 \\ -0.3 \cdot (-1) + 0.7 \cdot 0.567 + 0.2 \cdot 0.591 + 0.8 \cdot 0.512 \end{bmatrix} = \begin{bmatrix} 1.5168 \\ 1.2247 \end{bmatrix} \tag{2.43}$$

$$I^2 = g(u_r^2) = \begin{bmatrix} g(1.5168) \\ g(1.2247) \end{bmatrix} = \begin{bmatrix} 0.82 \\ 0.772 \end{bmatrix} \tag{2.44}$$

Por último é processada a última camada com o vetor resultante I^2 para obter o s valor de um exemplo para essa rede:

$$I^2 = \begin{bmatrix} -1 \\ 0.82 \\ 0.772 \end{bmatrix} \tag{2.45}$$

$$u_r^3 = w^3 \cdot I^2 = \begin{bmatrix} 0.1 \cdot (-1) + 0.8 \cdot 0.82 + 0.5 \cdot 0.772 \end{bmatrix} \qquad = \begin{bmatrix} 0.942 \end{bmatrix} \tag{2.46}$$

$$s^{(i)} = g(u_r^3) = \begin{bmatrix} g(0.942) \end{bmatrix} = \begin{bmatrix} 0.719 \end{bmatrix} \tag{2.47}$$

Para uma fase de operação o valor de $s^{(i)}$ seria a classe predita pela RNA PMC. Em uma fase de treinamento a diferença entre $s^{(i)}$ processado pela Rede Neural Artificial e o valor esperado $y^{(i)}$. Com o valor da diferença é realizada uma calibragem na Rede Neural, usando o algoritmo de *backpropagation* ou retropopagação do erro e atualiza os peso para cada camada da Rede.

Usando o gradiente descendente estocástico para a calibragem de pesos da Rede Neural ou algum outro algoritmo que faça essa calibragem, uma equação da diferença do valor da nossa hipótese pelo valor real é usada:

$$E = \frac{1}{2}\sum_{r=1}^{m_3}(y_r^{(i)} - s_r^{(i)})^2 \tag{2.48}$$

onde m_3 é quantidade de neurônios da camada de saída da rede.

Derivando a expressão (2.48) em relação aos pesos da última camada w^3:

$$\frac{\partial E}{\partial w_{ri}^3} = \frac{\partial E}{\partial s_r^{(i)}} \cdot \frac{\partial s_r^{(i)}}{\partial u_r^3} \cdot \frac{\partial u_r^3}{\partial w_{ri}^3} \tag{2.49}$$

Fazendo as parciais, os seguintes termos são obtidos para o ajuste de pesos:

$$\frac{\partial E}{\partial s_r^{(i)}} = -(y_r^{(i)} - s_r^{(i)}) \tag{2.50}$$

$$\frac{\partial s_r^{(i)}}{\partial u_r^3} = g'(u_r^3) \tag{2.51}$$

$$\frac{\partial u_r^3}{\partial w_{ri}^3} = I_r^2 \tag{2.52}$$

$$\frac{\partial E}{\partial w_{ri}^3} = -(y_r^{(i)} - s_r^{(i)}) \cdot g'(u_r^3) \cdot I_r^2 \qquad \text{Obtido de (2.50), (2.51) e (2.52)} \tag{2.53}$$

$$\frac{\partial E}{\partial u_r^3} = -(y_r^{(i)} - s_r^{(i)}) \cdot g'(u_r^3) \tag{2.54}$$

Derivada a função aplica-se o gradiente descendente estocástico para os pesos camada de saída:

$$w_{ri}^3(novo) \leftarrow w_{ri}^3(antigo) - \eta \cdot (y_r^{(i)} - s_r^{(i)}) \cdot g'(u_r^3) \cdot I_r^2 \tag{2.55}$$

onde η é a taxa de aprendizagem ($\eta > 0$).

Logo em seguida, aplica-se as camadas anteriores com suas devidas correções levando em conta os pesos da camada de saída. A seguinte equação representa essa tarefa e é desenvolvida:

$$\frac{\partial E}{\partial w_{ri}^2} = \frac{\partial E}{\partial I_r^2} \cdot \frac{\partial I_r^2}{\partial u_r^2} \cdot \frac{\partial u_r^2}{\partial w_{ri}^2} \tag{2.56}$$

$$\frac{\partial E}{\partial I_r^2} = \sum_{k=1}^{m_3} \frac{\partial E}{\partial u_r^3} \cdot \frac{\partial u_r^3}{\partial I_r^2} = \sum_{k=1}^{m_3} \frac{\partial E}{\partial u_j^3} \cdot w_{kr}^{(3)} \tag{2.57}$$

$$\frac{\partial I_r^2}{\partial u_r^2} = g'(u_r^2) \tag{2.58}$$

$$\frac{\partial u_r^2}{\partial w_{ri}^2} = I_r^1 \tag{2.59}$$

$$\frac{\partial E}{\partial w_{ri}^2} = \sum_{k=1}^{m_3} \frac{\partial E}{\partial u_r^3} \cdot w_{kr}^{(3)} \cdot g'(u_r^2) \cdot I_r^1 \tag{2.60}$$

A segunda camada escondida é atualizada deste modo:

$$w_{ri}^2(novo) \leftarrow w_{ri}^2(antigo) - \eta \cdot \frac{\partial E}{\partial w_{ri}^2} \tag{2.61}$$

Esse método de atualização será propagado até os pesos da primeira camada escondida, daí o nome de retropropagação do erro. Note que sempre se leva em consideração o valor da camada posterior, como na fórmula (2.57) no resultado final para atualização de pesos em (2.60) e (2.61).

A última camada a ser atualizada é $W^{(1)}$:

$$\frac{\partial E}{\partial w_{ri}^1} = \frac{\partial E}{\partial I_r^1} \cdot \frac{\partial I_r^1}{\partial u_r^1} \cdot \frac{\partial u_r^1}{\partial w_{ri}^1} \tag{2.62}$$

$$\frac{\partial E}{\partial I_r^1} = \sum_{k=1}^{m_2} \frac{\partial E}{\partial u_r^2} \cdot \frac{\partial u_r^2}{\partial I_r^1} = \sum_{k=1}^{m_2} \frac{\partial E}{\partial u_r^1} \cdot w_{kr}^{(2)} \tag{2.63}$$

$$\frac{\partial I_r^1}{\partial u_r^1} = g'(u_r^1) \tag{2.64}$$

$$\frac{\partial u_r^1}{\partial w_{ri}^1} = x^{(i)} \tag{2.65}$$

$$\frac{\partial E}{\partial w_{ri}^1} = \sum_{k=1}^{m_2} \frac{\partial E}{\partial u_r^1} \cdot w_{kr}^{(2)} \cdot g'(u_r^1) \cdot x^{(i)} \tag{2.66}$$

A atualização dos pesos da primeira camada escondida:

$$w_{ri}^1(novo) \leftarrow w_{ri}^1(antigo) - \eta \cdot \frac{\partial E}{\partial w_{ri}^2} \tag{2.67}$$

Aprendizado Profundo

O Aprendizado Profundo ou *deep learning* é uma nova área no campo de Aprendizado de Máquina. Ela atualmente possui muitas definições em alto nível. Uma definição que se adapta a essa dissertação é:

> Um subconjunto dentro do Aprendizado de Máquina que é baseado em algoritmos para aprendizado de múltiplos níveis de representação para modelar relacionamentos complexos entre dados. Um alto nível de *features* e conceitos são assim definidos em termos de baixo níveis, e tais hierarquias de *features* é chamada de arquiteturas profundas (*deep*). Muitos desses modelos são baseados em aprendizado não supervisionado de representações (Deng et al., 2014).

A utilização de *deep learning* tem se mostrado popular e promissora em resultados que alcançam o estado da arte em diversas tarefas, sua escalabilidade com a quantidade de dados e a produção de diversas aplicações. Tudo isso foi possível pelo crescimento de disponibilidade de dados e pelo poder de processamento computacional. Os algoritmos de *deep learning* normalmente fazem uso intensivo das GPU's para realizar os cálculos e reduzir o tempo de computação.

Essas técnicas e algoritmos abrangem aplicações em diversas áreas como processamento de linguagem natural, reconhecimento de padrões em imagens e áudios. Os métodos *deep learning* são o aprendizado de representação com múltiplas camadas de representações, começando pelo nível de abstração dos dados em sua forma inicial até obter representações mais abstratas nas camadas mais profundas da arquitetura das redes neurais. Deste modo sua

capacidade de aprender e transformar representações, que por vezes são funções complexas se tornou possível de serem aprendidas (LeCun et al., 2015b).

Outra série de variações de redes neurais surgiram para obter uma representação melhor dos dados para um desempenho mais alto nas tarefas de classificação e reconhecimento. As Redes Neurais Convolucionais no reconhecimento de objetos em imagens (Simonyan e Zisserman, 2014), as Redes Neurais Recorrentes na categorização de textos (Lai et al., 2015) e no processamento de linguagem falada combinada com o método de *Long Short-term Memory* (LSTM) (Graves et al., 2013) alcançando o estado da arte.

2.3 Transformação e Representação de Dados

Grande parte das informações disponíveis na Internet está em formato textual. Entretanto, os algoritmos de Aprendizado de Máquina, em sua maioria precisam de uma representação estruturada de dados como uma tabela atributo valor. A conversão dos dados textuais em formatos estruturados tem sido, por muitos anos, um grande desafio para a área de Aprendizado de Máquina. Nestas seções serão apresentadas as técnicas mais populares para a representação de textos.

2.3.1 *Bag-of-Words*

Bag-of-Words (BOW) é um modo bem conhecido de representar palavras em documentos ou frases. O método consiste em criar um conjunto com todas as palavras dos documentos, chamado de vocabulário, e quantificar a ocorrência das palavras em uma configuração vetorial, representando cada frase por um vetor com o tamanho do vocabulário. Um exemplo dessa representação e sendo o *dataset* composto das seguintes frases:

$$F_1 = \text{“O Pedro comprou o jornal na feira.”} \tag{2.68}$$
$$F_2 = \text{“A Joana voltou da feira com o jornal.”} \tag{2.69}$$
$$F_3 = \text{“O Henrique comprou um abacate.”} \tag{2.70}$$
$$F_4 = \text{“O Artur saiu da casa.”} \tag{2.71}$$
$$F_5 = \text{“A Estela estava com a Joana.”} \tag{2.72}$$

O BOW dispões as frases do *dataset* como é exibido na Tabela 2.3. As suas representações vetoriais para essas duas frases como documentos, onde o vetor V representa o vocabulário desse *dataset*, são:

V = {o, pedro, comprou, jornal, na, feira, a, joana, voltou, da, com, henrique, um, abacate, artur, saiu, casa, estela, estava}

A representação BOW é uma das abordagens mais populares de representação de textos (Wallach, 2006). Apesar de muito simples, consegue obter resultados expressivos para a maioria dos problemas de classificação. Entretanto a literatura reporta diversas limitações da representação:

1. **Ignora a ordem das palavras**: a ordem das palavras são ignoradas na representação. Assim um texto com a sequencia “Cachorro quente” e “quente cachorro” são representadas da mesma maneira no vetor produzido por BOW.

		Vocabulário																			
		o	pedro	comprou	jornal	na	feira	a	joana	voltou	da	com	henrique	um	abacate	artur	saiu	casa	estela	estava	Total
Dataset	F_1	2	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	7
	F_2	1	0	0	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	8
	F_3	1	0	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	5
	F_4	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	5
	F_5	0	0	0	0	0	0	2	1	0	0	1	0	0	0	0	0	0	1	1	6
	Total	5	1	2	2	1	2	3	2	1	2	2	1	1	1	1	1	1	1	1	31

Tabela 2.3: Tabela de distribuição de termo frequência (TF).

2. **Ignora a semântica das palavras**: palavras como “poderoso”, “forte” e “Paris”, na representação BOW podem ser equidistantes, apesar de “poderoso” e “forte” serem mais similares que “poderoso” e “Paris” (Le e Mikolov, 2014).

3. **Alta dimensão**: o número de atributos é definido pelo tamanho do vocabulário. Assim um corpus com 10 textos e cada texto com 100 palavras pode produzir um vocabulário de tamanho 1000. Em problemas de mineração de textos é comum o número de atributos passar de 10 mil atributos (Yang e Pedersen, 1997; de Souza Rodrigues, 2016; Forman, 2003). Conjunto de dados *benchmarks* como a REUTERS, 20 NEWS GROUP, OHSUMED, possuem respectivamente 16.039, 12.7994 e 72.076 atributos. Segundo o trabalho de Forman (2003); Yang e Pedersen (1997), a quantidade de atributos relevantes é pequena.

4. **Produz vetores esparsos**: na representação BOW é comum encontrar conjunto de dados onde apenas um ou dois porcento dos valores são diferente de zero. Isso ocorre devido ao tamanho do vetor ser definido pelo tamanho do conjunto que representa a união de todas as palavras do corpus. Assim, mesmo que o texto tenha apenas uma sentença, este texto será representado por este vetor muito maior.

Boa parte dos trabalhos que buscam melhorar BOW atuam na melhoria de uma ou mais limitações mencionadas. Por exemplo, Sriram et al. (2010) atua na redução de atributos atuando na terceira limitação sobre alta dimensão. No trabalho os autores classificam textos no twitter utilizando um conjunto com 15 atributos (palavras) do texto para a sua classificação.

Campos e Matsubara (2014) também fizeram seleção de atributos e representam textos com BOW. O trabalho reduz uma representação de 16501 em apenas 72 palavras como atributos dos documentos no domínio de finanças brasileira. A representação com 72 palavras é melhor nos dois algoritmos testados, SVM e NB. Um resultado interessante deste trabalho é que NB com os 72 atributos selecionados apresentou melhor desempenho que SVM quando comparado com *accuracy*, *precision* e *recall*. Por meio das medidas de AUC ROC os resultados foram similares.

2.3.2 Word2vec

Uma abordagem que tem se tornado cada vez mais popular é a WORD2VEC (Mikolov et al., 2013a). WORD2VEC atua na melhoria das três últimas limitações do BOW, esparsidade, dimensionalidade e semântica. WORD2VEC foi originalmente proposto por Mikolov et al. (2013c). A abordagem utiliza dois tipos de arquiteturas de rede neural, que calculam os vetores de palavras ou também conhecido por *word embeddings*. As arquiteturas são o CONTINUOUS BAG-OF-WORDS (CBOW) e o SKIP-GRAM.

Os *word embeddings* produzidos por esses métodos carregam informações sobre similaridade de contexto e características consequentes dessa propriedade adquirida, como algumas regras sintáticas ou morfológicas. Essas características são exploradas por muitos pesquisadores para analisar textos de diversas maneiras.

No treinamento do WORD2VEC, o objetivo é maximizar o $\log$ da probabilidade média das palavras passadas e futuras:

$$\frac{1}{|V|} \sum_{t=i}^{|V|-i} \log Pr(\rho_t | \rho_{t-i}, ..., \rho_{t+i}) \tag{2.73}$$

sendo $|V|$ é a quantidade de palavras no vocabulário, i é o tamanho da janela, (i palavras passadas e i futuras) e ρ é uma palavra.

Topologia da rede CBOW *e* SKIP-GRAM

A topologia *Continuous Bag-Of-Words* é uma rede neural que tenta prever uma palavra em alguma posição de um contexto dado como entrada para a rede. Esse contexto é formado por i palavras a frente e i passadas. Para isso a rede faz estimativas usando um classificador log-linear. Ela é similar a uma rede *feedforward* NNLM (Modelo de Rede Neural *Feedforward* Probabilística), com a sua camada escondida removida e a camada de projeção é compartilhada com todas as palavras, assim todas as palavras obtêm a projeção na mesma posição (seus vetores são médias) (Mikolov et al. (2013a)).

Na Figura 2.7 é exibida uma representação da estrutura das duas topologias, com o tamanho da janela $i = 2$. Na arquitetura CBOW $w(t)$ é a palavra que está sendo predita, sua rede recebe as palavras que estão no contexto, nos arredores de $w(t)$, e são descritas como $w(t-1), w(t-2), w(t+1), w(t+2)$. Ainda na Figura 2.7, a arquitetura SKIP-GRAM, a palavra $w(t)$ é dada como entrada da rede e tenta prever seu contexto que são as palavras $w(t-1), w(t-2), w(t+1), w(t+2)$.

A complexidade do seu treinamento se dá pela seguinte expressão:

$$Q_{\text{CBOW}} = m \times d_{emb} + d_{emb} \times log_2(|V|),$$

sendo m é o tamanho da entrada da rede neural, d_{emb} é a quantidade de dimensões e $|V|$ é o tamanho do vocabulário.

O modelo SKIP-GRAM é análogo ao CBOW, mas a sua entrada é uma palavra e tenta predizer as palavras ao seu redor, o seu contexto, palavras passadas e futuras, também usando um classificador log-linear. A complexidade desse modelo é:

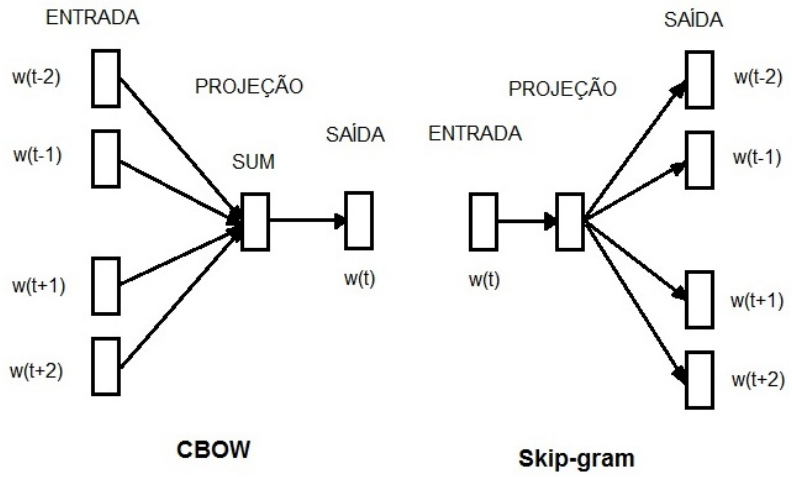


Figura 2.7: Diagrama das topologias de redes neurais.

$$Q_{\text{SKIP-GRAM}} = ctx \times (d_{emb} + d_{emb} \times log_2(|V|)),$$

sendo ctx é a distância máxima das palavras. Logo, se a escolha for de ctx = 5, para o treinamento de cada palavras, então será selecionado aleatoriamente um valor para R entre 1 e ctx. Depois R palavras do passado e R do futuro da palavra corrente serão usadas como rótulos corretos.

Com o treinamento dessas duas topologias é possível obter relações semânticas e sintáticas sobre seus *word embeddings*, dependendo da qualidade de base de dados. Quanto maior a base de dados, melhor será suas relações semânticas e sintáticas nos *word embeddings*. Um conjunto de *word embeddings*[1] treinado com o WORD2VEC sobre o corpus do *Google News* contém boas relações semânticas e sintáticas que são usadas por muitos pesquisadores e é utilizados neste trabalho.

Utilizando os *word embeddings* do *Google News*, algumas operações vetoriais sobre as representações das palavras demonstram relações semânticas entre capitais e países, gêneros. Por exemplo, sejam os *word embeddings* $emb_{Brasil}, emb_{Brasília}, emb_{Russia}, emb_{Moscou} \in \mathbb{R}^d$ que representam as palavras Brasil, Brasília, Rússia e Moscou, respectivamente, numa operação vetorial com os valores produz um resultado interessante como:

$$emb_{Russia} - emb_{Brasil} + emb_{Brasília} \approx emb_{Moscou} \tag{2.74}$$

ou seja, as representações guardam informações do sobre o capital e país, resultantes de uma abordagem sem supervisão. Na Figura 2.8 está exibida a relação vetorial entre as palavras e as distâncias entre elas separam o gênero entre os substantivos, como *queen* para rainha e *king* para rei.

Em outras representações produzidas com o WORD2VEC, também é possível realizar traduções de palavras apenas realizando operações vetoriais (Mikolov et al., 2013b).

Uma implementação do algoritmo do WORD2VEC foi disponibilizado em Python na biblioteca GENSIM com os dois modelos SKIP-GRAM e CBOW, usando um SOFTMAX hierárquico ou uma amostragem negativa (*Negative Sampling*) que são descritos em Mikolov et al. (2013c). Seus parâmetros são detalhados

[1]Disponível em https://code.google.com/archive/p/word2vec/

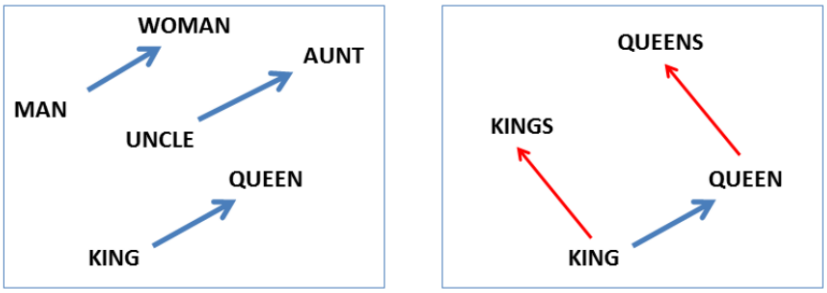


Figura 2.8: Relação entre as representações vetoriais com *word embedding* produzidos com WORD2VEC (Mikolov et al., 2013d).

no trabalho de Goldberg e Levy (2014) como métodos de otimização para obter aproximações para a rede calcular e construir seus vetores. Também está disponível em uma biblioteca JAVA, *open source*, DL4J (*Deep Learning for Java*), é escalável com `Hadoop` e `Spark`, onde possuem integração com a GPU.

Há também uma variação do WORD2VEC criada por um dos autores do WORD2VEC chamada de SENTENCE2VEC. Nessa ferramenta o foco é classificar as sentenças, parágrafos ou documentos. Ela usa os mesmos modelos de treinamentos do WORD2VEC, com o CBOW e SKIP-GRAM para vetores de palavras e sentenças, a diferença está na criação de representações vetoriais das sentenças e o modo de predição e treinamento consiste em prever as n sentenças a frente. Essa abordagem não melhora a representação das palavras em vetores, porém obtém ótimos resultados para outras tarefas semânticas e sintáticas que envolvem sentenças, parágrafos e documentos (Le e Mikolov (2014)).

Em Goldberg e Levy (2014) é explicada com mais detalhes os modelos SKIP-GRAM e o *negative sampling* descritos em Mikolov et al. (2013a) e Mikolov et al. (2013c). Usando da sua intuição, Goldberg e Levy (2014) afirmam que significado das palavras é compartilhado em vários contextos será similar com cada outra palavra de contexto similar.

Até um tempo atrás, os métodos tradicionais não conseguiam treinar grandes volumes de dados, documentos contendo bilhões de palavras e um vocabulário com cerca de milhões de palavras, que obtivesse resultados bem sucedidos e conseguisse gerar representações de palavras em vetores com propriedades e um múltiplo grau de similaridade (Mikolov et al., 2013d,a). O treinamento do WORD2VEC realiza o processamento de um grande volume de texto rapidamente, aproximadamente 1,6 bilhões de palavras, em menos de um dia (Mikolov et al., 2013a).

Métodos tradicionais tratavam com redução de dimensionalidade este problema usando técnicas como a *Principal Analysis Component* (PCA) e a *Singular Value Decomposition* (SVD).

O teorema SVD calcula as relações mais fortes entre os termos. Na SVD a matriz é decomposta no produto de três outras matrizes. Uma matriz descreve as linhas originais e outra matriz as colunas originais como vetores derivados dos fatores de valores ortogonais, e finalmente a última é uma matriz diagonal com as escalas de forma que a multiplicação das três matrizes formam a original (Landauer et al. (1998)).

A PCA é uma técnica já bem conhecida para processamento, compressão e visualização de dados apesar da sua eficiência limitada pela sua transfor-

mação global (Tipping e Bishop, 1999). A PCA tem como objetivo construir novas variáveis por meio da combinação linear das variáveis originais dos dados (Holland, 2008).

2.4 Métricas para Avaliação dos Classificadores

Para avaliar e comparar o desempenho dos classificadores serão usadas as métricas *precision*, *recall*, *accuracy*, *f1-score*, já bem conhecidas em Aprendizado de Máquina. As métricas utilizadas podem ser obtidas a partir da matriz de confusão.

A matriz de confusão possui colunas que indicam as classes reais dos dados e nas linhas as predições dos classificadores, e está exibido na Tabela 2.4.

		Classe Predita		
		Y	N	Total
Classe Verdadeira	Positivo = P	Verdadeiro Positivo = VP	Falso Positivo = FP	P
	Negativo = N	Falso Negativo = FN	Verdadeiro Negativo = VN	N
	Total	$VP + FN$	$FP + VN$	$P + N$

Tabela 2.4: Matriz de confusão

Sejam Cl_A e Cl_B dois classificadores de um problema que identifica autenticidade de assinaturas, sendo positiva quando é autentica e negativa para uma assinatura falsa. Na Tabela 2.5 e 2.6 há duas distribuições dos resultados de classificação e serão usados para demonstrar o como as métricas avaliam o desempenho da classificação de Cl_A e Cl_B.

		Classe Predita		
		Y	N	Total
Classe Verdadeira	Positivo	190	10	200
	Negativo	600	200	800
	Total	790	210	1000

Tabela 2.5: Tabela da matriz de confusão dos resultados para um classificador binário Cl_A sobre um T_{teste} desbalanceado $dataset_d$.

		Classe Predita		
		Y	N	Total
Classe Verdadeira	Positivo	450	50	500
	Negativo	120	380	500
	Total	570	430	1000

Tabela 2.6: Tabela da matriz de confusão para um classificador binário Cl_B sobre um T_{teste} balanceado $dataset_b$.

2.4.1 Precision

O *precision* mede a proporção de acertos em meio a quantidade de positivos (Powers (2011), Fawcett (2006)). Os valores de *precision* para os classificadores

Cl_A e Cl_B são:

$$\textit{precision} = \frac{VP}{VP + FP} \tag{2.75}$$

$$\textit{precision} \text{ do } \textit{dataset}_d = \frac{190}{200} = 0.95 \tag{2.76}$$

$$\textit{precision} \text{ do } \textit{dataset}_b = \frac{450}{500} = 0.9 \tag{2.77}$$

2.4.2 Accuracy

A *accuracy* está relacionada a quantidade de acerto pelo tamanho do conjunto (Powers (2011), Fawcett (2006)). Dos dados das Tabelas 2.5 e 2.6, obtém-se:

$$\textit{accuracy} = \frac{VP + VN}{P + N} \tag{2.78}$$

$$\textit{accuracy} \text{ do } \textit{dataset}_d = \frac{390}{1000} = 0.39 \tag{2.79}$$

$$\textit{accuracy} \text{ do } \textit{dataset}_b = \frac{830}{1000} = 0.83 \tag{2.80}$$

2.4.3 Recall

O *recall* também é conhecido por sensitividade e representa a quantidade de acertos positivos pela quantidade total de realmente positivos (Powers (2011), Fawcett (2006)). Para os classificadores Cl_A e Cl_B, os resultados de *recall* são:

$$\textit{recall} = \frac{VP}{VP + FN} \tag{2.81}$$

$$\textit{recall} \text{ do } \textit{dataset}_d = \frac{190}{790} \approx 0.24 \tag{2.82}$$

$$\textit{recall} \text{ do } \textit{dataset}_b = \frac{450}{570} \approx 0.79 \tag{2.83}$$

2.4.4 F1-Score

Esta medida é conhecida por F-score. Representa uma média ponderada de *precision* e *recall* (Powers (2011)). Seu valor é obtido da seguinte fórmula:

$$\textit{f1-score} = \frac{2 \cdot \textit{precision} \cdot \textit{recall}}{\textit{precision} + \textit{recall}} \tag{2.84}$$

$$\textit{f1-score} \text{ do } \textit{dataset}_d = \frac{2 \cdot 0.95 \cdot 0.24}{0.95 + 0.24} \approx 0.38 \tag{2.85}$$

$$\textit{f1-score} \text{ do } \textit{dataset}_b = \frac{2 \cdot 0.9 \cdot 0.79}{0.9 + 0.79} \approx 0.84 \tag{2.86}$$

Por meio dessas avaliações, nota-se que métricas como *recall*, *precision*, podem obter altos valores como resultados, porém não significa que correspondam com a qualidade do classificador. Para uma avaliações mais profunda é necessário conhecer a distribuição das classes no conjunto de exemplos e os valores de *f1-score*.

No exemplo entre $\textit{dataset}_d$ e $\textit{dataset}_b$, a métrica de *accuracy* encontrou a diferença de qualidade entre sobre os classificadores Cl_A e Cl_B. Mas além

dessa métrica há outra cálculo para comparar classificadores, obtida através da análise da curva *ROC*.

Em um problema que envolve a classificação em mais de uma classe é chamado de multi-classe (Aly, 2005). Para problemas multi-classe as métricas de *precision* e *recall* podem ser calculadas para cada classe e obter uma média como resultado final. Ainda pode ser ponderada pelo número de exemplos verdadeiros para cada classe, tornando as métricas de *precision* e *recall* ponderadas.

CAPÍTULO

3

Seleção de Instâncias

A Seleção de Instâncias seleciona um subconjunto do conjunto de treinamento que possua uma representação dos dados sintetizada que tente preservar o desempenho do algoritmo de Aprendizado de Máquina (Reinartz, 2002; Olvera-López et al., 2010). Em consequência deste objetivo, o método diminui o conjunto de treinamento T_{treino} eliminando ruídos, poupando memória, diminuindo o tempo de execução e ainda pode manter ou em alguns casos até melhorar o desempenho na classificação do conjunto inicial.

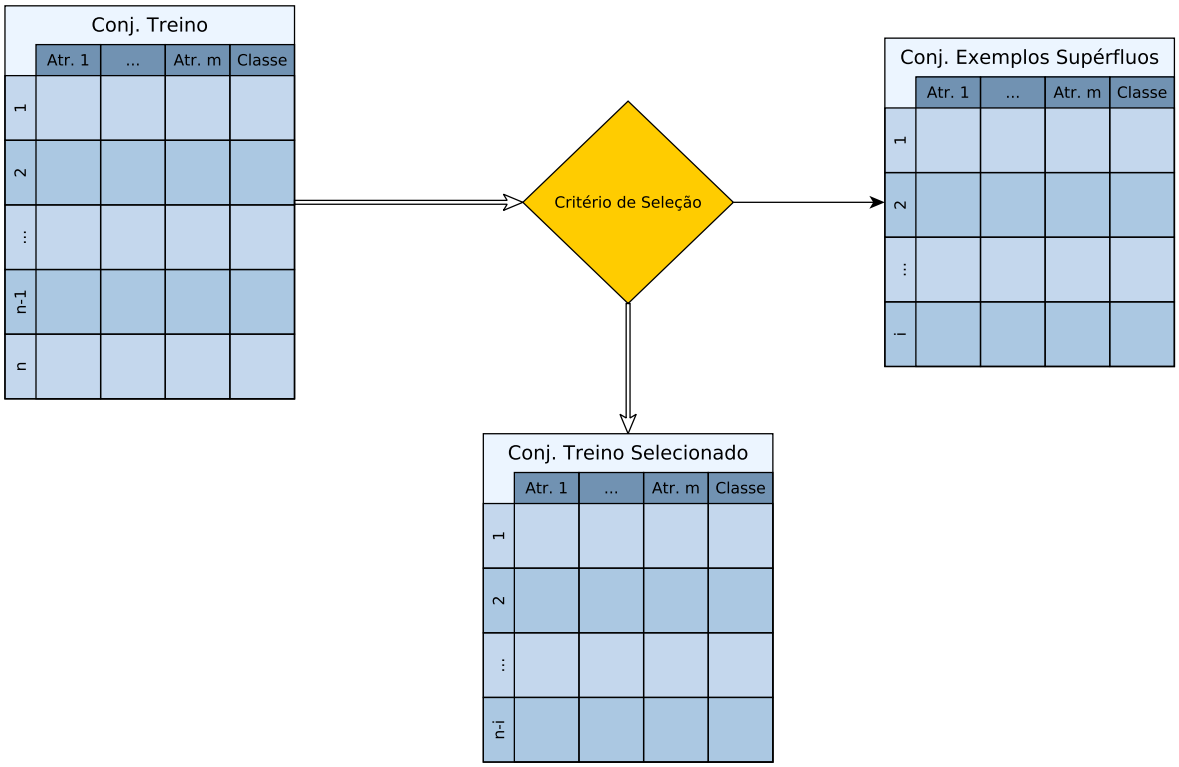


Figura 3.1: Representação gráfica do processo de Seleção de Instâncias.

Apesar de alguns algoritmos derivados do kNN usarem técnicas envolvendo

estruturas de dados como a construção da *k-d tree* para melhorar o tempo de computação no treinamento e teste, eles não conseguem reduzir o uso de memória pois todos os exemplos precisam ser alocados (Wilson e Martinez, 2000).

A Figura 3.1 exibe a ação de um algoritmo de Seleção de Instâncias sobre um conjunto de treinamento removendo exemplos supérfluos ou desnecessários. Em todo algoritmo há um tipo de critério para seleção que decidirá quais os melhores exemplos. Assim, i exemplos são descartados restando $n - i$, que são utilizados para o treinamento no algoritmo de Aprendizado de Máquina.

Há dois modos de se realizar Seleção de Instâncias, o primeiro avalia o conjunto selecionado e se apoia sobre a acurácia do classificador, e é dito de *wrapper*; o segundo é chamado de *filter* e se baseia em alguma regra que não está diretamente ligada ao classificador (Olvera-López et al., 2010). Sendo a maioria dos algoritmos do tipo *wrapper* propostos para a classificação KNN (Cover e Hart, 1967).

Em notação, o que a Seleção de Instâncias realiza é encontrar um subconjunto S que represente o seu superconjunto T_{treino} de modo que mantenha sua acurácia e isso pode ser feito nos seguintes moldes:

- Incremental: Adiciona-se a cada iteração do algoritmo um exemplo de T_{treino} para o conjunto selecionado S, desde de que atenda alguma métrica estabelecida.

- Decremental: Inicialmente o conjunto selecionado S possui todos os exemplos de T_{treino} e retira-se a cada iteração aqueles exemplos que não expressam significado para a tarefa de classificação.

- Lote: Um subconjunto que satisfaz um determinado critério de exclusão é removido por inteiro, tornando o algoritmo mais rápido do que remover exemplos individualmente.

- Misturado: O conjunto de selecionados S é inicializado aleatoriamente por técnica Incremental ou Decremental, podendo excluir ou incluir qualquer exemplo dependendo do critério.

- Fixo: É derivado do método Misturado, porém existe um grupo fixo de exemplos que não é removível.

Neste capítulo serão apresentados os algoritmos de Seleção de Instâncias utilizados neste projeto. Estes algoritmos já são bem conhecidos e servem como base para outros métodos de seleção mais próximos ao estado da arte.

3.1 *Condensed Nearest Neighbor*

O *Condensed Nearest Neighbor* (CNN) é um dos algoritmos mais primitivos de Seleção de Instâncias baseado em erros na classificação (Hart, 1968; Olvera-López et al., 2010). Sua proposta é incremental e o passo inicial é incluir randomicamente uma instância de cada classe do conjunto T_{treino}, que possui todos os exemplos, no subconjunto de selecionados S. Em seguida, a classificação utiliza como treino o conjunto S e o restante é usado como teste. Da predição destas instâncias de teste, os que diferem de sua classe verdadeira são adicionados à S. O processo termina quando todas as instâncias de

T_{treino} são verificadas e S é o subconjunto de T_{treino} com as melhores instâncias de acordo com o método do CNN.

No Algoritmo 5 está descrito os passos do método CNN. Como pode ser visto ele é um código de poucas linhas de implementação. Repare a ordem em que os elementos são incluídos na lista In não é importante. O método *índice* retorna um índice para o exemplo dado. O método *hipótese* computa o kNN para o exemplo dado de acordo com o conjunto de treino estabelecido anteriormente.

Algoritmo 5: Algoritmo CNN.

Entrada: O conjunto de treinamento T_{treino} e suas respectivas classes Y.
Resultado: O subconjunto $S \subseteq T_{treino}$; Uma lista In com os índices das instâncias de T_{treino}.

```
1 S = ∅;
2 S = seleciona aleatoriamente uma instância de cada classe de T_treino;
3 para todo x ∈ T_treino faça
4 |   hipótese = kNN(S);
5 |   se hipótese(x) ≠ Y[x] então
6 |   |   S = S ⋃ x;
7 |   |   T = T − x;
8 |   fim
9 fim
```

O CNN é um método do tipo *wrapper*. O *Selective Nearest Neighbour rule* (SNN) (Ritter et al., 1975) é uma variação do CNN onde toda instância selecionada está mais próxima do conjunto de S do que do original em T_{treino}, sendo as instâncias de T_{treino} as classificadas corretamente em um $1NN$. Outro método derivado é a generalização do CNN chamado de *Generalized Condensed Nearest Neighbor* (GCNN) (Chou et al., 2006). O GCNN insere instâncias em S caso respeite um critério de absorção de acordo com um limiar.

3.2 *Edited Nearest Neighbor*

O *Edited Nearest Neighbor* (ENN) é um método de Seleção de Instâncias baseado em erros na classificação (Wilson, 1972). Este método de seleção é decremental pois ele começa com todas instâncias de T_{treino} como selecionados e vai removendo de acordo com a classe majoritária da classificação do kNN. Utilizando o kNN com k = 3, se a classe majoritária predita pelo kNN for diferente da classe verdadeira, então essa instância é removida.

A remoção dos elementos deixa a fronteira de decisão entre as instâncias mais suave (Wilson e Martinez, 2000). No Algoritmo 6 está descrito os passos realizados pelo método de seleção do ENN.

3.3 *Random Mutation Hill Climbing*

O *Random Mutation Hill Climbing* (RMHC) é um algoritmo de Seleção de Instâncias baseado em busca local com um componente estocástico (Skalak, 1994) e é construído de modo fixo. O algoritmo usa os índices de uma cadeia de *string*'s binária para representar os exemplos que são selecionados pelo

Algoritmo 6: Algoritmo ENN.

Entrada: O conjunto de treinamento T_{treino} e suas respectivas classes Y.
Resultado: O subconjunto $S \subseteq T_{treino}$; Uma lista In com os índices das instâncias de T_{treino}.

```
1  S = T_treino;
2  I = índices(T_treino);
3  para todo x ∈ T_treino faça
4  |   S = S − x;
5  |   In = In − índice(x);
6  |   hipótese = KNN(S, k = 3);
7  |   se hipótese(x) = Y[x] então
8  |   |   S = S ⋃ x;
9  |   |   In = In ⋃ índice(x);
10 |   fim
11 fim
```

método. O objetivo é encontrar a *string*'s binária que obtenha um bom desempenho na classificação KNN com k=1, alterando a configuração da *string* randomicamente a cada iteração do algoritmo.

A base do método é descrito no Algoritmo 7. O str é uma cadeia de *string*'s binária com tamanho igual ao do conjunto de todos os dados. Fazem parte deste algoritmo as funções:

- *random*: cria aleatoriamente uma *string* binária de tamanho t_s e proporção p_r para quantidade de 1's;
- *evaluate*: avalia usando um KNN, para k=1, com as instâncias selecionadas para treino pelos índices da *string* str, onde suas posições possuem 1 e para instâncias de teste as que tem 0;
- *mutate*: cria uma nova *string* a partir da dada de entrada, mantendo a proporção.

Algoritmo 7: Algoritmo RMHC.

Entrada: O conjunto de treinamento T_{treino} e suas respectivas classes Y, avaliações e, porcentagem p_r
Resultado: O subconjunto $S \subseteq T_{treino}$.

$t_s = |T_{treino}|$;
$str = random(t, p_r)$;
$best = evaluate(str)$;
$iter = 0$;
enquanto $iter < e$ **faça**
    $sMutate = mutate(str)$;
    $evaluateMutate = evaluate(sMutate)$;
    se $evaluateMutate > best$ **então**
        $best = evaluateMutate$;
        $str = sMutate$;
    fim
    $iter = iter + 1$;
fim

3.4 *Silhouette*

Silhouette (SIL) é uma apresentação gráfica que possibilita a interpretação e validação em análises de agrupamento (Rousseeuw, 1987). Um agrupamento de dados é uma tarefa de formar grupos, na literatura também são chamados de clusters, e isto é feito de acordo com a similaridade das instâncias de dados (Jain e Dubes, 1988).

O valor de *Silhouette* utiliza uma métrica de distância dada, para cada instância x_i e seu valor é expresso na seguinte Equação 3.1:

$$sil(x^{(i)}) = \frac{b(x^{(i)}) - a(x^{(i)})}{\max\{a(x^{(i)}), b(x^{(i)})\}} \tag{3.1}$$

sendo $a(x^{(i)})$ a média da distância de $x^{(i)}$ com todos os outros exemplos do seu mesmo agrupamento e $b(x^{(i)})$ a média da distância entre $x^{(i)}$ e todos os outros exemplos do grupo mais próximo do agrupamento de $x^{(i)}$ que seja diferente de seu próprio agrupamento (Kaufman e Rousseeuw, 2009).

Na Figura 3.2 está uma representação de 10 exemplos em três agrupamentos A, B, C. O cálculo do valor de *Silhouette* para o exemplo $x^{(i)}$, computa todas as distância de $x^{(i)}$ com os outros exemplos do agrupamento A obtendo a média para o valor de $a(x^{(i)})$. O valor de $b(x^{(i)})$ é obtido calculando todas as distância entre os exemplos do agrupamento B, que é o mais próximo de A, com o exemplo $x^{(i)}$.

O intervalo produzido varia em $-1 \leq sil(x^{(i)}) \leq 1$. A análise do valor de *Silhouette* para cada exemplo diz que quanto mais próximo de 1, melhor agrupado está aquele exemplo. Quando perto de 0 esse exemplo está quase igualmente distante do agrupamento ao qual pertence e do mais próximo agrupamento. Se o valor é mais próximo de -1, significa que o exemplo $x^{(i)}$ foi agrupado erroneamente.

Uma modificação da *Silhouette* proposta pelo trabalho de Max (2016) é utilizada neste trabalho com o objetivo de explorar os resultados. Foram exploradas melhorarias nos índices de desempenho em classificação de textos

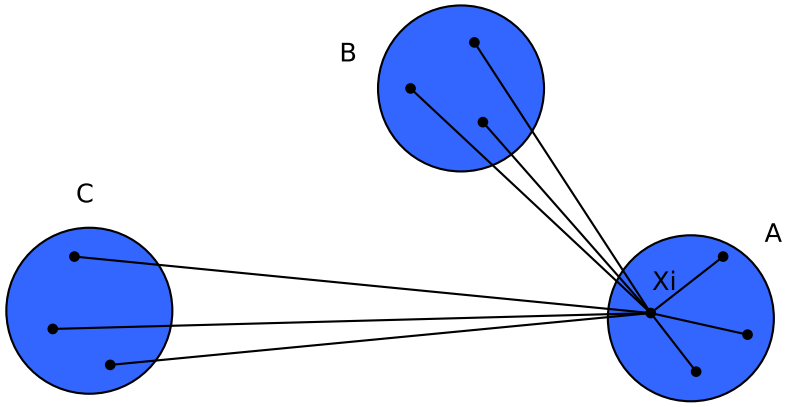


Figura 3.2: Representação gráfica de 3 agrupamentos para exemplificar o valor de *Silhouette*

com a predição do algoritmo do KNN. As representações textuais e métodos de seleção utilizadas auxiliaram no ganho de tempo, com e sem perdas consideráveis no desempenho de classificação. Já é conhecido que o algoritmo do KNN possui um tempo de treinamento rápido, ou não possui, dependendo da sua implementação. Porém, o tempo de computação para predizer geralmente é alta, dependendo da métrica utilizada. A necessidade de comparar todos os exemplos de treinamento com os de teste é uma das principais desvantagens do método. A proposta é um método *filter* e visa filtrar o conjunto de treinamento reduzindo o tempo durante a fase de teste do algoritmo KNN. O método usa uma regra para selecionar os exemplos e esta não está ligada diretamente ao classificador.

A técnica toma os valores do coeficiente de *Silhouette* para mensurar todas as instâncias e com isso decidir quais devem ser escolhidas. Aquelas mais próximas a borda de decisão são selecionadas. As instâncias são ordenadas crescentemente de acordo com o valor do coeficiente. Para o método quanto mais próximas a 0, ou seja, mais próximas a fronteira de decisão, melhor é a sua posição para separar os agrupamentos.

Usando o coeficiente da medida de *Silhouette* para avaliar quais exemplos são qualificados sobre uma das métricas de distância. O método recebe uma matriz com a distância entre todos os exemplos de treinamento. Adaptando ao coeficiente de *Silhouette* para a proposta, a seleção realiza um novo cálculo para a Equação 3.1. A média em a e b, fixa em selecionar o vizinho mais próximo da classe de $x^{(i)}$ e o vizinho mais próximo das classes oposta à $x^{(i)}$. O método SILHOUETTE SELECTION utiliza dos valores $sil(x^{(i)})$ de cada instância $x^{(i)}$ do conjunto de treinamento. Quando o valor $sil(x^{(i)})$ está próxima de 0, implica dizer que a instância $x^{(i)}$ está próxima da borda de decisão, sendo considerada uma determinada medida de distância para o método.

Na Figura 3.3 está exibido como é computado os valores de *Silhouette* para todos os exemplos. As setas tracejadas descrevem a distância de um exemplo com o seu vizinho mais próximo de outra classe. As setas contínuas sinalizam a distância com o seu vizinho mais próximo dentro da própria classe. Assim, para os valores anotados para a instância mais acima da classe estrela de 6

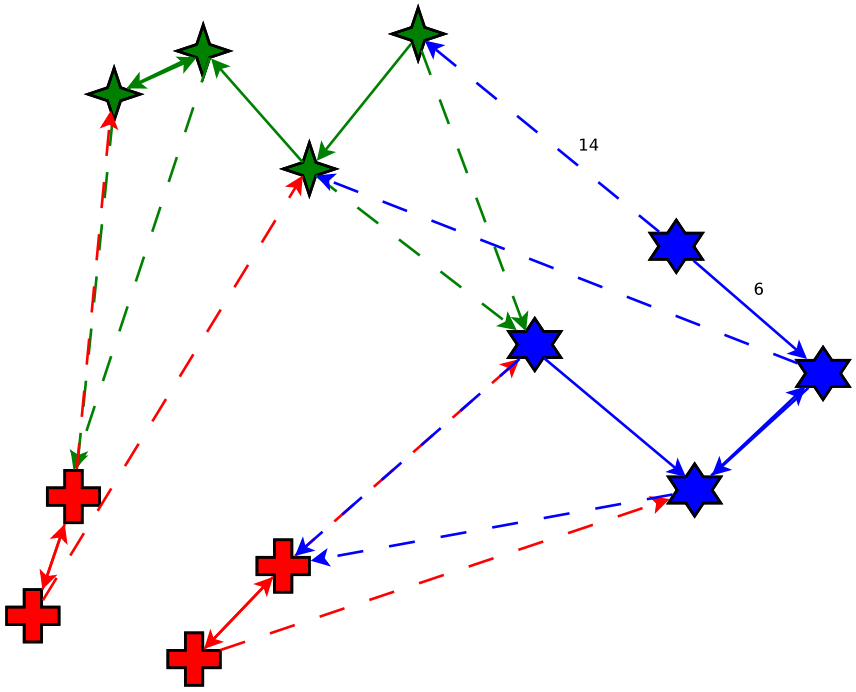


Figura 3.3: Representação gráfica do processo de construção da distância usando a *Silhouette* para seleção de instância do SILHOUETTE SELECTION.

pontas na Figura 3.3, sua computação de *Silhouette* é:

$$sil(x^{(i)}) = \frac{14 - 6}{\max\{6, 14\}} = \frac{8}{14} \approx 0,5714 \tag{3.2}$$

O Algoritmo 8 exibe os passos para seleção dos exemplos. Repare que os valores de *Silhouette* são calculadas e retornadas em uma função de módulo na linha 5. Exemplos que extrapolam um pouco a fronteira de decisão também são considerados borda e são escolhidos como relevantes para o método. Nas linhas 11 e 12, S e U são conjuntos de arranjos e a ordem em que os elementos são colocados importam para a relevância do exemplo. A quantidade de exemplos selecionado pode ser em um valor inteiro como exibido no Algoritmo 8 ou em porcentagem (necessária uma modificação no algoritmo). A quantidade ou percentagem são configurados pelo usuário.

Ainda observando o Algoritmo 8, na linha 4, onde é calculada a distância entre as instâncias de treinamento pode ser substituída por outros métricas de distâncias. Por exemplo pelas distâncias de métrica já estabelecida pelo algoritmo do KNN ou resultados de transformação por uma matriz *Mahalanobis* de um método de Aprendizado de Métrica como o NCA ou o S-WCD.

Em trabalhos onde transformações nos espaços dos exemplos são transformados por métricas, que aproximam instâncias de classes semelhantes e separam diferentes, o método SILHOUETTE SELECTION também ganha em acurácia e fornece vantagem em tempo de predição.

O SILHOUETTE SELECTION é um método rápido de se computar, pois se enquadra nos modelos de seleção do modo *filter* de critérios e nos moldes de Lote para selecionar instâncias. Além disso, com o coeficiente de *Silhouette* permite comparar as melhores instâncias, realizando uma ordenação para que seja selecionada na proporção desejada pelo usuário.

O custo computacional desse algoritmo depende do método de ordenação, foi utilizado o *quicksort* com tempo médio de $O(n \log n)$, as distâncias Euclide-

Algoritmo 8: Pseudo algoritmo de seleção de instâncias SILHOUETTE SELECTION.

Entrada: o conjunto de treinamento T_{treino} com suas respectivas classes rotuladas Y; Parâmetro p_r para quantidade e forma a ser selecionado.

Resultado: Um subconjunto S de T_{treino} com os exemplos selecionados, uma lista In com respectivos índices no conjunto de T_{treino} selecionados em S, ambos com tamanho ou proporcional a p.

```
1  S = ∅;
2  In = ∅;
3  n = 0;
4  D_{T_treino} = distancia_Euclideana(T_treino, T_treino);
5  Sil_{T_treino} = |silhouette(D_{T_treino})|;
6  O_{T_treino} = ordenar_crescente(Sil_{T_treino});
7  para ∀x^(i) ∈ O_{T_treino} faça
8  |   y^(i) = Y[i];
9  |   se n ≤ p_r então
10 |   |   n = n + 1;
11 |   |   S = S ∪ O_{T_treino}[i];
12 |   |   In = In ∪ i;
13 |   fim
14 fim
```

anas em tempo $O(n)$, o cálculo da *Silhouette* é feita em $O(n^2 \log n)$, pois percorre todos os exemplos procurando qual o exemplo mais próximo da mesma e oposta da sua classe.

O valor de *Silhouette* também já foi utilizado para Seleção de Instâncias em tarefas de classificação de texto (Dey et al., 2011).

CAPÍTULO

4

Algoritmos de Otimização com Aprendizado de Métrica

Este capítulo discute sobre os algoritmos de classificação de textos do *Word Mover's Distance* e *Supervised Word Mover's Distance*. O WMD foi inspirado no *Earth Mover's Distance* (EMD) e está descrito da Seção 4.1. Logo em seguida é explicado a relação do EMD com o WMD na Seção 4.2 juntamente com mais detalhes do algoritmo do WMD. O *Supervised Word Mover's Distance* é apresentado uma versão supervisionado na Seção 4.3.

4.1 *Earth Mover's Distance*

O *Earth Mover's Distance* (EMD) foi apresentado por Rubner et al. (1998) como métrica para medir similaridades entre imagens usando cor e textura. O seu nome faz referência a analogia intuitiva por trás da métrica na qual, dadas duas distribuições de probabilidade, uma é vista como massas de terra espalhadas no espaço e outra como uma coleção de buracos no mesmo espaço. Deste modo, o EMD mede a menor distância em trabalho necessário para mover a terra e preencher todos os buracos (Rubner et al., 1998). A unidade de trabalho corresponde ao transporte de uma unidade de terra por uma unidade de distância. Neste problema, assume-se que há terra suficiente para preencher todos os buracos. Se este não é o caso, então inverte-se qual distribuição é terra e qual é buraco.

O cálculo do EMD pode ser reduzido ao conhecido problema de fluxo de custo mínimo (PFCM) em grafo bipartido. Este problema pode ser formalizado como o seguinte problema de programação linear: seja $\mathcal{I}$ um conjunto de fornecedores, $\mathcal{J}$ um conjunto de consumidores, e c_{ij} o custo de enviar uma unidade de suprimento de $i \in \mathcal{I}$ para $j \in \mathcal{J}$. Na Figura 4.1 está exibido um exemplo do problemas com três fornecedores e dois consumidores. Nós queremos encontrar o conjunto de fluxos f_{ij} que minimiza o custo total (Rubner

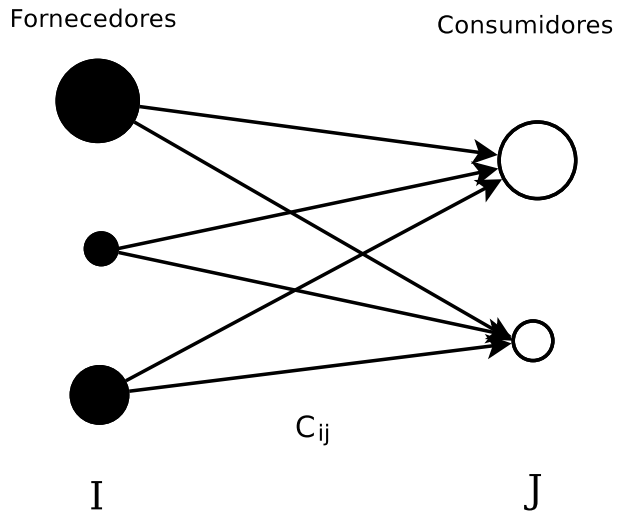


Figura 4.1: Exemplo de grafo bipartido com três fornecedores e dois consumidores. Extraída de Rubner et al. (1998).

et al., 1998) dado por:

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}f_{ij} \tag{4.1}$$

$$\text{restrito à: } f_{ij} \geq 0 \qquad i\in\mathcal{I}, j\in\mathcal{J} \tag{4.2}$$

$$\sum_{i\in\mathcal{I}} f_{ij} = con_j \qquad j\in\mathcal{J} \tag{4.3}$$

$$\sum_{j\in\mathcal{J}} f_{ij} \leq for_i \qquad i\in\mathcal{I} \tag{4.4}$$

no qual for_i é o total de suprimento do fornecedor i e con_j é o total da capacidade do consumidor j. A restrição apresentada pela Equação 4.2 permite o envio de suprimento do fornecedor até o consumidor e não o contrário. A Equação 4.3 restringe o consumidor a preencher sua total capacidade e a Equação 4.4 limita o fornecedor a enviar até a sua capacidade máxima. É razoável que a capacidade total do consumidor não exceda a de suprimentos do fornecedor:

$$\sum_{j\in\mathcal{J}} con_j \leq \sum_{i\in\mathcal{I}} for_i \tag{4.5}$$

O problema de fluxo mínimo pode ser naturalmente aplicado para o casamento de distribuições de probabilidade e por definição como uma distribuição de probabilidade do fornecedor e outra do consumidor, e o custo é c_{ij}, que é a distância entre o elemento i na primeira distribuição de probabilidade e o elemento j na segunda. Quando os pesos totais da distribuição de probabilidade não são iguais (casamento parcial), a menor distribuição de probabilidade será igual na ordem de satisfazer a condição viável. Dado o fluxo ótimo F, o EMD é definido como:

$$\text{EMD}(for, con) = \frac{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}f_{ij}}{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} f_{ij}} = \frac{\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}f_{ij}}{\sum_{j\in\mathcal{J}} con_j} \tag{4.6}$$

sendo o denominador é um fator normalizador que evita favorecer distribuições de probabilidade com pequenos pesos totais. Em geral a distância do chão c_{ij} pode ser qualquer distância e será escolhida de acordo com o problema em questão.

São 4 vantagens do EMD sobre as definições e maneira de tratar os dados:

1. O EMD aplica as distribuição de probabilidade, que são histogramas. A maior compactação e flexibilidade das distribuições de probabilidade é em si uma vantagem, e tendo uma métrica de distância que pode manipular esta estrutura de tamanho variável é importante.

2. O custo de mover terra reflete a noção da propriedade de proximidade.

3. O EMD permite o casamento parcial nos caminhos naturais. Isto é importante para negociar com oclusões e agrupamentos na tarefa de recuperação de imagem.

4. Se a distância chão é uma métrica e os pesos totais de duas distribuições de probabilidade são iguais, o EMD é uma verdadeira métrica.

Além disso a métrica pode ser computada rapidamente pois possui limitantes inferiores que calculam a distância entre seus centroides. Sejam v_i e q_j as coordenadas do *cluster* $\mathcal{C}_i$ na primeira distribuição de probabilidade, e o *cluster* $\mathcal{C}_j$ na segunda distribuição de probabilidade respectivamente. Então, usando a Equação 4.1 pode ser criado um limitante inferior para as computações de EMD serem reduzidas:

$$\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}f_{ij} = \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} \|v_i - q_j\| f_{ij} \tag{4.7}$$

$$= \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} \|f_{ij}(v_i - q_j)\| \qquad (f_{ij} \geq 0) \tag{4.8}$$

$$\geq \left\| \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} f_{ij}(v_i - q_j) \right\| \tag{4.9}$$

$$= \left\| \sum_{i\in\mathcal{I}} \left(\sum_{j\in\mathcal{J}} f_{ij} \right) v_i - \sum_{j\in\mathcal{J}} \left(\sum_{i\in\mathcal{I}} f_{ij} \right) q_j \right\| \tag{4.10}$$

$$= \left\| \sum_{i\in\mathcal{I}} for_i v_i - \sum_{j\in\mathcal{J}} con_j q_j \right\| \tag{4.11}$$

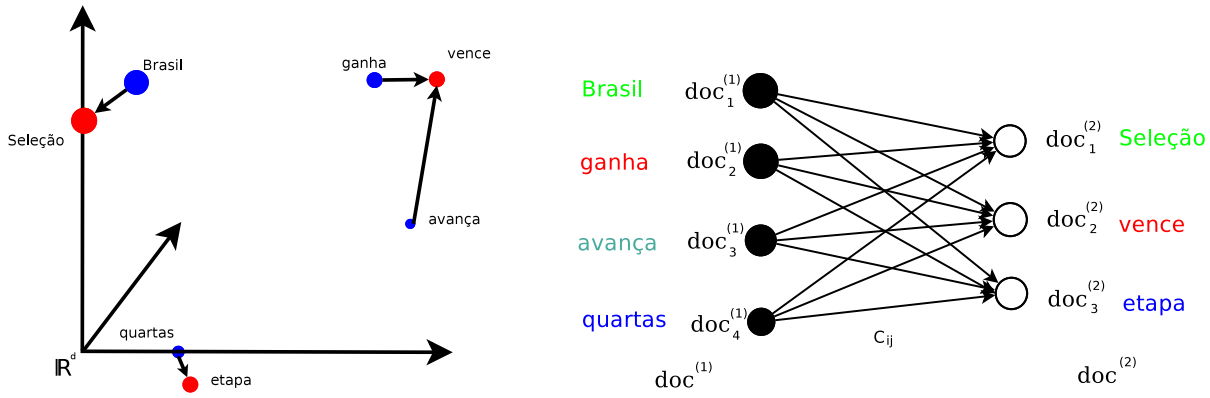
O algoritmo do EMD é uma inspiração para o WMD. O EMD foi utilizado para reconhecimento de padrões e em visão computacional também é encontrado no trabalho de Grauman e Darrell (2004), Wan (2007), Wan e Peng (2005). Na Seção 4.2 é descrita a métrica *Word Mover's Distance* derivada do EMD e que é um dos temas desse trabalho de pesquisa e dissertação.

4.2 *Word Mover's Distance*

O *Word Mover's Distance* (WMD) é uma métrica para calcular distância entre documentos. Um desafio na definição deste tipo de métricas é que documentos nem sempre usam as mesmas palavras para retratar o mesmo assunto. Um exemplo desta situação está nas seguintes frases: $doc^{(1)}$: Brasil

ganha e avança para as quartas. $doc^{(2)}$: Seleção vence e está na próxima etapa. As duas frases têm o mesmo sentido, mas usam palavras e expressões diferentes. Realizando um tratamento para retirar artigos, preposições e palavras frequentes (*stopwords*), os seus vetores não teriam similaridade alguma na representação BOW.

As informações semânticas e sintáticas dos *word embedding* podem ser usadas para enfrentar este problema. O WMD faz uso dos *word embedding* para definir uma medida de distância entre as palavras dos dois documentos. O WMD utiliza o algoritmo do EMD relatado na Seção 4.1 e adapta para o contexto de documentos. Fazendo uma comparação entre o WMD e o EMD, as distribuições de probabilidade do EMD são como as frequências normalizadas e a distâncias dos *word embeddings* do WMD são como o custo de enviar os suprimentos do método WMD. As representações que compõem o cálculo da distância WMD estão na Figura 4.2. O custo da viagem usando os *word embeddings* é mostrado na Figura 4.2a e a representação em grafo bipartido do fluxo das palavras entre os dois documentos na Figura 4.2b.



(a) Representação do Espaço de *word embeddings* com suas distâncias entre palavras similares.

(b) Grafo bipartido do WMD usando os documentos $doc^{(1)}$ e $doc^{(2)}$.

Figura 4.2: Caracterização da distância WMD entre os documentos $doc^{(1)}$ e $doc^{(2)}$.

Assim, os autores definem o custo de viagem entre duas palavras como a métrica da distância Euclidiana entre o *word embedding* da palavra i do $doc^{(1)}$ até a palavra j do $doc^{(2)}$: $c(i,j) = \|emb_i - emb_j\|_2$.

Esta abordagem rompe um dos problemas do BOW, quando suas representações possuem pouca ou nenhuma semelhança pois suas frases foram construídas com palavras diferentes, porém são documentos do mesmo assunto.

Para que a métrica fique completa, os autores utilizam a frequência normalizada[1] das palavras de cada documento para computar o transporte. Nesta representação um documento é um vetor $doc \in \mathbb{R}^m$, em que cada documento tem seu próprio valor para m (depende da quantidade de palavras do documento). Então, para cada palavra em um documento $doc^{(1)}$ é calculada como $doc_i^{(1)} = \frac{oc_i}{\sum_{j=1}^{m} oc_j}$, sendo oc_i a quantidade de vezes que a palavra ocorre no

[1]Nos artigos de origem, esse termo é encontrado como vetores BOW normalizado (nBOW) ou vetores BOW n-dimensionais normalizados, mas para melhor entendimento foi utilizado o termo frequência normalizada.

documento 1 e i é o índice da palavra dentro do vocabulário do documento. Portanto, o vetor para o $doc^{(1)}$ é:

$$[1/4, 1/4, 1/4, 1/4] \tag{4.12}$$

e para o $doc^{(2)}$:

$$[1/3, 1/3, 1/3] \tag{4.13}$$

A representação de frequência normalizada é usada para calcular a matriz de transporte T entre dois documentos. Uma matriz $T \in \mathbb{R}^{m \times m}$ diz quanto da palavra i em doc^a viaja até a palavra j em doc^b, sendo que $T_{ij} > 0$. A transformação de doc^a em doc^b assegura que em um fluxo total da palavra i é igual a doc_i^a, ou seja, $\sum_j T_{ij} = doc_i^a$. E o que a palavra j em doc^b pode receber é $\sum_i T_{ij} = doc_j^b$. Então a distância entre dois documentos pelo método WMD pode ser mensurada pelo mínimo acumulativo de T multiplicado pelo custo de viagem entre as palavras dos doc^a e doc^b (Kusner et al., 2015):

$$D(doc^a, doc^b) = \sum_{i,j} T_{ij} c(i, j) \tag{4.14}$$

Escolhendo uma das duas restrições sobre a distância, ela se torna um problema de programação linear:

$$\min_{T \geq 0} \sum_{i,j=1}^{m} T_{ij} c(i, j) \tag{4.15}$$

$$\text{restrito à: } \sum_{j=1}^{m} T_{ij} = doc_i \qquad \forall i \in \{1, \ldots, |\mathcal{I}_{doc^a}|\} \tag{4.16}$$

$$\sum_{i=1}^{m} T_{ij} = doc_j \qquad \forall j \in \{1, \ldots, |\mathcal{J}_{doc^b}|\} \tag{4.17}$$

onde $\mathcal{I}_{doc^a}$ é o conjunto das palavras do documento doc^a e $\mathcal{J}_{doc^b}$ o conjunto das palavras do documento doc^b, observe que m é depende do tamanho desses conjuntos.

A complexidade do algoritmo escala na ordem de $O(\rho^3 \log \rho)$, sendo ρ o número de palavras únicas nos documentos. Mas a métrica tem limitantes inferiores que permitem a poda no momento de computar o WMD.

O WMD tem alcançado bons índices de *accuracy* na tarefa de classificação de documentos, além da fácil abstração e uma métrica sem parâmetros (Kusner et al., 2015). Nos resultados do WMD de Kusner et al. (2015) mostra que os *word embeddings* do WORD2VEC são melhores do que os modelos propostos por Collobert e Weston (2008) e Mnih e Hinton (2009), por este motivo a pesquisa utilizou somente os *word embeddings* do WORD2VEC.

Word Centroid Distances (WCD)

A definição de Distâncias de Palavras Centroides é a diferença entre vetores de dois documentos. Cada vetor de um documento é representado pela média dos vetores das palavras que o compõem (Kusner et al., 2015), usando algum *word embedding*. O WCD está representado na Figura 4.3. A figura exibe a diferença entre os dois documentos $doc^{(1)}$ e $doc^{(2)}$ em um espaço vetorial $\mathbb{R}^d$.

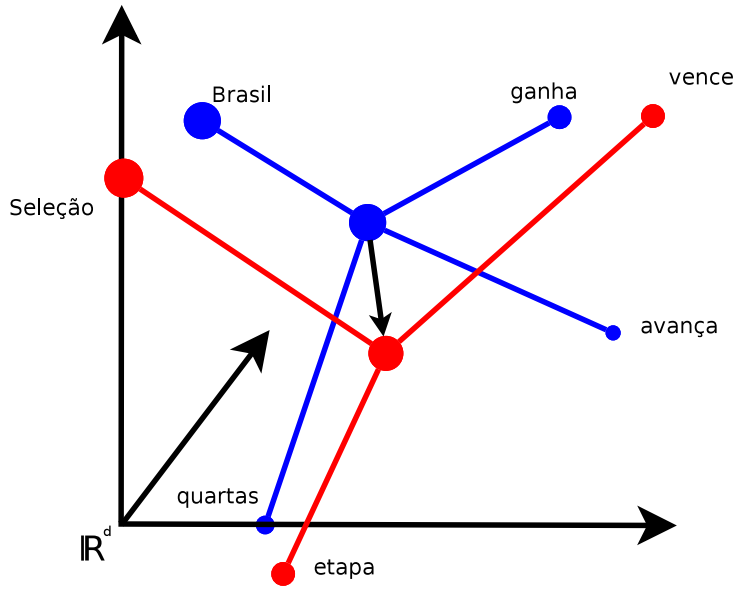


Figura 4.3: Representação do espaço de métrica WCD e suas distâncias entre a média dos seus vetores de palavras.

Seguindo a desigualdade triangular, o WCD é um limitante inferior para o WMD, na distância entre dois documentos (Kusner et al., 2015).

A complexidade de tempo do algoritmo que calcula esta medida de distância é barata e escalável, possui a ordem de $O(d_{emb}\rho)$. Desta modo, o WCD é usado para buscar os vizinhos mais próximos, e em seguida dos selecionados são calculados as distâncias WMD.

Apesar do WCD ser rápido, ela não é uma métrica suficientemente exata. Uma forma de obter uma aproximação mais justa é obedecendo apenas a uma restrição do problema de transporte (caso não obedeça a nenhuma requisição a matriz se torna $T = 0$, ou seja, todas as entradas são 0), então:

$$\min_{T \geq 0} \sum_{i,j=1}^{m} T_{ij} c(i,j) \tag{4.18}$$

$$\text{sujeito à: } \sum_{j=1}^{m} T_{ij} = doc_i^a \qquad \forall i \in \{1, \ldots, m\} \tag{4.19}$$

Uma matriz T^* é proposta para o problema de otimização com apenas uma restrição, e T^* é:

$$T^*_{ij} = \begin{Bmatrix} doc_i^a & \text{se } j = \arg\min_j c(i,j) \\ 0 & \text{caso contrário} \end{Bmatrix} \tag{4.20}$$

Fazendo esta restrição, a busca do problema sugere que o *word embedding* mais próximo de uma palavra do documento doc^a à outra palavra do documento doc^b seja o custo de transporte ótimo, ou seja, o *word embedding* mais parecido com um *word embedding* em outro documento é uma das soluções para o transporte. Se tomar a outra restrição no lugar da primeira também pode ser feita para obter uma otimização mais justa. Então, o valor máximo entre as duas otimizações obtidas é escolhido e formam a métrica nomeada por *Relaxed Word Mover's Distance* (RWMD).

Com dois limitantes para a métrica do WMD, alguns ajustes são feitos para otimizar o método de busca entre os documentos mais semelhantes. No

momento de comparação realizada pelo KNN são feitas algumas operações de ranqueamento entre um documento e o restante. Uma primeira ordenação é feita com o WCD em todos os documentos, em um segundo momento o RWMD elenca novamente os documentos para verificar se o k-ésimo elemento está na posição correta, caso esteja os demais são descartados, caso contrário o WMD é computado e atualiza os k primeiros documentos mais semelhantes.

4.3 *Supervised Word Mover's Distance*

O *Supervised Word Mover's Distance* (S-WMD) foi proposto como uma versão supervisionada da métrica de distância WMD. que pode ser aprimorada com o uso da supervisão. O S-WMD é uma métrica supervisionada que minimiza o erro sobre a classificação KNN. Ela realiza uma transformação sobre os *word embeddings* e outra sobre as frequências normalizadas para melhorar a classificação dos documentos (Huang et al., 2016).

A solução encontrada pelos autores foi utilizar o NCA para aplicar essa transformação sobre os *word embeddings* e embuti-la no treinamento supervisionado da métrica. Para a transformação das frequências normalizadas foi propostos um vetor w resultante de um gradiente que trabalha junto com o NCA para ponderar a importância das palavras. Essas transformações podem ser visualizadas na Figura 4.4. O gradiente produz uma matriz A e e um vetor de pesos w.

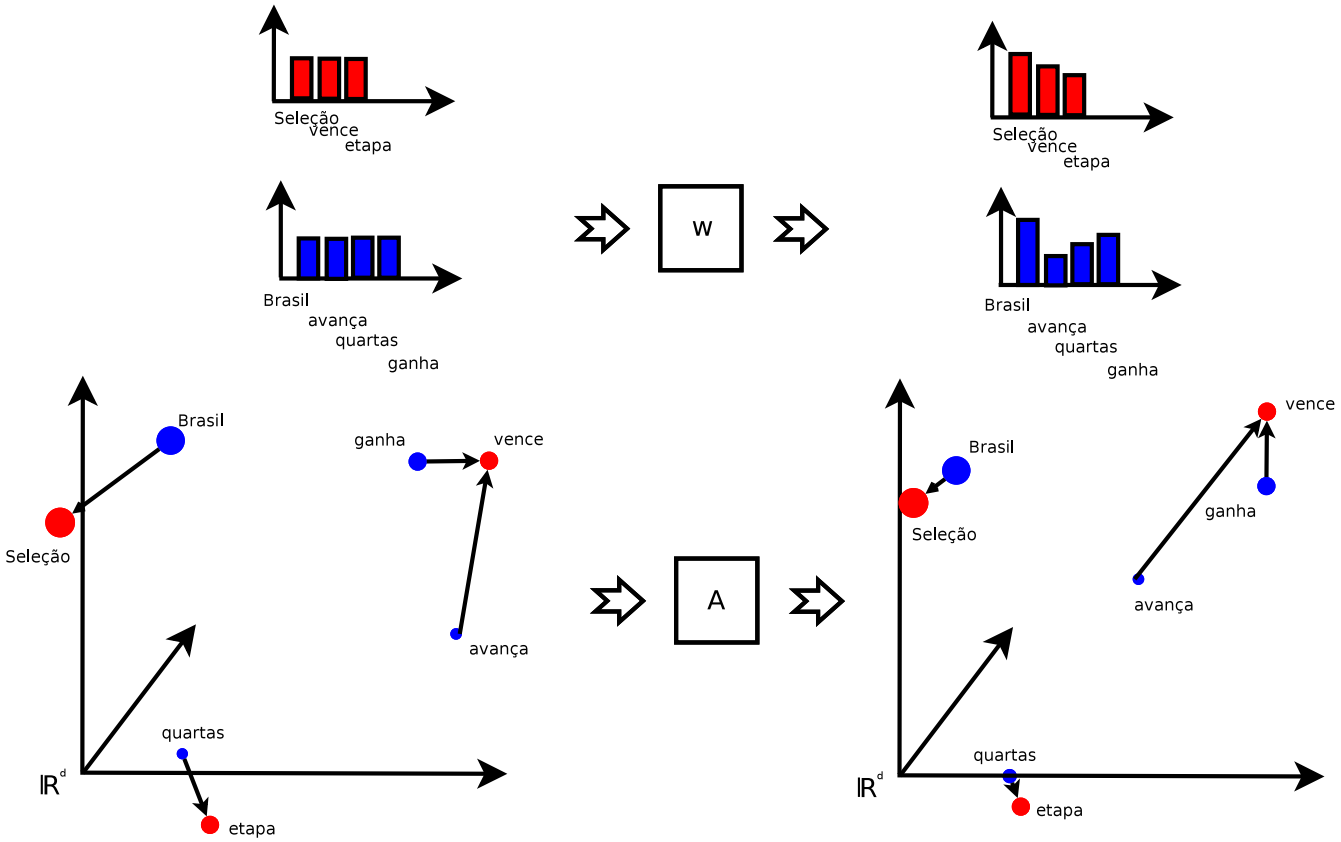


Figura 4.4: Transformações sobre os dados para produção das distância do método S-WMD.

A abordagem tem como objetivo melhorar a métrica entre dois documentos distanciando os que possuem uma classe diferente e aproximando os que pos-

suem a mesma classe. Para isso a métrica usa a noção de similaridade nas seguintes formas:

- Uma transformação linear é aplicada ao *word embedding* que irá capturar sua representação:

$$\tilde{emb}_i \rightarrow Aemb_i, \text{ onde } emb_i \text{ é o } i\text{-ésimo } \textit{word embedding} \tag{4.21}$$

- Para ponderar a importância das palavras, dentro da tarefas de classificação é inserido um vetor w ajustando os valores das frequências normalizadas:

$$\tilde{doc}^a = \frac{(w \circ doc^a)}{(w^\top \cdot doc^a)}, \text{ onde } \circ \text{ é o produto elemento a elemento de Hadamard.} \tag{4.22}$$

O produto de *Hadamard* na prática é a multiplicação da frequência normalizada das palavras do doc^a pelos pesos das palavras respectivas em w, não com todos os elementos do vetor w.

Considerando o vetor de pesos w e a matriz e aprendizado A, a distância entre documentos doc^a e doc^b é (respeitando a somente uma das restrições):

$$\mathcal{D}_{A,w}(doc^a, doc^b) \triangleq \min_{T \geq 0} \sum_{i,j=1}^{m} T_{ij} \left\| A(emb_i - emb_j) \right\|_2^2 \tag{4.23}$$

$$\text{restrito à: } \sum_{j=1}^{m} T_{ij} = \tilde{doc}_i^a, \tag{4.24}$$

$$\sum_{i=1}^{m} T_{ij} = \tilde{doc}_j^b \qquad \forall i, j \tag{4.25}$$

A matriz T de transporte é aproximada por uma fórmula relaxada. O algoritmo *sinkhorn*[2] proposto por Cuturi e Doucet (2014) resolve o problema do fluxo mínimo usando escalonamento de matrizes que produz essa aproximação utilizando esta fórmula relaxada.

Huang et al. (2016) desenvolve uma função que aprende a matriz A e o vetor w de tal modo que respeite as similaridades entre os documentos classificados. A função deve minimizar também a taxa de erro da classificação *Leave-one-out* do kNN. A taxa de erro do LOO kNN não é diferenciável, então foi usado a *KL-divergence* como função de perda do NCA, pois assim como no NCA é uma forma de relaxar a vizinhança entre os dados:

$$\ell(A, w) = -\sum_{a=1}^{n} \log \left(\sum_{b:y_b = y_a}^{n} \frac{\exp(-\mathcal{D}_A(doc^a, doc^b))}{\sum_{c \neq a} \exp(-\mathcal{D}_{A,w}(doc^a, doc^c))} \right) \tag{4.26}$$

A computação do gradiente pode ser feita considerando A e w como:

$$\frac{\partial}{\partial(A, w)} \ell(A, w) = \sum_{a=1}^{n} \sum_{b \neq a} \frac{p_{ab}}{p_a} (\delta_{ab} - p_a) \frac{\partial}{\partial(A, w)} \mathcal{D}_{A,w}(doc^a, doc^b) \tag{4.27}$$

[2] http://marcocuturi.net/SI.html

onde $\delta_{ab} = 1$ se e somente se $y_a = y_b$, e $\delta_{ab} = 0$ caso contrário. Na Equação 4.27 p_{ab} e p_b são probabilidades do NCA como na Equação 2.8 e Equação 2.9. Reescrevendo estas equações nas notações usadas nesta seção, tem-se a Equação 4.28 e Equação 4.29.

$$p_{ab} = \frac{\exp(-\mathcal{D}_{A,w}(doc^a, doc^b))}{\sum_{k \neq a} \exp(-\mathcal{D}_{A,w}(doc^a, doc^k))}, p_{aa} = 0 \tag{4.28}$$

$$p_a = \sum_{b \in C_a} p_{ab} \tag{4.29}$$

$$C_a = \{b | c_a = c_b\} \tag{4.30}$$

Algoritmo 9: Algoritmo de treinamento do S-WMD.

Entrada: o conjunto de treinamento T_{treino} com seus documentos doc rotulados Y; o conjunto de *word embeddings* $\mathcal{W}$; η_A é a taxa de aprendizado do gradiente de A; η_w é a taxa de aprendizado do gradiente de w.

Resultado: Matriz A de transformação das palavras e o vetor w com o peso da importância das palavras.

1 $dataset : \{(doc^1, y^{(1)}), \ldots, (doc^n, y^{(n)})\}$;
2 $c^a = \mathcal{W} doc^a, \forall a \in \{1, \ldots, n\}$;
3 $A = NCA((c^1, y^{(1)}), \ldots, (c^n, y^{(n)}))$;
4 $w = 1$;
5 **enquanto** *não convergir* **faça**
6 Selecione aleatoriamente $\mathcal{B} \in \{1, \ldots, n\}$;
7 Compute os gradientes usando a Equação 4.31;
8 $A \leftarrow A - \eta_A g_A$;
9 $w \leftarrow w - \eta_w g_w$;
10 **fim**

Para uma computação mais rápida do gradiente, há uma inicialização da matriz A do algoritmo e o uso do gradiente descendente em porções do conjunto de treinamento.

No Algoritmo 9 é exibido os passos para realizar o treino do método do S-WMD. A Equação 4.26 não é convexa e é altamente dependente da correta inicialização de A, w. O vetor w é inicializado com 1 em todas as suas entradas, essa atribuição é feita na linha 4 do Algoritmo 9. A matriz A é inicializada aplicando os WCD's dos documentos de treinamento ao algoritmo NCA. Como visto na Seção 2.2.3, o NCA pode ser usado para reduzir a dimensionalidade dos dados. Para encontrar o melhor valor da redução, os autores usam um algoritmo de otimização de parâmetros Bayesiana (Gardner et al., 2014). Na linha 6, $\mathcal{B}$ é uma porção de exemplos menor do que n e são selecionados aleatoriamente para ser dada a computação do gradiente.

Os resultados da classificação de documentos com o S-WMD foi comparada a 26 métodos em 8 *datasets* nos quais se mostrou superior.

Supervised Word Centroids Distance

O WCD é uma aproximação razoável para o WMD (Kusner et al., 2015). A transformação dos WCD's pela matriz gerada do NCA é chamada de *Supervi-*

sed Word Centroids Distance (S-WCD) e também é uma métrica de distância que pode ser usada para comparar exemplos de treino e teste afim de realizar uma classificação com o algoritmo do kNN. A Figura 4.5 exibe a transformação do espaço métrico do WCD para o S-WCD pela multiplicação da matriz A.

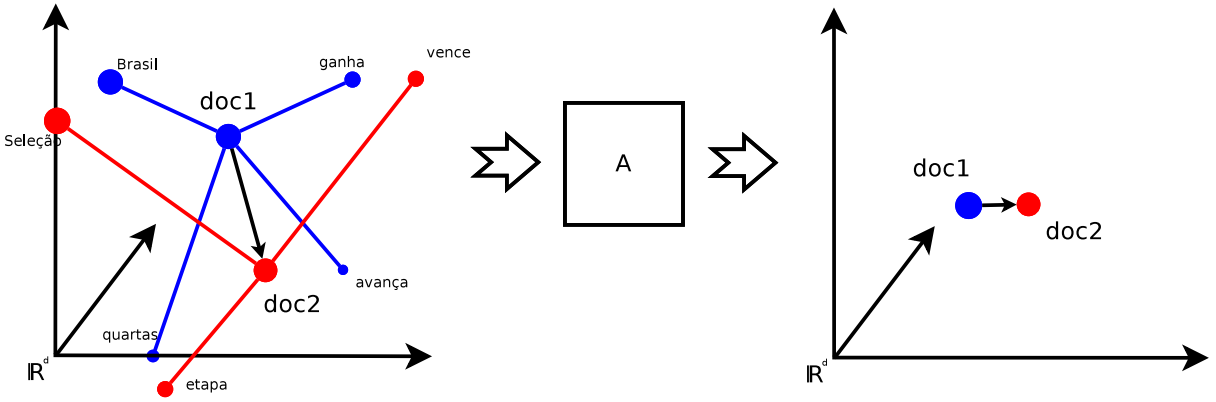


Figura 4.5: Transformação do WCD em S-WCD pela matriz A.

Reescrevendo a Equação 4.27, usando porções de tamanho $|\mathcal{B}|$ para o treinamento do gradiente por seleção aleatória, como feito na linha 6 do Algoritmo 9, e computa-se esse grupo. Ainda são selecionados os WCD mais próximos de doc^a para computar o gradiente e as probabilidades vizinhas do NCA, formando o grupo $\mathcal{N}_a$. O gradiente em relação a A, w torna-se:

$$g_{A,w} = \sum_{a \in \mathcal{B}} \sum_{b \in \mathcal{N}_a} \frac{p_{ab}}{p_a}(\delta_{ab} - p_a) \frac{\partial}{\partial(A,w)} \mathcal{D}_{A,w}(doc^a, doc^b) \tag{4.31}$$

A Figura 4.6 apresenta a visão mais externa do algoritmo separada em treino e teste com o kNN. Como saída o algoritmo produz a matriz A que é a computada pelo NCA dentro do algoritmo do gradiente do S-WMD e o vetor w que possui o peso para cada palavra do *dataset*. Então na etapa de teste, essas duas saídas são usadas para computar a distância dos documentos de treino com os de teste.

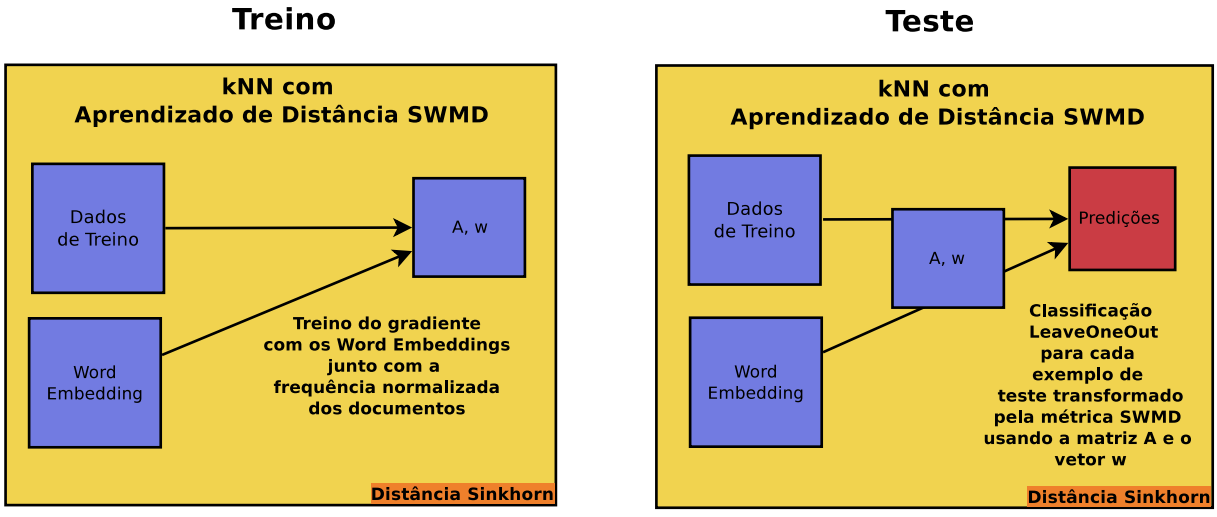


Figura 4.6: Representação gráfica do processo de construção da distância S-WMD.

A distância é definida pela soma de todas as entradas da matriz resultante do algoritmo de transporte de Cuturi e Doucet (2014). A distância proporcionada pelo *sinkhorn* leva como entrada de dois documentos seus *word embeddings* multiplicados pela matriz A e a frequência normalizada multiplicada elemento a elemento pelo vetor w do algoritmo S-WMD, o resultado é a distância entre os dois documentos. A Figura 4.4 exibe as transformações realizadas pelo métodos sobre a frequência normalizada e sobre os *word embeddings*.

CAPÍTULO

5

Avaliação Experimental

Uma melhor representação de exemplos pode ser benéfica para os algoritmos de Seleção de Instâncias. Assim este capítulo apresenta a avaliação experimental buscando responder a pergunta: a combinação de Seleção de Instâncias, Aprendizado de Métrica e WMD por meio da junção de S-WMD e Seleção de Instâncias pode trazer algum ganho? Assim a avaliação compara os seguintes cenários buscando responder parcialmente cada elemento da combinação com as perguntas:

1. *word embedding* pode melhorar métodos de Seleção de Instâncias? Para essa avaliação foram realizados experimentos comparando a Seleção de Instâncias com e sem *word embedding*.

2. Aprendizado de Métrica pode melhorar a combinação de *word embedding* + Seleção de Instâncias? Para essa avaliação foram realizados experimentos comparando *word embedding* + Seleção de Instâncias com e sem Aprendizado de Métrica.

3. Qual é a melhor combinação de método de Seleção de Instâncias e de representação de dados? Para essa avaliação foram tabulados os resultados anteriores, mas dispostos em uma tabela que permite comparar representação e Seleção de Instâncias.

Com foco nestas três perguntas a avaliação experimental foi conduzida verificando os seguintes critérios:

1. Desempenho do classificador (*precision*,*recall*,*accuracy*,*f1-score*);

2. Redução do conjunto de treinamento.

3. Tempo de execução;

5.1 Critérios de Avaliação

Cada combinação de representação requer uma configuração diferente para avaliação do teste. Assim, para facilitar a compreensão deste capítulo a seguir são listados as representações utilizadas no trabalho, e como foi realizado o treinamento e teste de cada representação.

- BOW: a seleção é feita sobre o BOW do conjunto de treinamento usando a distância Euclideana, em seguida para a predição do KNN é usada a distância Euclideana para comparação dos exemplos. A Figura 5.1 apresenta a configuração de treino e teste.

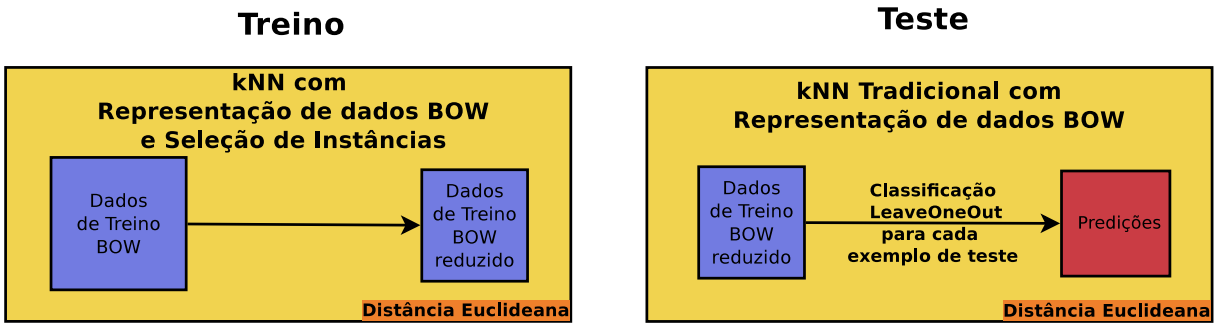


Figura 5.1: Configuração de treino e teste para a representação BOW sobre o KNN usando distâncias Euclideanas.

- WCD: a seleção é feita sobre o WCD do conjunto de treinamento usando a distância Euclideana, em seguida para a predição do KNN é usada a distância Euclideana para comparação dos exemplos. A configuração desta representação é apresentado na Figura 5.2.

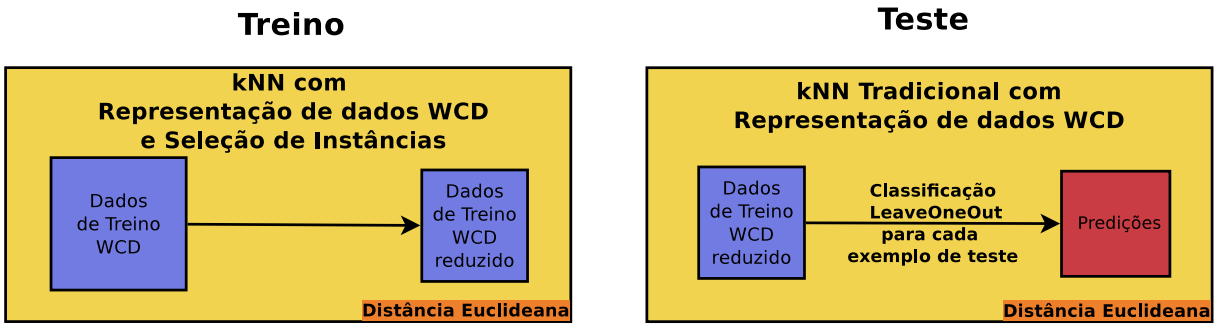


Figura 5.2: Configuração de treino e teste para a representação WCD sobre o KNN usando distâncias Euclideanas.

- S-WMD: a seleção é feita sobre o WCD do conjunto de treinamento usando a distância Euclideana, no passo seguinte para a predição do KNN é usada a distância *sinkhorn* para comparação dos exemplos, a mesma usada na predição dos exemplos do S-WMD vistos na Seção 4.3. A Figura 5.3 mostra os passos para realizar seleção de instância com o S-WMD.

- S-WCD: a seleção é feita sobre o WCD depois de multiplicado matricialmente ou distorcido pela matriz *Mahalanobis* resultante do S-WMD

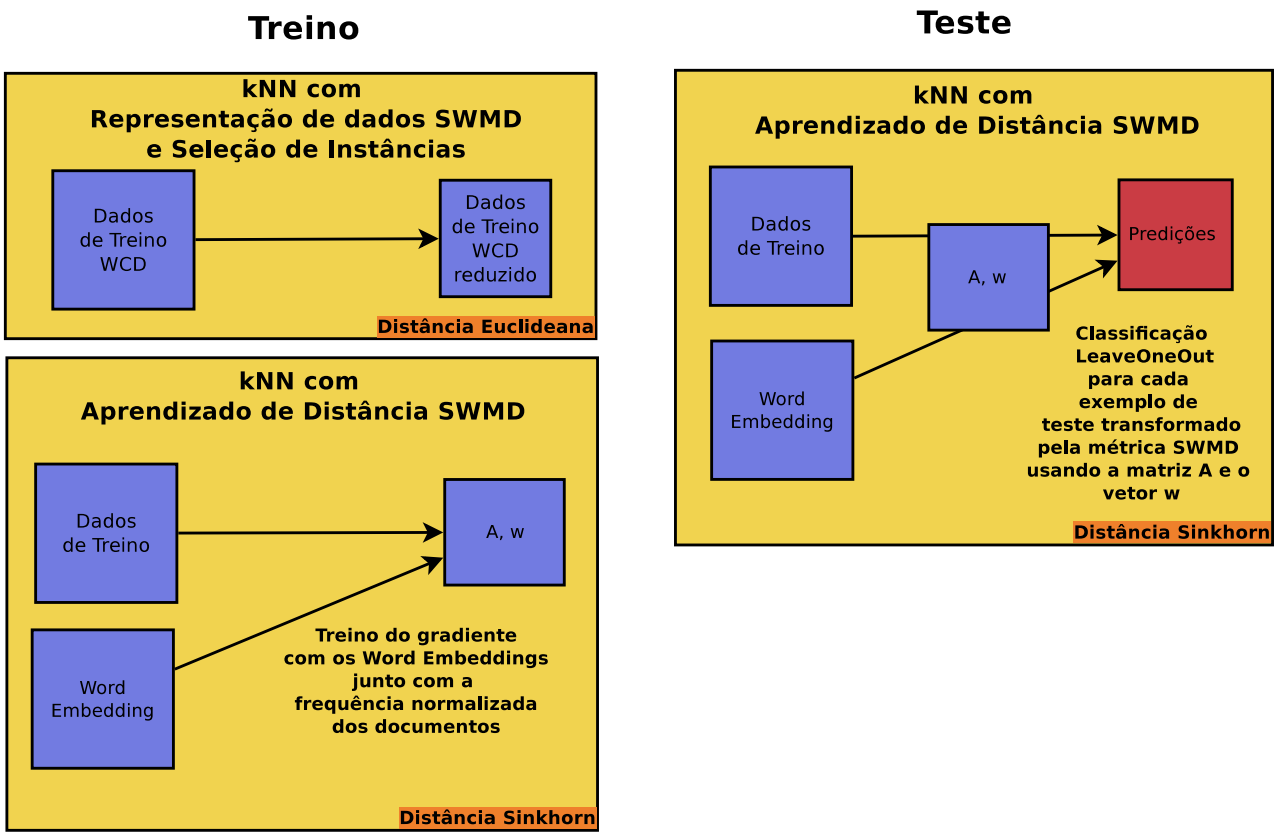


Figura 5.3: Configuração de treino e teste para a distância S-WMD sobre o kNN usando distâncias Euclideanas.

usando a distância Euclideana, em seguida para a predição do kNN é usada a distância Euclideana para comparação dos exemplos. A Figura 5.4 exibe a configuração de treino e teste sobre o kNN usando representações S-WCD.

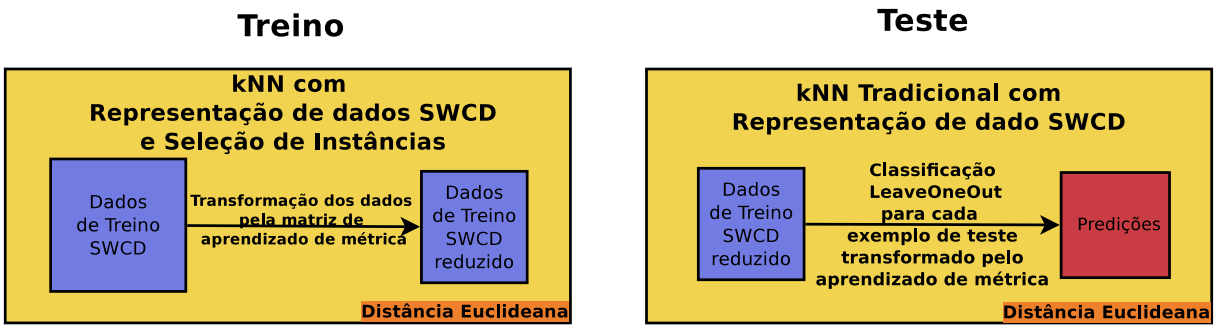


Figura 5.4: Configuração de treino e teste para a representação S-WCD sobre o kNN usando distâncias Euclidianas.

5.2 Gráficos de Avaliação Multi-critério

Para facilitar a avaliação multi-critério foram utilizados gráficos similares ao gráficos ROC onde os valores no eixo y, quanto maior melhor; e os valores no eixo x, quanto menor melhor. Assim os melhores resultados estão próximos ao ponto (0,1) que representa o canto superior esquerdo. A Figura 5.5 ilustra um gráfico com esta representação. Os gráficos apresentados neste

capítulo ilustram os seguintes critérios: desempenho vs número de exemplos selecionados, desempenho vs tempo, e redução vs tempo.

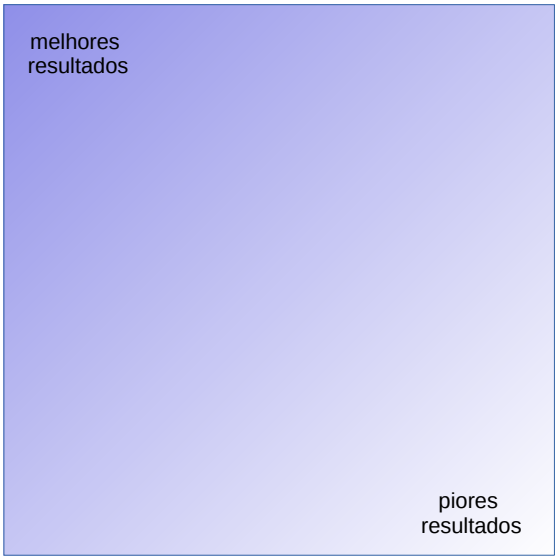


Figura 5.5: Exemplo de gráfico de avaliação multi-critério utilizado neste trabalho. Eixo y, quanto maior melhor; eixo y, quanto menor melhor

Nos gráficos também são ilustrados o fecho convexo, que auxilia a indicar os pontos com melhor contra-partida (*trade-off*) entre os critérios avaliados. Além disso, há uma linha com traços que indica o desempenho da representação ou método sem a Seleção de Instâncias em cada gráfico que mensura desempenho de classificação.

5.3 Descrição dos *Datasets* Textuais Utilizados

Os *datasets* usados para avaliação foram o BBC SPORT, TWITTER e REUTERS[1]. A Tabela 5.1 exibe as informações sobre as classes, dimensões na representação BOW, quantidades de documentos e médias de palavras por documento para cada *dataset*.

Nome	Descrição	Classes	nº ex. treino	nº ex. validação	nº ex. teste	Total	BOW dim.	Média de palavras por documento	Classe Majoritária (%)
BBC SPORT	Artigos rotulados por esporte	5	414	103	220	737	13243	117	35,96
TWITTER	Tweets categorizados por sentimento	3	1741	435	932	3108	6344	9,9	68,82
REUTERS	Artigos de notícia	8	4388	1098	2189	7624	31789	59,2	51,12

Tabela 5.1: Descrição dos *datasets*.

Os *datasets* BBC SPORT e TWITTER encontram-se pré-configurados para realizar validação cruzada k-*fold* para $k = 5$ e somente o *dataset* REUTERS possui a configuração *hold-out*, onde os conjuntos são separados apenas em treino, validação e teste. Os *datasets* utilizados foram obtidos dos trabalhos de (Kusner et al., 2015; Huang et al., 2016) e seguem esta configuração. A seleção

[1]Os *datasets* podem ser encontrados no link: https://github.com/gaohuang/S-WMD#paper-datasets.

é realizada somente do conjunto de treinamento, não seleciona exemplos do conjunto de validação.

A Tabela 5.2 detalha as classes e a distribuição dos exemplos em cada *dataset* utilizado. A primeira coluna da esquerda para a direita contém os nomes dos *datasets*, seguido da quantidade de exemplos e coluna com as percentagens da classe em relação ao total e finalizando pelo nome da classe.

dataset	# Exemplos	%	Classes
REUTERS	3923	51,12	earn
	2292	29,87	acq
	326	4,25	trade
	144	1,88	ship
	51	0,66	grain
	374	4,87	crude
	271	3,53	interest
	293	3,82	money-fx
	7624	**100**	**todas**
BBC SPORT	101	13,7	athletics
	124	16,82	cricket
	265	35,96	football
	147	19,95	rugby
	100	13,57	tennis
	737	**100**	**todas**
TWITTER	461	14,83	positivo
	508	16,83	negativo
	2139	68,82	neutro
	3108	**100**	**todas**

Tabela 5.2: Distribuição dos exemplos por classes dos *datasets*: BBC SPORT, TWITTER e REUTERS.

Foram utilizados três modelos de computadores para realizar os experimentos e são descritos na Tabela 5.3 as características da CPU e sua capacidade de processamento e também uma coluna com a memória RAM.

Arquitetura	CPU	Potência da CPU	Memória
1	i7-7700HQ	2.80 GHz $\sim$ 3,80 GHz	8GB
2	i3-3220	3.30 GHz	4GB
3	i5-3230M	2.60 GHz $\sim$ 3,20 GHz	12GB

Tabela 5.3: Capacidade dos processadores e memórias utilizados nos experimentos.

5.4 Resultados

Com foco nas três perguntas definidas no começo deste capítulo, esta seção apresenta os resultados obtidos.

Quanto aos hiper-parâmetros dos algoritmos de Seleção de Instâncias podem ser feitas as seguintes considerações: o SIL e o RMHC possuem parâmetros para escolher a quantidade de exemplos a serem selecionados. Nos

experimentos avalidados neste trabalho foram testados 30%, 50%, 70% e 90% de redução do conjunto de treinamento. O RMHC ainda possui um meta-parâmetro para quantidade de iterações do algoritmo, que por recomendações o seu valor deve ser de 10000 (Garain, 2008), porém devido ao alto custo computacional foi utilizado o valor de 1000 iterações. Em quase todos os experimentos o RMHC se mostrou superior aos outros métodos mesmo usando apenas 1000 iterações.

Os algoritmos do CNN e ENN percorrem uma vez todo conjunto de treinamento para Seleção de Instâncias, assim não possuem hiper-parâmetros quanto ao número de exemplos selecionados. Tanto o CNN, quanto o ENN apresentaram valores diferentes de instâncias selecionadas para cada fold nos experimentos, e o que é mostrado nos gráficos é a média dos exemplos selecionados.

O conjunto de treinamento e validação dos *datasets* foram usados para a validação do hiper-parâmetro k do algoritmo do KNN, que variou entre os ímpares de 1 até 19. Esses hiper-parâmetros são os mesmos utilizados nos trabalhos de (Huang et al., 2016; Kusner et al., 2015).

Os hiper-parâmetros selecionados para cada *dataset* variam de acordo com as técnicas de Seleção de Instâncias avaliadas (e sua porcentagem de seleção configuradas para o SIL e o RMHC) e também para cada representação de dados. Na Tabela 5.4 está resumido os parâmetros para o *dataset* REUTERS. Na Tabela 5.5 está resumido os parâmetros para o *dataset* BBC SPORT. Na Tabela 5.6 está resumido os parâmetros para o *dataset* TWITTER.

		BOW	WCD	S-WCD	S-WMD
CNN		17	11	13	9
ENN		-	15	3	7
SIL	30%	1	3	3	15
	50%	1	5	1	7
	70%	1	1	9	11
	90%	1	3	1	5
RMHC	30%	1	9	3	5
	50%	1	7	3	5
	70%	1	1	3	5
	90%	1	1	3	5

Tabela 5.4: Valores de k do hiper-parâmetro do KNN selecionados para o *dataset* REUTERS.

A métrica de distância WMD foi excluída pois apresenta grande custo computacional exigindo mais tempo de pesquisa e possui resultados semelhantes e/ou piores do que o S-WMD (Huang et al., 2016).

Nos experimentos realizados por este trabalho foram utilizados *datasets* que possuem uma configuração multi-classe. As métricas de *precision* e *recall* ponderadas foram utilizadas para avaliar o desempenho dos métodos de Seleção de Instâncias.

5.4.1 *Word Embedding* Melhora Métodos de Seleção de Instâncias?

Nesta seção são comparados os resultados de métodos de Seleção de Instâncias sem e com *word embeddings* utilizando a representação WCD. Na Ta-

		BOW	WCD	S-WCD	S-WMD
CNN		3	1	1	1
ENN		1	19	5	3
SIL	30%	1	11	1	5
	50%	1	7	3	3
	70%	1	15	1	1
	90%	1	1	1	1
RMHC	30%	15	1	7	3
	50%	3	5	9	3
	70%	1	19	1	5
	90%	1	1	1	3

Tabela 5.5: Valores de k do hiper-parâmetro do kNN selecionados para o *dataset* BBC Sport.

		BOW	WCD	S-WCD	S-WMD
CNN		3	19	7	1
ENN		19	9	13	19
SIL	30%	19	19	19	17
	50%	19	15	17	17
	70%	19	15	15	17
	90%	15	19	19	19
RMHC	30%	19	19	17	17
	50%	15	17	17	19
	70%	13	19	15	15
	90%	13	13	7	19

Tabela 5.6: Valores de k do hiper-parâmetro do kNN selecionados para o *dataset* Twitter.

bela 5.7 está apresentado os resultados de métrica de *f1-score* para o *dataset* Reuters. Da esquerda para direita a tabela compara: número de exemplos selecionados (#Selecionados), o desempenho na classificação para a métrica *f1-score*. Observando todas as entradas da tabela é possível ver que o WCD apresenta ganhos em todas as técnicas de Seleção de Instâncias. Outro resultado interessante é a quantidade de exemplos selecionado pelo BOW na seleção do CNN é reduzido de 834 para 580 e ganho em aproximadamente 10 pontos na medida de desempenho *f1-score*.

Na Tabela 5.8 está exibido também os valores da medida de desempenho *f1-score* para o *dataset* BBC Sport. Nesta tabela, a quantidade de exemplos selecionados pelo método ENN com representação WCD aumentou significativamente, no entanto, o ganho nessa métrica de desempenho foi de pouco mais de 25%.

Na Tabela 5.9 é mostrado os resultados para o *dataset* do Twitter. Os valores apresentam ganhos na maioria dos resultados, em apenas dois casos o desempenho foi inferior, porém com pouca diferença.

Na Tabela 5.10 que segue o mesmo entendimento que a 5.7 e também está comparado o desempenho de BOW e WCD para mais métricas: *accuracy*[2],

[2]Disponível em http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html.

precision[3] ponderada, *recall*[4] ponderada e *f1-score*.

#Selecionados: o destaque para o número de exemplos selecionado é o algoritmo CNN. A redução para REUTERS foi de 86,78% do conjunto de treinamento (4388 exemplos para 580 exemplos com a redução) e BBC SPORT foi de 83,27% do conjunto de treino (414 exemplos para 69,4 exemplos com a redução). Com reduções drásticas, ainda assim o algoritmo consegue melhorar o seu desempenho de classificação em todas as métricas avaliadas.

***accuracy*, *precision*, *recall* e *f1-score*:** independente da métrica, WCD predomina com os melhores resultados nos três conjuntos de dados avaliados. Em todos os *datasets* há ganhos em *accuracy* de 1% a 30% ao se utilizar dos *word embeddings* para Seleção de Instâncias. Um ganho médio de 13% em *accuracy* só no *dataset* REUTERS utilizando os WCD com Seleção de Instâncias e no BBC SPORT para a mesma medida de desempenho um ganho médio de 11%. No *dataset* TWITTER não houve um ganho acentuado, porém com valor médio de 1,87% a mais que o BOW.

Com estes resultados o ganho de desempenho pelo uso de *word embeddings* é bem evidente.

No *dataset* REUTERS não existem valores para o ENN com representação de dados BOW, por restrições computacionais da pesquisa relacionadas a memória RAM. Isso também ocorre no gráfico da Figura 5.9a e nas tabelas a seguir.

		#Selecionados		*f1-score*	
	Mét.	BOW	WCD	BOW	WCD
REUTERS	CNN	834	580	83.28	**93.22**
	ENN	-	4133	-	**93.61**
	RMHC	3071	3071	85.25	**94.34**
		2194	2194	85.02	**93.59**
		1316	1316	85.99	**93.21**
		438	438	84.49	**93.14**
	SIL	3072	3072	82.87	**93.76**
		2194	2194	73.73	**93.49**
		1317	1317	65.58	**90.57**
		439	439	49.46	**88.39**

Tabela 5.7: Tabela com os resultados de *f1-score* para o *dataset* REUTERS.

[3]Disponível em http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html, foi usado o parâmetro *weighted* para produção dos resultados de *precision*.

[4]Disponível em http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html, foi usado o parâmetro *weighted* para produção dos resultados de *recall*.

		#Selecionados		*f1-score*	
	Mét.	BOW	WCD	BOW	WCD
BBC SPORT	CNN	116,6	69,4	71,1 ±4,6	**73,32 ±5,92**
	ENN	310,6	372,6	62,84 ±2,31	**85,22 ±1,81**
	RMHC	289	289	77,28 ±7,05	**87,28 ±0,74**
		207	207	74,62 ±3,7	**84,7 ±1,17**
		124	124	76,29 ±2,71	**84,01 ±1,39**
		41	41	73,13 ±3,48	**81,14 ±0,99**
	SIL	290	290	73 ±1,41	**84,37 ±3,00**
		207	207	68,36 ±4,11	**81,87 ±3,7**
		125	125	59,54 ±6,43	**77,79 ±3,94**
		42	42	46,82 ±7,18	**55,34 ±3,16**

Tabela 5.8: Tabela com os resultados de *f1-score* para o *dataset* BBC SPORT.

		#Selecionados		*f1-score*	
	Mét.	BOW	WCD	BOW	WCD
TWITTER	CNN	310	343,6	**57,29 ±1,68**	56,6 ±0,7
	ENN	1150,4	1263,8	60,09 ±1,04	**63,08 ±1,17**
	RMHC	1218	1218	59,91 ±1,54	**63,95 ±1,41**
		870	870	58,78 ±1,25	**63,94 ±1,58**
		522	522	58,11 ±0,60	**62,07 ±0,67**
		174	174	57,61 ±1,42	**57,87 ±1,72**
	SIL	1219	1219	58,37 ±2,05	**64,48 ±0,94**
		871	871	57,75 ±1,17	**63,26 ±0,96**
		523	523	**58,18 ±1,83**	58 ±1,46
		175	175	56,08 ±0,80	**56,19 ±0**

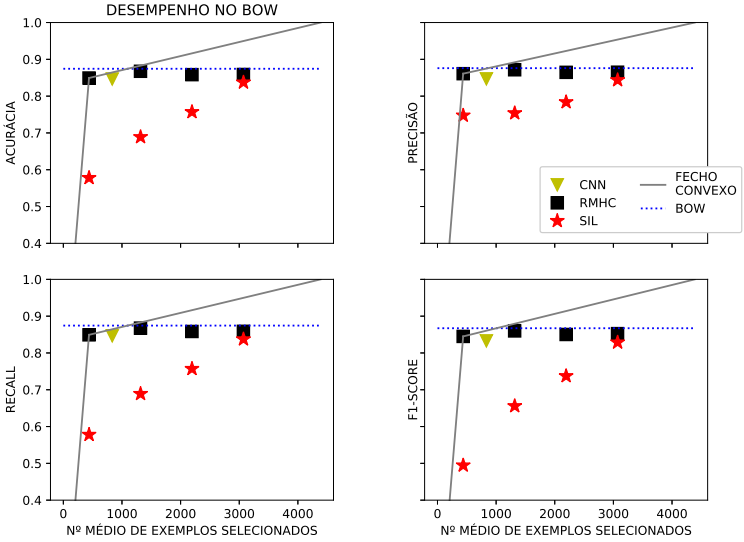
Tabela 5.9: Tabela com os resultados de *f1-score* para o *dataset* TWITTER.

		#Selecionados		*accuracy*		*precision*		*recall*		*f1-score*	
	Mét.	BOW	WCD	BOW	WCD	BOW	WCD	BOW	WCD	BOW	WCD
REUTERS	CNN	834	580	84,65	**93,51**	84,69	**94,00**	84,65	**93,51**	83,28	**93,22**
	ENN	-	4133	-	**93,74**	-	**94,19**	-	**93,74**	-	**93,61**
	RMHC	3071	3071	85,88	**94,38**	86,50	**94,55**	85,88	**94,38**	85,25	**94,34**
		2194	2194	85,84	**93,65**	86,47	**93,96**	85,84	**93,65**	85,02	**93,59**
		1316	1316	86,75	**93,24**	87,17	**93,4**	86,75	**93,24**	85,99	**93,21**
		438	438	84,92	**93,19**	86,10	**93,41**	84,92	**93,19**	84,49	**93,14**
	SIL	3072	3072	83,74	**93,74**	84,36	**94,00**	83,74	**93,74**	82,87	**93,76**
		2194	2194	75,70	**93,6**	78,38	**93,84**	75,70	**93,60**	73,73	**93,49**
		1317	1317	68,89	**90,54**	75,38	**90,96**	68,89	**90,54**	65,58	**90,57**
		439	439	57,79	**88,17**	74,75	**89,42**	57,79	**88,17**	49,46	**88,39**
BBC SPORT	CNN	116,6	69,4	70,91 ±4,65	**72,73 ±6,33**	76,2 ±4,64	**77,19 ±3,59**	70,91 ±4,65	**72,73 ±6,33**	71,1 ±4,6	**73,32 ±5,92**
	ENN	310,6	372,6	64,27 ±2,55	**85,27 ±1,88**	79,36 ±1,08	**86,25 ±2,14**	64,27 ±2,55	**85,27 ±1,88**	62,84 ±2,31	**85,22 ±1,81**
	RMHC	289	289	77,73 ±6,66	**87,18 ±0,78**	84,32 ±2,27	**87,78 ±0,69**	77,73 ±6,66	**87,18 ±0,78**	77,28 ±7,05	**87,28 ±0,74**
		207	207	74,36 ±5,17	**84,82 ±1,24**	80,75 ±1,57	**85,49 ±1,17**	74,36 ±5,17	**84,82 ±1,24**	74,62 ±3,7	**84,7 ±1,17**
		124	124	76,46 ±2,63	**84,00 ±1,64**	80,23 ±1,57	**85,59 ±0,69**	76,46 ±2,63	**84 ±1,64**	76,29 ±2,71	**84,01 ±1,39**
		41	41	73,46 ±3,48	**81,00 ±1,05**	78,58 ±1,49	**81,9 ±1,18**	73,46 ±3,48	**81 ±1,05**	73,13 ±3,48	**81,14 ±0,99**
	SIL	290	290	73,27 ±1,66	**84,45 ±3,03**	78,96 ±1,82	**85,15 ±3,12**	73,27 ±1,66	**84,45 ±3,03**	73 ±1,41	**84,37 ±3,00**
		207	207	69,00 ±3,84	**82,18 ±3,56**	76,71 ±1,73	**83,15 ±3,69**	69 ±3,84	**82,18 ±3,56**	68,36 ±4,11	**81,87 ±3,7**
		125	125	59,64 ±7,40	**77,91 ±3,64**	69,92 ±2,98	**79,91 ±4,39**	59,64 ±7,4	**77,91 ±3,64**	59,54 ±6,43	**77,79 ±3,94**
		42	42	46,55 ±7,75	**56,45 ±2,87**	**75,83 ±5,45**	71,5 ±9,72	46,55 ±7,75	**56,45 ±2,87**	46,82 ±7,18	**55,34 ±3,16**
TWITTER	CNN	310	343,6	67,55 ±0,97	**68,97 ±0,13**	51,71 ±5,02	**52,75 ±6,79**	67,55 ±0,97	**68,97 ±0,13**	**57,29 ±1,68**	56,6 ±0,7
	ENN	1150,4	1263,8	65,82 ±1,42	**69,14 ±0,76**	57,58 ±3,19	**64,89 ±1,79**	65,82 ±1,42	**69,14 ±0,76**	60,09 ±1,04	**63,08 ±1,17**
	RMHC	1218	1218	68,39 ±0,95	**68,78 ±1,41**	59,68 ±2,34	**65,35 ±3,05**	68,39 ±0,95	**68,78 ±1,41**	59,91 ±1,54	**63,95 ±1,41**
		870	870	68,41 ±0,70	**69,03 ±0,91**	57,40 ±3,00	**64,26 ±1,47**	68,41 ±0,70	**69,03 ±0,91**	58,78 ±1,25	**63,94 ±1,58**
		522	522	68,56 ±0,96	**69,68 ±0,85**	57,05 ±4,00	**63,00 ±2,32**	68,56 ±0,96	**69,68 ±0,85**	58,11 ±0,60	**62,07 ±0,67**
		174	174	68,88 ±0,19	**69,16 ±0,68**	58,74 ±5,32	**59,45 ±6,3**	68,88 ±0,19	**69,16 ±0,68**	57,61 ±1,42	**57,87 ±1,72**
	SIL	1219	1219	61,52 ±4,16	**68,18 ±1,49**	58,47 ±1,54	**64,26 ±1,16**	61,52 ±4,16	**68,18 ±1,49**	58,37 ±2,05	**64,48 ±0,94**
		871	871	66,78 ±1,82	**67,27 ±2,28**	55,73 ±5,03	**63,25 ±2,4**	66,78 ±1,82	**67,27 ±2,28**	57,75 ±1,17	**63,26 ±0,96**
		523	523	67,23 ±2,07	**67,79 ±0,70**	**57,9 ±4,59**	56,69 ±3,39	67,23 ±2,07	**67,79 ±0,7**	**58,18 ±1,83**	58 ±1,46
		175	175	66,44 ±5,11	**68,88 ±0,00**	**52,52 ±6,67**	47,45 ±0	66,44 ±5,11	**68,88 ±0**	56,08 ±0,80	**56,19 ±0**

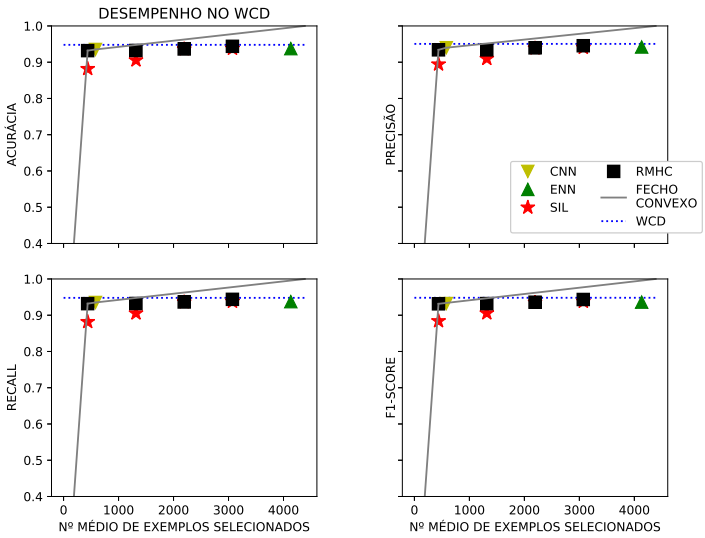
Tabela 5.10: Resultados de *accuracy*, *precision*, *recall* e *f1-score* para as representações BOW e WCD nos *datasets* REUTERS, BBC SPORT e TWITTER.

Na comparação multi-critério, as Figuras 5.6, 5.7 e 5.8 ilustram o desempenho no eixo y e o número de exemplos selecionados no eixo x. O fecho convexo destaca o algoritmo RMHC que se mantém estável apesar da redução de até 90% dos seus exemplos nas duas representações em todos os *datasets* utilizados. Outro resultado interessante é para o algoritmo SIL que ganha estabilidade nas medidas de desempenho com a representação WCD no *dataset* REUTERS.

Nas Figuras 5.8 é notado que os métodos de seleção apresentam estabilidade mesmo reduzindo até 90% com o SIL e RMHC. Isto é uma vantagem para o SIL que é mais rápido que os outros métodos apresentados e ficando mais claro na Figura 5.11. Isso sugere que pode ser diminuindo ainda mais a quantidade de instâncias selecionadas pelos métodos.



(a) Representação BOW.



(b) Representação WCD.

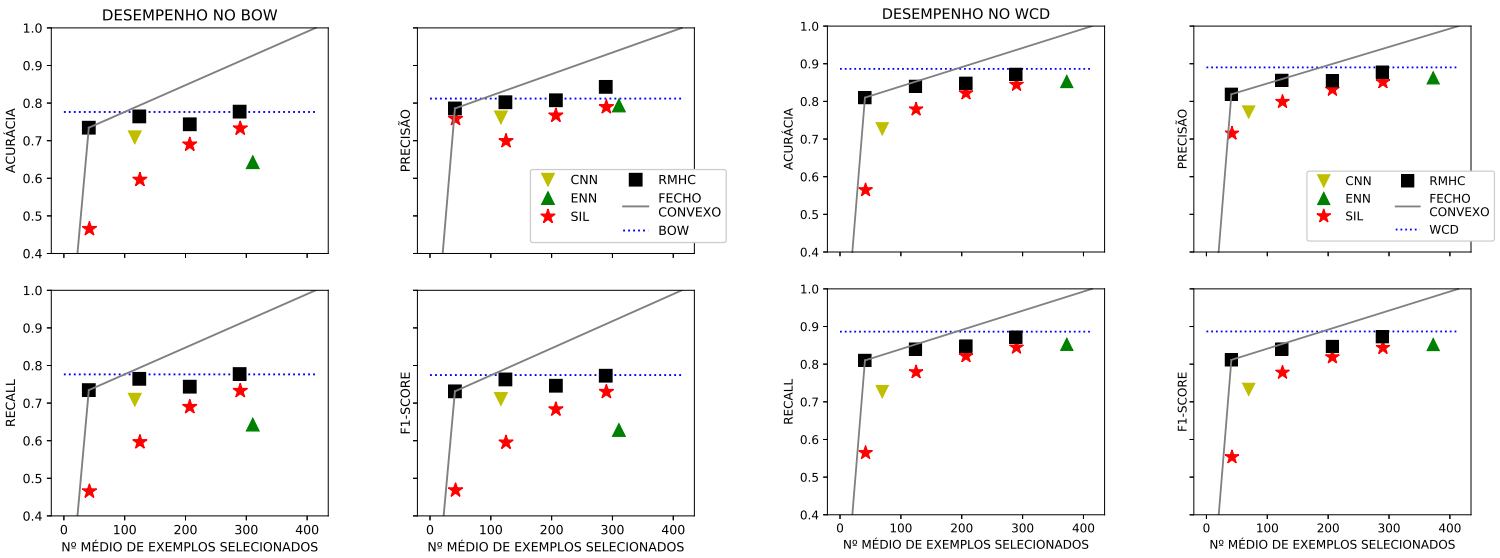
Figura 5.6: Desempenho vs número médio de exemplos selecionados no *dataset* REUTERS.

Nas Figuras 5.9, 5.10, 5.11 também é possível notar o ganho do WCD sobre o BOW. Além disso, os resultados mostram escalas diferentes para o tempo em cada representação dos dados de cada *dataset*. Porém, os resultados foram obtidos em computadores com processadores diferentes, mesmo assim possuem duas ordem de magnitude de diferença.

Na Figura 5.8a é mostrada que a Seleção de Instâncias com a representação BOW no *dataset* TWITTER melhora a *accuracy* de quase todos os métodos sobre a utilização do conjunto de treino completo.

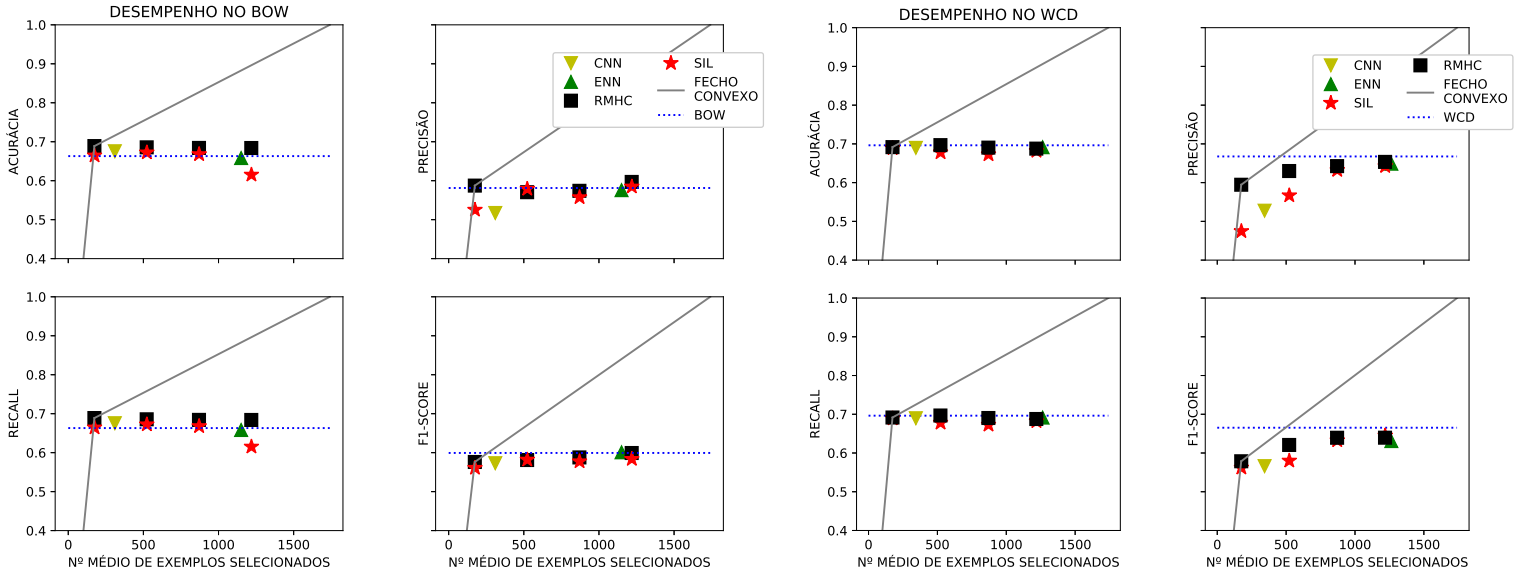
Há casos como na Figura 5.9a onde o RMHC com 90% de redução melhora a *accuracy* do método BOW com todos os exemplos, ou seja, nesse caso cumpre um dos aspectos de Seleção de Instâncias que é o de produzir um subconjunto ótimo para classificação, eliminando instâncias que diminuem a taxa de acerto.

O ENN produz uma vantagem na medida de classificação ao usar WCD do que BOW conforme ilustrada na Figura 5.10, saindo de 64,27% para 85,27%, a maior vantagem entre os outros métodos de Seleção de Instâncias.



(a) Representação BOW. (b) Representação WCD.

Figura 5.7: Desempenho vs número médio de exemplos selecionados no *dataset* BBC SPORT.

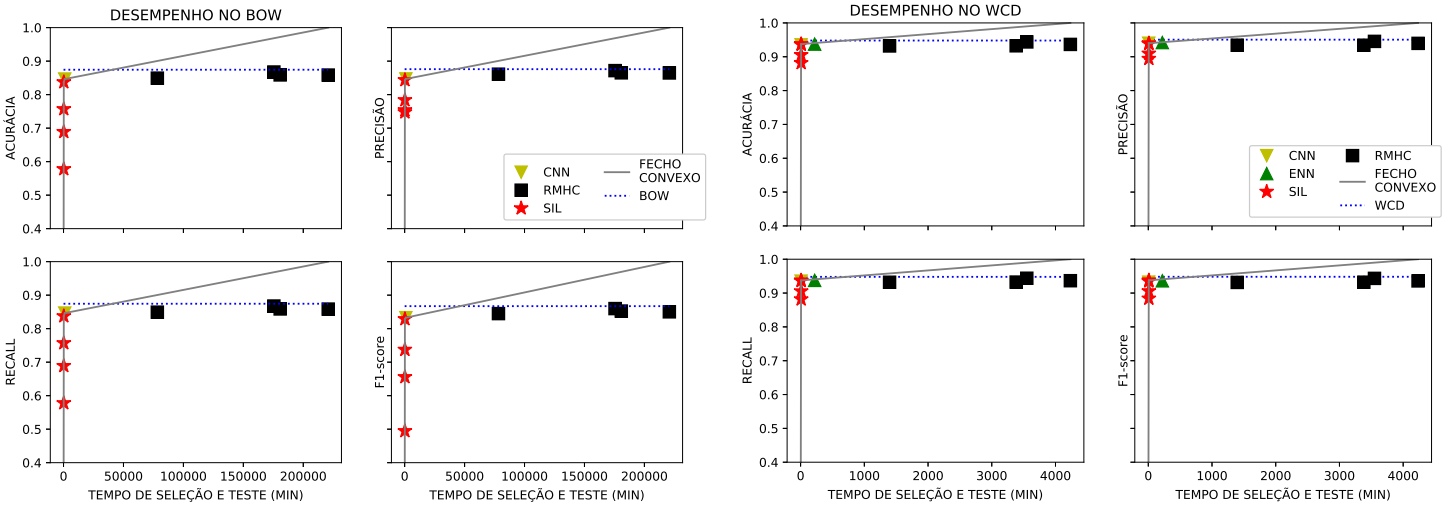


(a) Representação BOW. (b) Representação WCD.

Figura 5.8: Desempenho vs número médio de exemplos selecionados no *dataset* TWITTER.

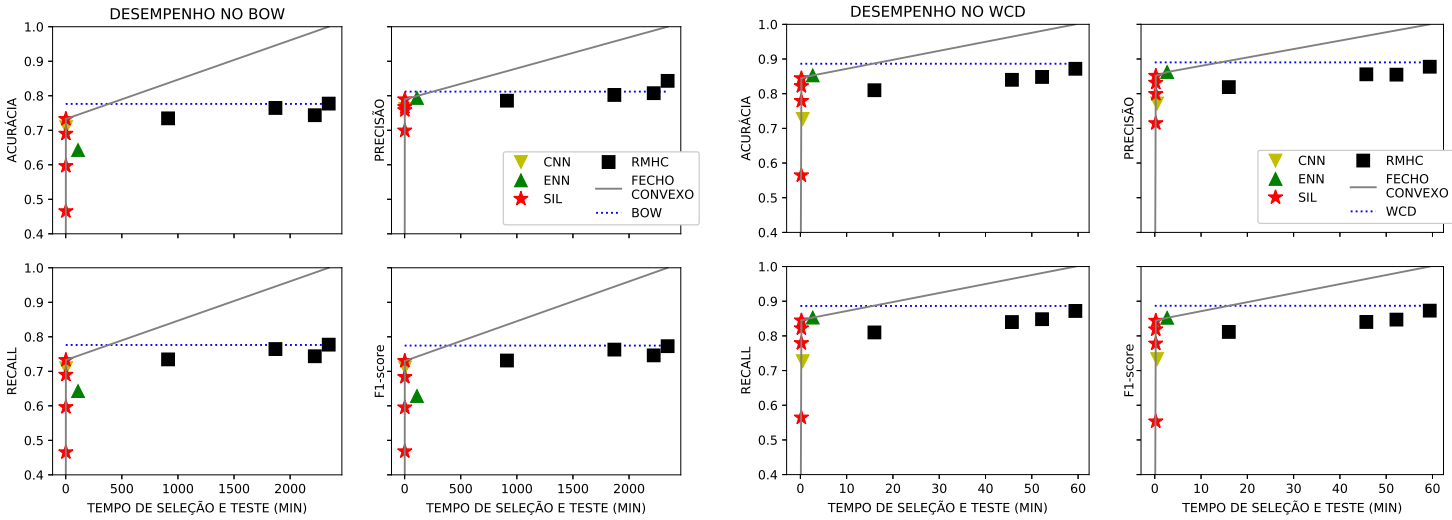
5.4.2 Seleção de Instâncias Somente com *Word Embeddings* vs com Aprendizado de Métrica e *Word Embeddings*

A Tabela 5.11 está organizada para melhor visualização da diferença entre as representações WCD e S-WCD, ambas usando Seleção de Instâncias. A tabela apresenta a coluna #Selecionados que contém a quantidade de exemplos selecionados pelos algoritmos de Seleção de Instâncias nas duas repre-



(a) Representação BOW. (b) Representação WCD.

Figura 5.9: Desempenho vs tempo de seleção mais tempo de teste no *dataset* REUTERS.



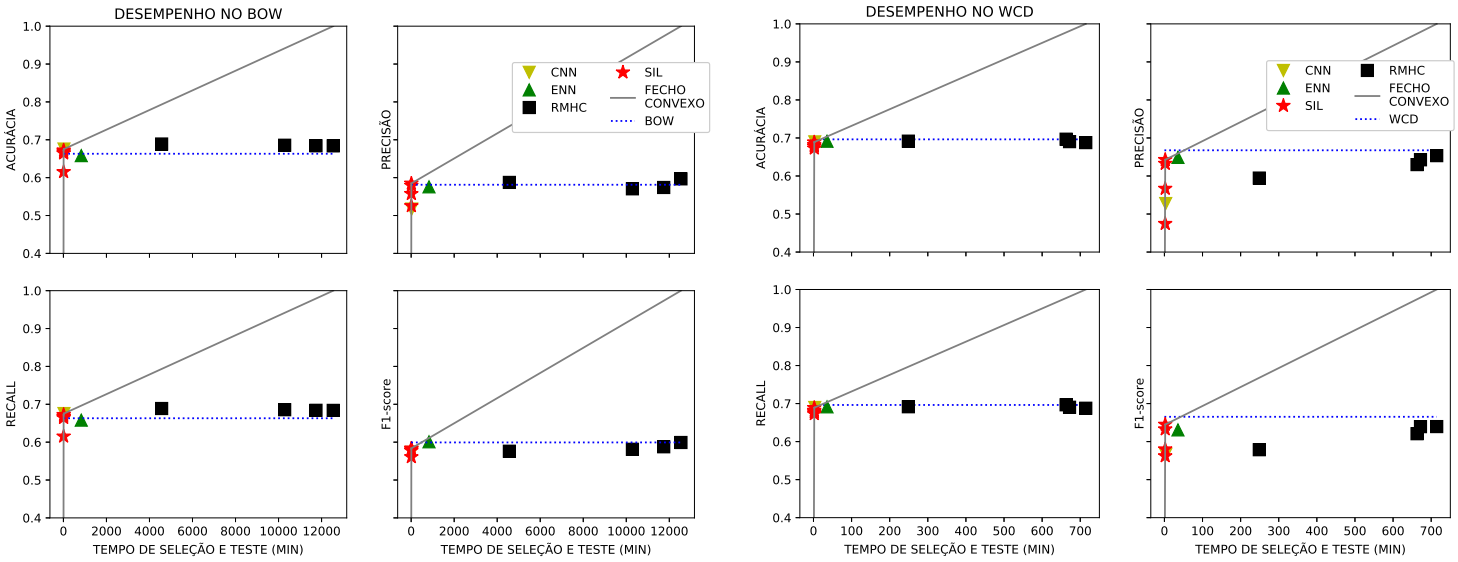
(a) Representação BOW. (b) Representação WCD.

Figura 5.10: Desempenho vs tempo de seleção mais tempo de teste no *dataset* BBC SPORT.

sentações. Nas demais colunas estão as medidas de desempenho: *accuracy*, *precision*, *recall*, *f1-score* e ao seu lado o desvio padrão para os *datasets* BBC SPORT e TWITTER.

#Selecionados: a redução de instâncias de uma representação WCD para S-WCD é equivalente para todos os métodos de seleção em todos os *datasets*, exceto o CNN. No CNN há uma redução de 46% dos exemplos em comparação com a representação WCD. Há também a redução 10% no *dataset* REUTERS utilizando S-WCD com CNN.

***accuracy*, *precision*, *recall*, *f1-score*:** os resultados do S-WCD com Sele-



(a) Representação BOW.

(b) Representação WCD.

Figura 5.11: Desempenho vs tempo de seleção mais tempo de teste no *dataset* TWITTER.

ção de Instâncias possuem consideráveis performances sobre o WCD com Seleção de Instâncias. Em todos os resultados dos *datasets* REUTERS e BBC SPORT apresentam melhorias em todas as medidas de desempenho. A maioria dos resultados de *accuracy* do S-WCD são melhores do que WCD no *dataset* TWITTER como destacados em negrito na tabela. As melhorias variam menos que os encontrados na Tabela 5.10, mas ainda assim há variações como a do método SIL que chegam a 14,5%.

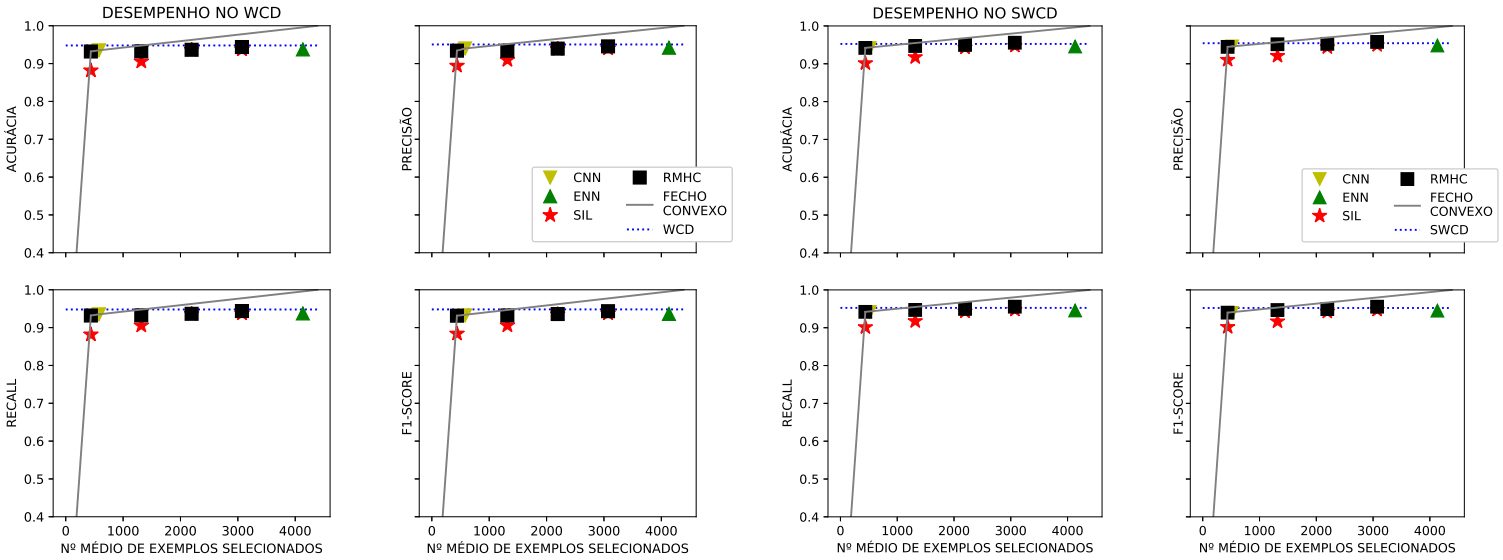
Para o *dataset* REUTERS, na Figura 5.12 está apresentado dois gráficos que são semelhantes em todos as medidas de desempenho e o ganho é mais perceptível ao olhar os resultados na Tabela 5.11.

Na Figura 5.13 está sugerido que o método do S-WCD com Seleção de Instâncias aumenta os valores para as medidas de classificação no *dataset* BBC SPORT. Observando a figura, o RMHC com a representação WCD possui uma variação de 6,18%, enquanto com o S-WCD isto diminui para 1%, ou seja uma estabilidade maior nos desempenhos do RMHC.

Na Figura 5.14 o comportamento dos métodos WCD e S-WCD é semelhante ao que acontece com o *dataset* REUTERS. Em ambas as figuras, o melhor método seguindo a linha do fecho convexo é o RMHC.

		# Selecionados		*accuracy*		*precision*		*recall*		*f1-score*	
	Mét.	WCD	S-WCD	WCD	S-WCD	WCD	S-WCD	WCD	S-WCD	WCD	S-WCD
Reuters	CNN	580	518	93,51	**94,15**	94,00	**94,59**	93,51	**94,15**	93,22	**93,92**
	ENN	4133	4133	93,74	**94,56**	94,19	**94,83**	93,74	**94,56**	93,61	**94,52**
	RMHC	3071	3071	94,38	**95,57**	94,55	**95,78**	94,38	**95,57**	94,34	**95,57**
		2194	2194	93,65	**94,97**	93,96	**95,27**	93,65	**94,97**	93,59	**94,93**
		1316	1316	93,24	**94,7**	93,4	**95,16**	93,24	**94,7**	93,21	**94,68**
		438	438	93,19	**94,2**	93,41	**94,47**	93,19	**94,2**	93,14	**93,99**
	SIL	3072	3072	93,74	**94,75**	94,00	**94,93**	93,74	**94,75**	93,76	**94,74**
		2194	2194	93,6	**94,29**	93,84	**94,37**	93,60	**94,29**	93,49	**94,22**
		1317	1317	90,54	**91,73**	90,96	**92,08**	90,54	**91,73**	90,57	**91,66**
		439	439	88,17	**90,13**	89,42	**91,02**	88,17	**90,13**	88,39	**90,18**
BBC Sport	CNN	69,4	37,4	72,73 ±6,33	**78,36 ±3,61**	77,19 ±3,59	**83,39 ±2,26**	72,73 ±6,33	**78,36 ±3,61**	73,32 ±5,92	**78,5 ±3,64**
	ENN	372,6	372,6	85,27 ±1,88	**87,82 ±0,78**	86,25 ±2,14	**89 ±1,07**	85,27 ±1,88	**87,82 ±0,78**	85,22 ±1,81	**87,44 ±0,77**
	RMHC	289	289	87,18 ±0,78	**92,55 ±1,48**	87,78 ±0,69	**92,83 ±1,44**	87,18 ±0,78	**92,55 ±1,48**	87,28 ±0,74	**92,55 ±1,46**
		207	207	84,82 ±1,24	**92,36 ±1,53**	85,49 ±1,17	**92,79 ±1,23**	84,82 ±1,24	**92,36 ±1,53**	84,7 ±1,17	**92,39 ±1,46**
		124	124	84,00 ±1,64	**92,36 ±1,56**	85,59 ±0,69	**92,78 ±1,26**	84 ±1,64	**92,36 ±1,56**	84,01 ±1,39	**92,4 ±1,51**
		41	41	81,00 ±1,05	**91,55 ±1,88**	81,9 ±1,18	**92,07 ±1,79**	81 ±1,05	**91,55 ±1,88**	81,14 ±0,99	**91,62 ±1,83**
	SIL	290	290	84,45 ±3,03	**91 ±0,88**	85,15 ±3,12	**91,78 ±0,87**	84,45 ±3,03	**91 ±0,88**	84,37 ±3,00	**91,04 ±0,86**
		207	207	82,18 ±3,56	**89,82 ±1,76**	83,15 ±3,69	**90,75 ±1,36**	82,18 ±3,56	**89,82 ±1,76**	81,87 ±3,7	**89,88 ±1,71**
		125	125	77,91 ±3,64	**89,45 ±2,56**	79,91 ±4,39	**90,32 ±2,01**	77,91 ±3,64	**89,45 ±2,56**	77,79 ±3,94	**89,54 ±2,48**
		42	42	56,45 ±2,87	**71 ±9,49**	71,5 ±9,72	**78,41 ±8,38**	56,45 ±2,87	**71 ±9,49**	55,34 ±3,16	**69,97 ±10,72**
Twitter	CNN	343,6	351,6	68,97 ±0,13	**69,33 ±0,66**	52,75 ±6,79	**55,81 ±5,14**	68,97 ±0,13	**69,33 ±0,66**	56,6 ±0,7	**59,31 ±2,18**
	ENN	1263,8	1263,8	**69,14 ±0,76**	68,99 ±1,32	64,89 ±1,79	**66,86 ±3,6**	**69,14 ±0,76**	68,99 ±1,32	**63,08 ±1,17**	62,6 ±0,53
	RMHC	1218	1218	68,78 ±1,41	**69,74 ±1,27**	65,35 ±3,05	**65,77 ±1,62**	**68,78 ±1,41**	69,74 ±1,27	63,95 ±1,41	**65,31 ±0,78**
		870	870	69,03 ±0,91	**69,27 ±1,33**	64,26 ±1,47	**64,82 ±2,58**	69,03 ±0,91	**69,27 ±1,33**	**63,94 ±1,58**	63,44 ±1,32
		522	522	69,68 ±0,85	**69,72 ±0,81**	63,00 ±2,32	**66,3 ±2,85**	69,68 ±0,85	**69,72 ±0,81**	62,07 ±0,67	**63,05 ±1,47**
		174	174	**69,16 ±0,68**	68,3 ±0,5	59,45 ±6,3	**61,77 ±1,35**	**69,16 ±0,68**	68,3 ±0,5	57,87 ±1,72	**61,25 ±1,17**
	SIL	1219	1219	68,18 ±1,49	**69,4 ±1,51**	64,26 ±1,16	**65,65 ±1,52**	68,18 ±1,49	**69,4 ±1,51**	64,48 ±0,94	**65,35 ±1,09**
		871	871	67,27 ±2,28	**67,94 ±2**	**63,25 ±2,4**	63,1 ±1,38	67,27 ±2,28	**67,94 ±2**	**63,26 ±0,96**	63,19 ±0,84
		523	523	67,79 ±0,70	**68,07 ±0,92**	56,69 ±3,39	**56,99 ±3,11**	67,79 ±0,7	**68,07 ±0,92**	58 ±1,46	**58,02 ±1,48**
		175	175	**68,88 ±0,00**	**68,88 ±0**	**47,45 ±0**	**47,45 ±0**	**68,88 ±0**	**68,88 ±0**	**56,19 ±0**	**56,19 ±0**

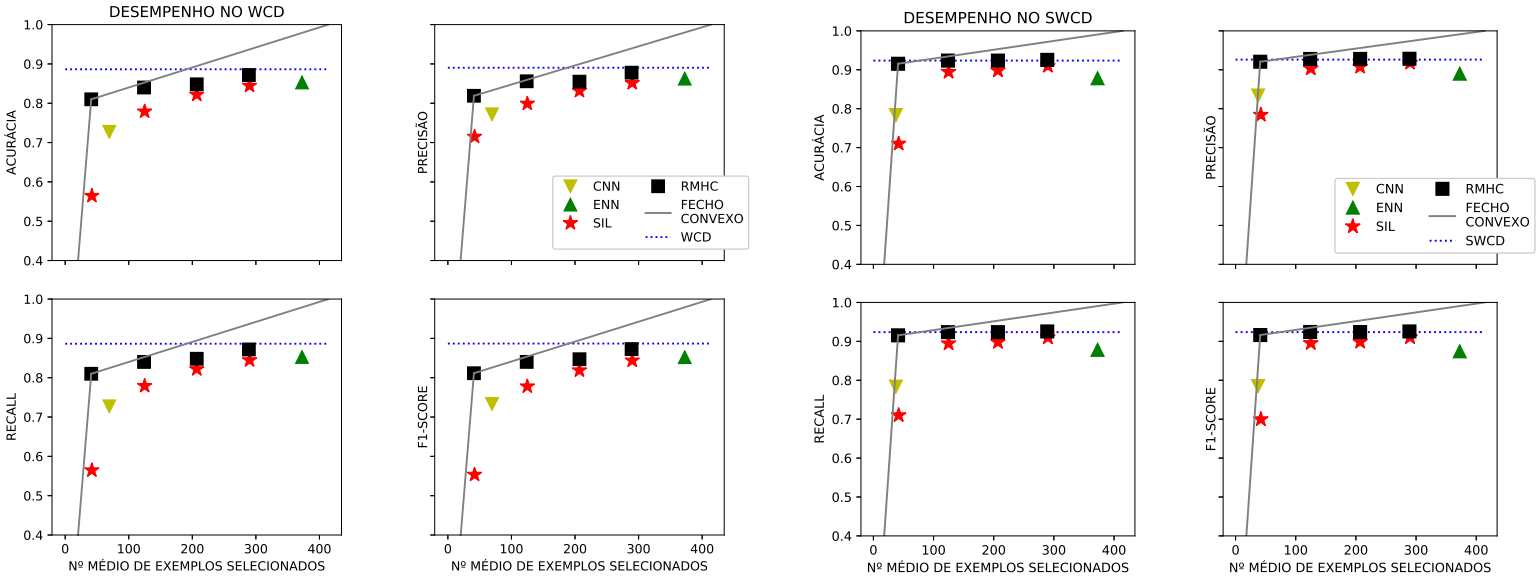
Tabela 5.11: Resultados de *accuracy*, *precision*, *recall* e *f1-score* para as representações WCD e S-WCD nos *datasets* Reuters, BBC Sport e Twitter.



(a) Representação WCD.

(b) Representação S-WCD.

Figura 5.12: Desempenho vs número médio de exemplos selecionados no *dataset* REUTERS.



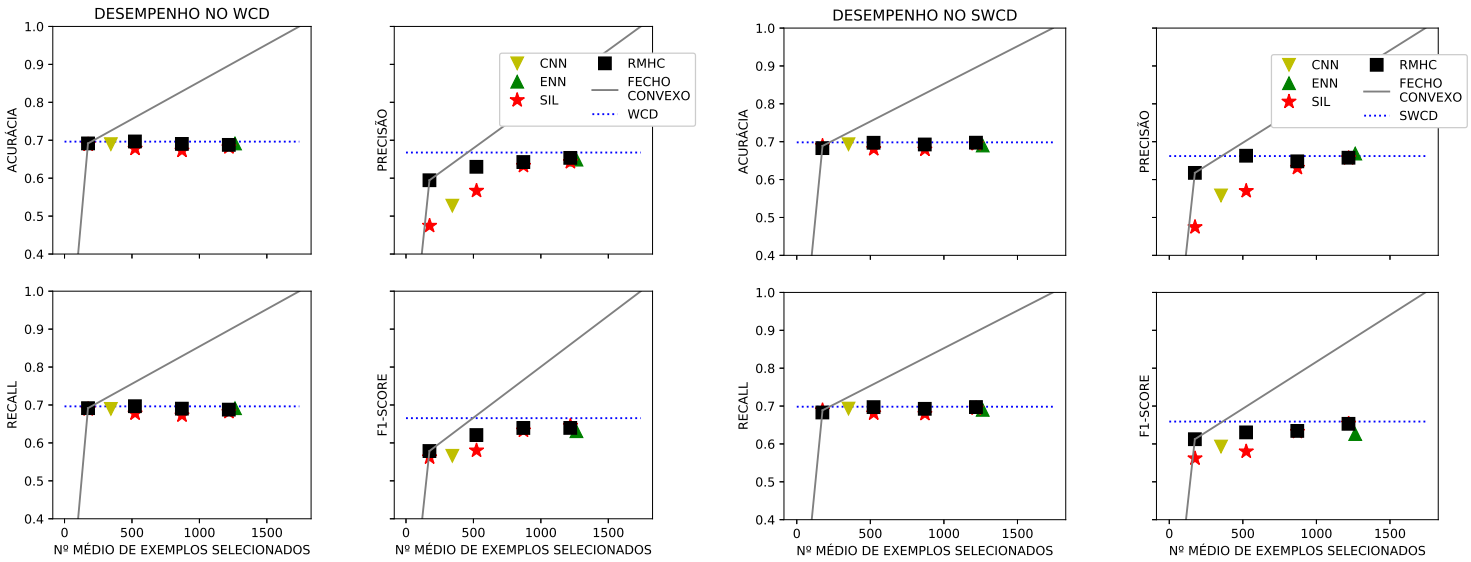
(a) Representação WCD.

(b) Representação S-WCD.

Figura 5.13: Desempenho vs número médio de exemplos selecionados no *dataset* BBC SPORT.

5.4.3 Desempenho em *Accuracy* com Seleção de Instâncias

Na Tabela 5.12 é mostrado a quantidade de instâncias selecionadas nas colunas de acordo com o método de Seleção de Instâncias. RMHC e SIL possuem o seu meta-parâmetro de porcentagem de redução utilizada e está



(a) Representação WCD.

(b) Representação S-WCD.

Figura 5.14: Desempenho vs número médio de exemplos selecionados no *dataset* TWITTER.

descrito junto ao seu nome. As linhas desta tabela exibem a quantidade de exemplos selecionados na representação de dados para cada *dataset* utilizado.

A Tabela 5.13 apresenta o desempenho de *accuracy* dos métodos de Seleção de Instâncias para todos os *datasets* utilizados. As colunas são os métodos de seleção utilizados, em especial o SIL e o RMHC possuem suas variações de acordo a porcentagem escolhida, para cada proporção usada tem uma coluna para seus valores. As linhas da tabela apresentam os resultados das representações utilizadas dos *datasets* da coluna logo à esquerda. Nessa tabela percebe-se que quando se trata de melhor desempenho em *accuracy* o método ENN e o RMHC são os melhores dentre os executados nessa avaliação experimental.

O ENN não produz uma redução tão considerável quanto os outros métodos. O RMHC produz uma ótima redução e mantêm bem os seus valores de *accuracy* ao passo que vai reduzindo o número de exemplos a serem selecionados. Há situações onde o RMHC com 90% de redução melhora a *accuracy* da representação BOW com todos os exemplos. Embora, o RMHC seja um ótimo em estabilizar e melhorar seus resultados de *accuracy*, o seu tempo de seleção ultrapassa facilmente o tempo de classificação e seleção dos outros métodos em todos os experimentos realizados nessa avaliação.

Observando a Tabela 5.13 todos os métodos de Seleção de Instâncias melhoram seu desempenho *accuracy* ao melhorar a representação e/ou usando aprendizado de métrica. Dos resultados do *dataset* BBC SPORT destacam-se o SIL com 90% de redução utilizando o BOW obteve 46,55% e ao utilizar o S-WMD o aumento foi de 83,78% atingindo o valor de 85,55% de *accuracy*. A representação S-WMD é tão efetiva que usando o SIL com redução de 30% no BOW não chega à *accuracy* do SIL com 90% de redução em todos os *datasets*. Com redução de 30% na representação WCD ocorre algo semelhante nos *datasets* BBC SPORT e TWITTER. Esses comportamentos que mostram a

qualidade do S-WMD sobre o BOW, e o WCD sobre BOW também acontecem com o RMHC ao analisar a sua eficiência nesses *datasets*.

		Número de exemplos selecionados									
		CNN	ENN	RMHC 30%	RMHC 50%	RMHC 70%	RMHC 90%	SIL 30%	SIL 50%	SIL 70%	SIL 90%
Reuters	BOW	834	-	3071	2194	1316	438	3072	2194	1317	439
	WCD	580	4133								
	S-WCD	518									
	S-WMD	580									
BBC Sport	BOW	116,6	310,6	289	207	124	41	290	207	125	42
	WCD	69,4	372,6								
	S-WCD	37,4									
	S-WMD	69,4									
Twitter	BOW	310	1150,4	1218	870	522	174	1219	871	523	175
	WCD	343,6	1263,8								
	S-WCD	351,6									
	S-WMD	343,6									

Tabela 5.12: Número de exemplos selecionados pelo método Seleção de Instâncias e pela representação para cada *dataset* avaliado.

		CNN	ENN	SIL				RMHC			
				30%	50%	70%	90%	30%	50%	70%	90%
Reuters	BOW	84,65	–	83,74	75,70	68,89	57,79	**85,88**	85,84	86,75	84,92
	WCD	93,51	93,74	93,74	93,6	90,54	88,17	**94,38**	93,65	93,24	93,19
	S-WCD	94,15	94,56	94,75	94,29	91,73	90,13	**95,57**	94,97	94,7	94,2
	S-WMD	95,02	**96,94**	95,75	95,07	94,66	93,01	96,85	96,76	95,75	95,71
BBC	BOW	$70,91 \pm 4,65$	$64,27 \pm 2,55$	$73,27 \pm 1,66$	$69,00 \pm 3,84$	$59,64 \pm 7,40$	$46,55 \pm 7,75$	**77,73 ±6,66**	$74,36 \pm 5,17$	$76,46 \pm 2,63$	$73,46 \pm 3,48$
	WCD	$72,73 \pm 6,33$	$85,27 \pm 1,88$	$84,45 \pm 3,03$	$82,18 \pm 3,56$	$77,91 \pm 3,64$	$56,45 \pm 2,87$	**87,18 ±0,78**	$84,82 \pm 1,24$	$84,0 \pm 1,64$	$81,0 \pm 1,05$
	S-WCD	$78,36 \pm 3,61$	$87,82 \pm 0,78$	$91,0 \pm 0,88$	$89,82 \pm 1,76$	$89,45 \pm 2,56$	$71,0 \pm 9,49$	**92,55 ±1,48**	$92,36 \pm 1,53$	$92,36 \pm 1,56$	$91,55 \pm 1,88$
	S-WMD	$85,36 \pm 3,94$	$96,18 \pm 1,24$	$95,18 \pm 1,67$	$94,27 \pm 1,17$	$94,18 \pm 1,45$	$85,55 \pm 6,71$	**96,55 ±0,46**	$96,45 \pm 0,67$	$96,09 \pm 1,3$	$94,73 \pm 1,9$
Twitter	BOW	$67,55 \pm 0,97$	$65,81 \pm 1,42$	$61,52 \pm 4,16$	$66,78 \pm 1,82$	$67,23 \pm 2,07$	$66,44 \pm 5,11$	$68,39 \pm 0,95$	$68,41 \pm 0,70$	$68,56 \pm 0,96$	**68,88 ±0,19**
	WCD	$68,97 \pm 0,13$	**69,14 ±0,76**	$68,18 \pm 1,49$	$67,27 \pm 2,28$	$67,79 \pm, 0,7$	$68,88 \pm 0,0$	$68,78 \pm 1,41$	$69,03 \pm 0,91$	$69,68 \pm 0,85$	$69,16 \pm 0,68$
	S-WCD	$69,33 \pm 0,66$	$68,99 \pm 1,32$	$69,4 \pm 1,51$	$67,94 \pm 2,0$	$68,07 \pm 0,92$	$68,88 \pm 0,0$	**69,74 ±1,27**	$69,27 \pm 1,33$	$69,72 \pm 0,81$	$68,3 \pm 0,5$
	S-WMD	$59,91 \pm 2,33$	$69,7 \pm 1,14$	$70,32 \pm 1,18$	$70,0 \pm 0,48$	$68,65 \pm 0,43$	$68,88 \pm 0,0$	**70,75 ±0,67**	$70,47 \pm 0,85$	$69,98 \pm 0,94$	$68,45 \pm 0,24$

Tabela 5.13: Desempenho de *accuracy* dos métodos de Seleção de Instâncias para todos os *datasets* utilizados

CAPÍTULO

6

Conclusões

Esta dissertação de mestrado responde às seguintes questões:

1. *word embedding* pode melhorar métodos de Seleção de Instâncias?

   Sim, como apresentado nos resultados do Capítulo 5, a representação de dados WCD supera o BOW em todos as medidas de classificação e também em relação ao tempo gasto para computar e classificar exemplos selecionados para todos os *datasets* e todos os métodos de Seleção de Instâncias dessa avaliação experimental.

2. Aprendizado de Métrica pode melhorar a combinação de *word embedding* + Seleção de Instâncias?

   Sim, de acordo com os resultados exibidos nas Tabelas 5.11, 5.13 há ganhos significativos em todas as medidas de desempenho de classificação com os *datasets* REUTERS e BBC SPORT. Para o TWITTER na maioria das métricas de desempenho em classificação o S-WCD é melhor que o WCD. No entanto, a computação do WCD é rápida, a do S-WCD depende do Aprendizado de Métrica.

3. Qual é a melhor combinação de método de Seleção de Instâncias e de representação de dados?

   Para melhor desempenho em redução e medida de avaliação de classificação é o método do RMHC. Para uma resposta rápida e com bom desempenho em classificação podem ser atribuídos ao CNN e SIL. Com o SIL uma melhor representação dos dados pode auxiliar muito em seu desempenho e isso é notado nos resultados da Tabela 5.13.

Os *word embeddings* utilizados são do WORD2VEC e possuem dimensão igual à 300, se comparados com as dimensões em BOW, há uma enorme economia de memória nessa troca, proporcionada pelo *word embeddings*. Para o WCD, S-WCD, WMD, S-WMD é fundamental que hajam *word embeddings* com qualidade como os proporcionados pelo WORD2VEC.

6.1 Contribuições

As contribuições deste trabalho estão enumeradas a seguir:

1. Os *word embeddings* auxiliam na construção de um Espaço de Métrica melhor para aumentar a efetividade dos algoritmos de Seleção de Instâncias;

2. O algoritmo RMHC é efetivo na redução de instâncias mantendo o desempenho de classificação.

3. A representação do WCD possui bons resultados e junto com os métodos de Seleção de Instâncias exibem um bom desempenho.

6.2 Trabalhos Futuros

Usar outra métricas de distâncias como a similaridade de cosseno, ou outras relacionadas a mineração de textos e *big data*. Obter mais resultados em outros *datasets*, para uma análise mais abrangente no comportamento dos métodos de Seleção de Instâncias. Realizar experimentos com o *dataset* do TWITTER para os algoritmos SIL e RMHC mais extremos, reduzindo por mais de 90% o conjunto de exemplos de treinamento. Usar outras transformações de espaço para ganho de desempenho como o a distorção do espaço proporcionada pelo t-SNE (Maaten e Hinton, 2008) usando os *word embeddings*.

Referências Bibliográficas

Aly, M. (2005). Survey on multiclass classification methods. *Neural Netw*, 19:1–9. Citado na página 33.

Campos, E. e Matsubara, E. (2014). An experimental evaluation of sentiment analysis on financial news using prior polarity words. Em *Advances in Artificial Intelligence–IBERAMIA 2014*, pgs. 218–228. Springer. Citado na página 27.

Chomsky, N. (1957). Syntactic structures. Citado na página 6.

Chou, C.-H., Kuo, B.-H., e Chang, F. (2006). The generalized condensed nearest neighbor rule as a data reduction method. Em *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pgs. 556–559. IEEE. Citado na página 37.

Collobert, R. e Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. Em *Proceedings of the 25th international conference on Machine learning*, pgs. 160–167. ACM. Citado na página 47.

Cortes, C. e Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297. Citado na página 13.

Cover, T. e Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27. Citado nas páginas 7 e 36.

Cuturi, M. e Doucet, A. (2014). Fast computation of wasserstein barycenters. Em *International Conference on Machine Learning*, pgs. 685–693. Citado nas páginas 50 e 53.

da Silva, I. N., Spatti, D. H., e Flauzino, R. A. (2010). Redes neurais artificiais para engenharia e ciências aplicadas curso prático. *São Paulo: Artliber*. Citado nas páginas xv, xvii, 11, 12, e 13.

da Silva Brito, G. (2000). Lingüistas e computadores: que relação é essa? *Working Papers em Linguística*, 4(1):7–23. Citado na página 6.

Davis, J. V., Kulis, B., Jain, P., Sra, S., e Dhillon, I. S. (2007). Information-theoretic metric learning. Em *Proceedings of the 24th international conference on Machine learning*, pgs. 209–216. ACM. Citado na página 11.

de Souza Rodrigues, L. (2016). Flexrank: Um rankeador lexicográfico rápido. Dissertação de Mestrado. Citado na página 27.

Deng, L., Yu, D., et al. (2014). Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387. Citado na página 25.

Dey, D., Solorio, T., y Gomez, M. M., e Escalante, H. J. (2011). Instance selection in text classification using the silhouette coefficient measure. Em *Mexican International Conference on Artificial Intelligence*, pgs. 357–369. Springer. Citado na página 42.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211. Citado na página 2.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874. Citado nas páginas 31 e 32.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, 3:1289–1305. Citado na página 27.

Friedman, J., Hastie, T., e Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin. Citado na página 14.

Friedman, J. H. (1997). On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77. Citado na página 7.

Garain, U. (2008). Prototype reduction using an artificial immune model. *Pattern analysis and applications*, 11(3-4):353–363. Citado na página 60.

Gardner, J. R., Kusner, M. J., Xu, Z. E., Weinberger, K. Q., e Cunningham, J. P. (2014). Bayesian optimization with inequality constraints. Em *International Conference on Machine Learning*, pgs. 937–945. Citado na página 51.

Goldberg, Y. e Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*. Citado na página 30.

Goldberger, J., Hinton, G. E., Roweis, S. T., e Salakhutdinov, R. R. (2005). Neighbourhood components analysis. Em *Advances in Neural Information Processing Systems*, pgs. 513–520. Citado nas páginas 9 e 10.

Goodfellow, I., Bengio, Y., Courville, A., e Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge. Citado na página 2.

Gou, J., Du, L., Zhang, Y., Xiong, T., et al. (2012). A new distance-weighted k-nearest neighbor classifier. *Journal of Information & Computational Science*, 9(6):1429–1436. Citado na página 8.

Grauman, K. e Darrell, T. (2004). Fast contour matching using approximate earth mover's distance. Em *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pgs. I–I. IEEE. Citado na página 45.

Graves, A., Mohamed, A.-r., e Hinton, G. (2013). Speech recognition with deep recurrent neural networks. Em *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pgs. 6645–6649. IEEE. Citado na página 26.

Hart, P. (1968). The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516. Citado na página 36.

Haykin, S. S. (2000). Redes neurais artificiais: princípio e prática. *2ª Edição, Bookman, São Paulo, Brasil*. Citado na página 13.

Hinton, G. E. e Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507. Citado na página 1.

Holland, S. M. (2008). Principal components analysis (pca). *Department of Geology, University of Georgia, Athens, GA*, pgs. 30602–2501. Citado na página 31.

Huang, G., Guo, C., Kusner, M. J., Sun, Y., Sha, F., e Weinberger, K. Q. (2016). Supervised word mover's distance. Em *Advances in Neural Information Processing Systems*, pgs. 4862–4870. Citado nas páginas xv, 2, 3, 4, 49, 50, 58, e 60.

Jain, A. K. e Dubes, R. C. (1988). Algorithms for clustering data. Citado na página 39.

John Walker, S. (2014). Big data: A revolution that will transform how we live, work, and think. Citado na página 1.

Kaufman, L. e Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons. Citado na página 39.

Krizhevsky, A., Sutskever, I., e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Em *Advances in Neural Information Processing Systems*, pgs. 1097–1105. Citado na página 1.

Kuramochi, M. e Karypis, G. (2005). Gene classification using expression profiles: a feasibility study. *International Journal on Artificial Intelligence Tools*, 14(04):641–660. Citado na página 7.

Kusner, M., Sun, Y., Kolkin, N., e Weinberger, K. (2015). From word embeddings to document distances. Em *International Conference on Machine Learning*, pgs. 957–966. Citado nas páginas xv, 2, 47, 48, 51, 58, e 60.

Lai, S., Xu, L., Liu, K., e Zhao, J. (2015). Recurrent convolutional neural networks for text classification. Em *Association for the Advancement of Artificial Intelligence*, volume 333, pgs. 2267–2273. Citado na página 26.

Landauer, T. K., Foltz, P. W., e Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284. Citado na página 30.

Le, Q. V. e Mikolov, T. (2014). Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*. Citado nas páginas 27 e 30.

LeCun, Y., Bengio, Y., e Hinton, G. (2015a). Deep learning. *Nature*, 521(7553):436–444. Citado na página 13.

LeCun, Y., Bengio, Y., e Hinton, G. (2015b). Deep learning. *Nature*, 521(7553):436–444. Citado na página 26.

LeCun, Y., Bottou, L., Bengio, Y., e Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. Citado na página 19.

LeCun, Y., Jackel, L., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U., Sackinger, E., Simard, P., et al. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276. Citado na página 19.

Liddy, E. (2001). In encyclopedia of library and information science, marcel decker. *Inc.- Natural Language Processing*. Citado nas páginas 5 e 6.

Lima, E. L. (1983). *Espaços métricos*, volume 4. Instituto de Matemática Pura e Aplicada, CNPq Rio de Janeiro. Citado nas páginas 8 e 9.

Ma, C.-M., Yang, W.-S., e Cheng, B.-W. (2014). How the parameters of k-nearest neighbor algorithm impact on the best classification accuracy: In case of parkinson dataset. *Journal of Applied Sciences*, 14:171–176. Citado na página 8.

Maaten, L. v. d. e Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605. Citado nas páginas 3 e 76.

Max, E. Z. G. (2016). Seleção de instâncias baseado em aprendizado de métricas para k vizinhos mais próximos. Master's thesis, FACOM-UFMS, Campo Grande, MS. Dissertação de Mestrado, FACOM-UFMS. http://posgraduacao.ufms.br/portal/trabalho-arquivos/download/3019. Citado nas páginas 3 e 39.

Mikolov, T., Chen, K., Corrado, G., e Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. Citado nas páginas 2, 28, e 30.

Mikolov, T., Le, Q. V., e Sutskever, I. (2013b). Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*. Citado na página 29.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., e Dean, J. (2013c). Distributed representations of words and phrases and their compositionality. Em *Advances in Neural Information Processing Systems*, pgs. 3111–3119. Citado nas páginas 28, 29, e 30.

Mikolov, T., Yih, W.-t., e Zweig, G. (2013d). Linguistic regularities in continuous space word representations. Em *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pgs. 746–751. Citado nas páginas xv e 30.

Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45:37. Citado na página 7.

Mnih, A. e Hinton, G. E. (2009). A scalable hierarchical distributed language model. Em *Advances in Neural Information Processing Systems*, pgs. 1081–1088. Citado na página 47.

Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., e Kittler, J. (2010). A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143. Citado nas páginas 35 e 36.

Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. Citado nas páginas 31 e 32.

Reinartz, T. (2002). A unifying view on instance selection. *Data Mining and Knowledge Discovery*, 6(2):191–210. Citado na página 35.

Ritter, G., Woodruff, H., Lowry, S., e Isenhour, T. (1975). An algorithm for a selective nearest neighbor decision rule (corresp.). *IEEE Transactions on Information Theory*, 21(6):665–669. Citado na página 37.

Rocha, R. (2007). Tecnologia adaptativa aplicada ao processamento computacional de língua natural. Em *Workshop de Tecnologias Adaptativas–WTA*, volume 2007. Citado na página 5.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386. Citado na página 11.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65. Citado na página 39.

Rubner, Y., Tomasi, C., e Guibas, L. J. (1998). A metric for distributions with applications to image databases. Em *Computer Vision, 1998. Sixth International Conference on*, pgs. 59–66. IEEE. Citado nas páginas xv, 43, e 44.

Rumelhart, D. E., Hinton, G. E., e Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document. Citado na página 13.

Rumelhart, D. E., Hinton, G. E., e Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533. Citado na página 2.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252. Citado na página 1.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., e Edwards, D. D. (2003). *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River. Citado nas páginas 7, 15, e 16.

Simonyan, K. e Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. Citado na página 26.

Skalak, D. B. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. Em *Machine Learning Proceedings 1994*, pgs. 293–301. Elsevier. Citado na página 37.

Sriram, B., Fuhry, D., Demir, E., Ferhatosmanoglu, H., e Demirbas, M. (2010). Short text classification in twitter to improve information filtering. Em *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pgs. 841–842, New York, NY, USA. ACM. Citado na página 27.

Theodoridis, S. e Koutroumbas, K. (2003). Feature selection. *Pattern recognition*, 5:261–322. Citado na página 7.

Tipping, M. E. e Bishop, C. M. (1999). Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482. Citado na página 31.

Trombley, S. (2014). *50 Pensadores que formaram o mundo moderno*. Leya. Citado na página 6.

Vidyarthi, A. e Mittal, N. (2016). Avnm: A voting based novel mathematical rule for image classification. *Computer methods and programs in biomedicine*, 137:195–201. Citado na página 8.

Vieira, R. e Lima, V. L. (2001). Lingüística computacional: princípios e aplicações. Em *Anais do XXI Congresso da SBC. I Jornada de Atualização em Inteligência Artificial*, volume 3, pgs. 47–86. sn. Citado na página 6.

Wallach, H. M. (2006). Topic modeling: Beyond bag-of-words. Em *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pgs. 977–984, New York, NY, USA. ACM. Citado na página 26.

Wan, X. (2007). A novel document similarity measure based on earth mover's distance. *Information Sciences*, 177(18):3718–3730. Citado na página 45.

Wan, X. e Peng, Y. (2005). The earth mover's distance as a semantic measure for document similarity. Em *Proceedings of the 14th ACM international conference on Information and knowledge management*, pgs. 301–302. ACM. Citado na página 45.

Weinberger, K. Q. e Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244. Citado nas páginas 8 e 11.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421. Citado na página 37.

Wilson, D. R. e Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine learning*, 38(3):257–286. Citado nas páginas 36 e 37.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37. Citado nas páginas 7 e 8.

Yang, Y. e Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. Em *International Conference on Machine Learning*, volume 97, pgs. 412–420. Citado na página 27.