



Universidade Federal
do Rio de Janeiro

Escola Politécnica

APRENDIZADO SEMI-SUPERVISIONADO PARA CLASSIFICAÇÃO DE POLARIDADE DE TWEETS

Breno Vieira Arosa

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Luiz Pereira Calôba

Rio de Janeiro

Julho de 2017

APRENDIZADO SEMI-SUPERVISIONADO PARA CLASSIFICAÇÃO DE POLARIDADE DE TWEETS

Breno Vieira Arosa

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO
DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA PO-
LITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU
DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

Autor:

Breno Vieira Arosa

Orientador:

Prof. Luiz Pereira Calôba, Dr. Ing.

Examinador:

Prof Frances Elizabeth Allen, D. Sc.

Examinador:

Prof. Alan Jay Perlis, D. E.

Rio de Janeiro

Julho de 2017

Declaração de Autoria e de Direitos

Eu, *Breno Vieira Arosa* CPF 131.187.117-95, autor da monografia *Aprendizado Semi-Supervisionado para Classificação de Polaridade de Tweets*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.
2. Excetua-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e ideias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.
3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.
4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.
5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.
6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.
7. Por ser verdade, firmo a presente declaração.

Breno Vieira Arosa

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Escola Politécnica - Departamento de Eletrônica e de Computação

Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária

Rio de Janeiro - RJ CEP 21949-900

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmear ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es).

DEDICATÓRIA

Opcional.

AGRADECIMENTO

Sempre haverá. Se não estiver inspirado, aqui está uma sugestão: dedico este trabalho ao povo brasileiro que contribuiu de forma significativa à minha formação e estada nesta Universidade. Este projeto é uma pequena forma de retribuir o investimento e confiança em mim depositados.

RESUMO

O presente trabalho apresenta a elaboração de um classificador de *deep learning* para análise de sentimento de mensagens de redes sociais sem a necessidade de bases de dados anotadas manualmente. Tendo em vista o crescimento de influência das mídias sociais, são propostos como objeto de estudos as mensagens da rede social Twitter, as quais serão classificadas em positivas ou negativas. Dado o custo de elaboração de bases de dados anotadas manualmente, principalmente se considerando a dependência de grandes volumes de dados para o sucesso de técnicas de *deep learning*, utiliza-se supervisão distante para formação de base de dados de treinamento. São abordados o uso de redes neurais convolucionais para classificação de texto e comparados seus resultados com algoritmos de *machine learning* tradicionalmente aplicadas no processamento de linguagem natural.

Palavras-Chave: machine learning, deep learning, processamento de linguagem natural, análise de sentimento.

ABSTRACT

The present work presents the elaboration of a deep learning classifier for sentiment analysis of messages of social networks without the need of annotated datasets. In view of the growing influence of social media, Twitter social network messages are proposed as object of study, which will be classified in positive or negative. Given the cost of developing manually annotated datasets, especially considering the dependence of large volumes of data for the success of deep learning techniques, distant supervision is used to build a training dataset. The use of convolutional networks for text classification is discussed and their results compared with machine learning algorithms traditionally applied in natural language processing.

Keywords: machine learning, deep learning, natural language processing, sentiment analysis.

SIGLAS

UFRJ - Universidade Federal do Rio de Janeiro

EMQ - Erro Médio Quadrático

SVM - *Suport Vector Machine*

Sumário

1	Introdução	1
1.1	Tema	1
1.2	Delimitação	2
1.3	Justificativa	2
1.4	Objetivos	3
1.5	Metodologia	3
1.6	Organização	4
2	Contextualização	6
2.1	Análise de Sentimento	7
2.2	Twitter	10
3	Processamento de Linguagem Natural	12
3.1	Pré-processamentos	12
3.2	Representações Numéricas	13
3.2.1	Codificação One-Hot	13
3.2.2	Bag-of-Words	14
3.2.3	Word2Vec	14
4	Aprendizado Supervisionado	16
4.1	Naïve Bayes	16
4.2	Support Vector Machine	17
4.3	Redes Neurais	20
4.3.1	Deep Learning	23
4.3.2	Redes Neurais Convolucionais	25
4.4	Funções Custo	26

4.4.1	Erro Médio Quadrático	26
4.4.2	Erro Médio Absoluto	26
4.4.3	Erro Médio Absoluto Percentual	27
4.4.4	Entropia Cruzada	27
4.5	Otimizadores	28
4.5.1	Gradiente Descendente Estocástico	29
4.5.2	Gradiente Descendente com Momento	29
4.5.3	Adam	30
4.6	Generalização	31
4.6.1	Regularização de Tikhonov	33
4.6.2	Dropout	35
4.7	Supervisão Distante	36
5	Métodos	37
5.1	Bases de Dados	37
5.2	Métricas	39
5.3	Desenvolvimento	40
6	Resultados	43
6.1	Primeira Etapa	43
6.2	Segunda Etapa	44
6.3	Terceira Etapa	49
7	Conclusões	53
7.1	Trabalhos Futuros	54
	Bibliografia	55

Lista de Figuras

2.1	Exemplo de <i>tweet</i> contendo <i>emoticon</i> , abreviação e neologismos. . . .	11
4.1	Reta de maior margem entre classes.	18
4.2	SVM em dados não linearmente separáveis.	19
4.3	Transformação de dados por função de base radial.	19
4.4	SVM com <i>kernel</i> de base radial.	20
4.5	Rede neural multilayer perceptron	21
4.6	Diferentes pontos de ajuste dos algoritmos de aprendizado de máquina.	34
6.1	Seleção de hiperparâmetros de Naïve Bayes - Etapa 1.	44
6.2	Seleção de hiperparâmetros de SVM - Etapa 1.	45
6.3	Seleção de hiperparâmetros de Naïve Bayes - Etapa 2.	46
6.4	Seleção de hiperparâmetros de SVM - Etapa 2.	47
6.5	Curva ROC dos modelos aplicados aos dados de teste do Sentiment140.	47
6.6	Curva ROC dos modelos aplicados aos dados de teste do SemEval.	48

Lista de Tabelas

2.1	Exemplo de classificação de sentimento em <i>tweets</i>	10
2.2	Dificuldades encontradas na classificação de sentimento.	10
5.1	Emoticons selecionados para aplicação de supervisão distante.	38
5.2	Acurácia obtida pelos classificadores do Sentiment140.	40
6.1	Comparação de resultados da replicação dos classificadores do Senti- ment140.	45
6.2	Resultados obtidos pelos classificadores.	49
6.3	Seleção de hiperparâmetros de CNN.	52

Lista de Algoritmos

4.1	Gradiente Descendente	29
4.2	Gradiente Descendente com Momento	30
4.3	Adam	32

Capítulo 1

Introdução

Neste capítulo serão descritos brevemente o problema abordado, as áreas de conhecimento que cerceiam o estudo, e os métodos que compõem a solução proposta por este trabalho. Serão ainda ressaltados os motivos que caracterizam a necessidade do projeto apresentado e seu objetivo.

1.1 Tema

O tema do projeto é o estudo de técnicas de aprendizado de máquina para processamento de linguagem natural de redes sociais. Em especial, avaliar o impacto da utilização de técnicas de *deep learning*, recentemente desenvolvidas, quando aplicadas neste contexto.

O processamento de linguagem natural é o campo dentro de inteligência artificial que estuda a geração e compreensão de línguas humanas. Uma das principais características deste campo é o fato de lidar com dados não estruturados. Nesse sentido, estudos que envolvam formas de comunicação, como por texto, apresentam diversas dificuldades. Dentre elas estão: diferentes idiomas e alfabetos, variações linguísticas temporais e determinadas pelo meio em que a mensagem está inserida, entre outros. Somando-se a este fato que apesar de estudos envolvendo o processamento de linguagem natural começarem a aparecer desde a década de 50, as redes sociais são um fenômeno recente. Por sua vez, estas se diferem de textos mais amplamente abordados principalmente por seu tamanho normalmente reduzido, seu nível de formalidade, e a frequente associação com mídias não textuais. Tais dis-

tinções ressaltam a necessidade de desenvolvimento e validação de técnicas aplicadas especificamente ao contexto de mídias sociais.

1.2 Delimitação

O objeto de estudo deste trabalho são os chamados *tweets*, mensagens curtas publicadas na rede social Twitter. O Twitter é uma das principais redes sociais contando com 310 milhões de usuários ativos, gerando um total de meio bilhão de *tweets* por dia.

Sobre essas mensagens será aplicada a análise de sentimento, na qual seu enfoque será na polaridade das mensagens, separando-as entre positivas e negativas. Não serão abordadas por esse projeto a objetividade ou neutralidade de um texto.

Por fim, o modelo a ser obtido visa a classificação de mensagens escritas em língua inglesa. A língua inglesa foi escolhida por ter o maior conjunto de estudos e dados disponíveis. Desta forma, é possível validar o método proposto comparando seus resultados com os obtidos na literatura.

1.3 Justificativa

As redes sociais são escolhidas como objeto de estudo por que longo da última década observamos a massificação de seu uso. A medida que essas plataformas passam a ser cada vez mais relevantes, cresce a importância de se analisar o conteúdo que trafega pelas redes. Dentre essas mensagens, frequentemente se encontram opiniões sobre um assunto, evento ou produto. Neste quesito, é possível aplicar a análise de sentimento, ou mineração de opinião, que é o campo de estudo que visa resgatar as informações que transitam nos textos, de maneira a agregar conhecimento sobre os tópicos falados.

Entretanto, o crescimento da produção de dados inviabiliza o processo de análise manual deste conteúdo. Logo, há a necessidade de técnicas, como a abordada por este trabalho, capazes de operar dados no volume em que são gerados atualmente.

Em paralelo, observamos nos últimos anos o desenvolvimento do chamado *deep learning*, que é uma família de técnicas as quais tomam decisões em nível mais

alto de abstração. Estes níveis são alcançados através do mapeamento obtidos pelos dados por redes neurais de diversas camadas. Este conjunto de técnicas, viabilizadas pelo aumento de poder computacional, apresentou sucesso nas mais diversas aplicações como no processamento de imagens, reconhecimento de fala, entre outros. O campo do processamento de linguagem natural, por sua vez, foi fortemente impactado por estes algoritmos. Ferramentas como redes neurais convolutivas, a princípio desenvolvidas para processamento de imagens, e *long short-term memory* propulsionaram o salto de desempenho obtido nos últimos anos. O impacto deste avanço é notório no nosso dia a dia. Frequentemente utilizamos, por exemplo, serviços automatizados de atendimento ao cliente ou ferramentas de tradução simultânea que fazem uso desta abordagem.

É necessário ressaltar que parte do sucesso atribuído ao *deep learning* provem de seu maior poder de abstração dos dados. Entretanto, essa característica faz com que haja uma dependência de grandes volumes de dados durante o treinamento destes algoritmos. Por esse motivo, se resalta a necessidade de métodos capaz de gerar bases de treinamento de maneira automática.

1.4 Objetivos

O objetivo deste projeto, portanto, consiste em gerar um modelo computacional capaz de sistematizar a classificação de sentimento de *tweets*. A produção deste modelo será feita de maneira a não depender de uma base de dados anotada manualmente, visto o alto custo desta ser reproduzida.

1.5 Metodologia

Para alcançar esse objetivo, o projeto foi dividido em três etapas: (1) replicar técnicas consolidadas de análise de sentimento para *tweets* sobre dados de treinamento obtidos por anotação automática; (2) formar uma base de dados própria e aplicar os mesmos algoritmos utilizados anteriormente para validar o procedimento de sua elaboração; (3) aplicar nos *tweets* técnicas de *deep learning* que vêm obtendo sucesso em processamento de linguagem natural e avaliar o impacto das mesmas comparado-as a classificadores menos robustos.

A primeira etapa do trabalho será a replicação de estudos desenvolvidos por Go *et al.* [1], que aplica técnicas de *Naïve Bayes* e *support vector machine* na classificação de polaridade de *tweets*, sendo seu treinamento feito em cima de uma base de dados disponibilizada por Go *et al.* formada por anotação automática. Os resultados obtidos por estas técnicas serão utilizados como patamar para comparação dos modelos posteriormente gerados.

Posteriormente, a segunda etapa consiste em produzir uma base de treinamento gerada pelo método proposto por Go *et al.*, anotada automaticamente. Para sua validação, serão replicados os mesmo algoritmos de *Naïve Bayes* e *support vector machine*, treinados com esse novo conjunto de dados, e comparados seus resultados com os obtidos pelos classificadores da etapa anterior.

Finalmente, a terceira etapa fundamenta-se na aplicação de técnicas de *Deep Learning*, como apresentadas por Kim [2], em que se utiliza redes neurais convolucionais para classificação de texto. O treinamento continua sendo feito a partir do banco de dados produzido por anotação automática, o qual foi apresentado anteriormente. Serão comparados seus resultados com os obtidos pelos classificadores que compõe a etapa anterior para analisar a eficiência de algoritmos de *deep learning* aplicadas ao processamento de linguagem natural de redes sociais.

1.6 Organização

O Capítulo 2 aborda definições relevantes ao problema que será abordado, ressaltam-se a importância do objetivo proposto, situa o trabalho em sua área de conhecimento e apresenta o Twitter, objeto do estudo.

O Capítulo 3 apresenta as técnicas necessárias para permitir o uso de aprendizado de máquina no processamento de texto. Este Capítulo mostra as diferentes abordagens possíveis para transformações de texto em valores numéricos, ressaltando como estas decisões afetam os algoritmos de aprendizado posteriormente aplicados.

Os algoritmos de aprendizado, por sua vez, são apresentados no Capítulo 4. Nele serão explicitadas as fundamentações teóricas de cada algoritmo e variações necessárias para aplicações destas técnicas em texto. Também são apresentadas as métricas de avaliação de resultado que foram utilizadas para a realização do projeto.

O Capítulo 5 descreve em detalhes cada etapa que compõe o trabalho. Destacando-se os bancos de dados utilizados, a escolha dos parâmetros dos algoritmos de aprendizado e o procedimento metodológico aplicado.

No Capítulo 6 são apresentados os resultados obtidos pelas técnicas de aprendizado de máquina propostas. Neste Capítulo, os resultados encontrados são avaliados, comparados e discutidos.

O Capítulo 7 é composto pela conclusão, mostrando os objetivos cumpridos, destacando os pontos positivos e apontando as limitações encontradas durante o desenvolvimento do projeto.

Capítulo 2

Contextualização

A internet apresenta um crescimento exponencial no mundo atual. Estima-se que no ano de 2016 cerca de 66% da população brasileira tenha acesso à rede [3]. Ela é usada massivamente no dia a dia das pessoas, estando presente desde a realização de tarefas básicas e essenciais, como cozinhar e se locomover numa cidade, até o preenchimento do tempo de lazer, com vídeos, notícias, mensagens instantâneas e redes sociais.

Através especialmente das redes sociais, a internet modificou a forma de interação entre as pessoas. No Brasil, 58% da população, ou seja, 120 milhões de pessoas, participam de pelo menos uma rede social, gastando em média 220 minutos por dia [3]. Isso faz do Brasil o segundo país em tempo navegando em redes sociais, atrás apenas das Filipinas [3]. Isso não só cria uma nova instância de comunicação entre as pessoas, mas também abre um lugar para a expressão dos sentimentos e opiniões de cada um. Com isso, após o surgimento das redes sociais, a presença online deixou de ser uma comunicação de via única. Agora, os leitores interagem com a fonte de informação recebida, seja ela uma notícia, atualização de amigos, etc.

Essa presença central das mídias sociais na vida das pessoas faz com que os usuários passem a ser mais influenciados por opiniões que trafegam pela mesma. Desta maneira, as redes sociais passam a ter um peso maior na tomada de decisão de cada indivíduo, como na compra de um produto ou na escolha de um candidato para a ser votado. O impacto destes novos meios de comunicação podem ser observados, por exemplo, nas mobilizações de massa ocorridas na primavera árabe em 2011 que

levaram a queda de governos no qual as mídias sociais exerceram papel crítico [4].

Nesse sentido, tornaram-se necessários trabalhos que analisem o que se está dizendo nas redes sociais. Para uma emissora de TV, por exemplo, é interessante saber o que está sendo dito sobre sua programação, assim como para a assessoria de um cantor, é importante descobrir se sua nova canção está agradando ou não seu público.

Jim Yu, presidente da empresa de marketing digital BrightEdge, ressalta que os sentimentos expressos por clientes em relação a uma marca é uma das principais métricas de *branding* [5]. O colunista da Forbes, Daniel Newman, destaca que a utilização de técnicas de análise de sentimento em conjunto com outras técnicas de *big data* vem revolucionando a forma de se fazer marketing, permitindo uma maior personalização dos produtos e fortalecimento de relação com os clientes [6].

Entretanto, a criação de conteúdo digital acompanha o crescimento da internet. No ano de 2016, a cada sessenta segundos foram adicionadas 500 horas vídeo no YouTube, realizadas 3.8 milhões de buscas no Google e publicados 450 mil *tweets* [7]. Estes números tornam impraticável a realização de análise de sentimento manualmente sobre mídias com esse volume de dados. Portanto, a utilização de métodos de automatizados é um modo de viabilizar a realização dessa tarefa. Porém, a extração de informação de linguagem natural não é uma tarefa simples, pois envolve conhecimentos da língua explícitos e implícitos, regulares e irregulares, sintáticos e semânticos [8]. Sobretudo, a análise de sentimento ainda envolve problemas ainda não resolvidos no processamento de linguagem natural [8] como a resolução de correferências [9], resolução de anáforas [10], reconhecimento de entidades [11] e ambiguidade de palavras [12]. Estes fatores constituem desafios adicionais para a tarefa.

2.1 Análise de Sentimento

O campo de estudos da análise de sentimento, também chamado de mineração de opinião, é um dos ramos mais ativos do processamento de linguagem natural [13]. A análise de sentimento visa analisar a opinião, a avaliação, a emoção e a atitude de um documento em relação a um evento, produto, serviço, organização, ou qual-

quer outra entidade. Apesar do processamento de linguagem natural ter um longo histórico de estudos, a análise de sentimento se consolidou apenas no início dos anos 2000 com a crescente demanda comercial. Observa-se que o crescimento de interesse nessa área tem grande correlação com o crescimento de mídias sociais e do volume de dados, visto que o sucesso das redes sociais disponibilizam quantidades massivas de dados possibilitando tanto a oferta de novas aplicações quanto oportunidades de desenvolvimento de técnicas [13].

O campo de análise de sentimento apresenta diferentes subáreas de estudo. Dentre elas, a tarefa mais presente na literatura e que abordaremos neste trabalho é a análise de polaridade de um documento. A análise de polaridade visa classificar textos em uma escala entre positivo e negativo. Encontra-se também estudos que visam determinar a subjetividade ou objetividade de um texto, como no trabalho de Wiebe e Rilof [14]. Também entra no escopo de análise de sentimento a classificação de emoções presentes em uma mensagem, como felicidade, raiva, tristeza, etc [15]. Outro foco de pesquisa com grande aplicações práticas é a análise de sentimento porém não da mensagem como um todo, mas do sentimento em relação a uma entidade presente na mensagem. O contrário também é possível, a análise da influência do sentimento de um termo em relação a mensagem. Percebe-se que a análise de sentimento se desmembra em varias subdivisões e, novas subdivisões surgem a cada dia visto que a área ainda está em processo de expansão, o que confirma seu grande potencial de aplicação.

Com o crescimento da participação dos usuários na formação de conteúdo da Web torna-se cada vez mais relevante a análise do conteúdo sendo produzido. Zhuang *et al.* [16], por exemplo, utiliza mineração de opinião para extrair avaliações a partir de críticas de filmes feitas por usuários. Hu e Liu [17], por sua vez, desenvolveram métodos para resumo de sentimento de avaliações de produtos vendidos por comercio eletrônico.

A utilização de classificadores de opinião sobre dados de redes sociais se mostrou relevante em diversas aplicações. Bollen *et al.* [18], por exemplo, aplicam análise de sentimento de mídias sociais para predição de mercados financeiros. Tumasjan *et al.* [19], por sua vez, utilizam o sentimento exposto no Twitter para prever resultados eleitorais. Até mesmo na predição de bilheteria de filmes, como feito por Du

et al. [20].

Porém, a produção de base de dados para o treinamento de modelos de aprendizado supervisionado é um processo custoso. O seu processo de desenvolvimento envolve a anotação manual de dados, preferencialmente por múltiplas pessoas. Contudo, apesar de modelos simples como os obtidos com técnicas como Naïve Bayes e *support vector machines* conseguirem desempenhos satisfatórios com quantidades limitadas de dados, modelos mais complexos como redes neurais profundas dependem de grandes volumes de dados para obter êxito, tornando o processo de anotação manual extremamente ineficiente. Sobretudo, uma vez formada esta base de dados, ela só é eficiente para o desenvolvimento de classificadores aplicados na mesma língua, no mesmo formato, e sem grandes defasagens de tempo. Por exemplo, uma base de dados composta por *tweets* seria ineficiente para modelagem de classificadores de emails, ou mesmo o uso de uma base formada por *tweets* de 2010 sofreria queda de performance se o modelo treinado com esta for aplicado em *tweets* de 2018, visto a evolução da língua neste período de tempo.

Portanto, se faz necessário desenvolver métodos como o apresentado por Go *et al.* [1], que desenvolve uma base composta de *tweets* a partir de anotação automática por *emoticons*. O trabalho de Go *et al.* [1] visa definir um sistema de classificação de sentimento sem a necessidade de anotação manual dos dados para treinamento, utilizando a técnica de supervisão distante, técnica de anotação ruidosa automática. Neste artigo a análise de sentimento tratada é a extração de polaridade, negativa ou positiva, de uma mensagem. Em especial, as mensagens desse trabalho são originárias do Twitter, serviço de microblogs, sendo este trabalho o primeiro a estudar este tipo de mensagem.

Por ser uma plataforma aberta, o Twitter é composto de mensagens dos mais diversos domínios. Esse fator dificulta a tarefa de classificação a ser executada quando comparado com trabalhos de um domínio específico, como por exemplo a análise de sentimento para avaliação de filmes a partir de resenhas; ou na predição de flutuações do mercado financeiro pela análise de artigos jornalísticos.

A Tabela 2.1 exemplifica a classificação de polaridade de *tweets*, foco deste trabalho. As mensagens presentes nessa tabela são transcrições de *tweets* reais selecionados de maneira a ilustrar a classificação de sentimento.

Sentimento	<i>Tweet</i>
Positivo	Grande notícia, João Sousa e Gastão Elias representam Portugal nos Jogos Olímpicos do Rio de Janeiro.
Neutro	Entrevistei hoje a Priscilla Carnaval do BMX. Classificada para o #Rio2016, ela tem novidades na preparação olímpica.
Negativo	Estão transformando Olimpíadas que é algo sério num espetáculo triste e de mau gosto.

Tabela 2.1: Exemplo de classificação de sentimento em *tweets*.

Entretanto, nem todos exemplos são claramente distinguíveis como os apresentados na Tabela 2.1. A comunicação frequentemente apresenta fatores como ironia, ambiguidade de sentimentos e multiplicidade de idiomas. Em outros casos, a mensagem sendo analisada pode ser complementar a mensagens anteriores ou informações não textuais que a acompanham. Estes fatores são complexidades adicionais e diminuem a acurácia dos classificadores. A Tabela 2.2 demonstra exemplos das dificuldades indicadas.

Fator	<i>Tweet</i>
Ironia	Recomendo chegar para dar aula e descobrir que mudaram seu horário sem avisar.
Ambiguidade	Estou igualmente fascinada e enojada.
Multiplicidade de idiomas	Macarrão de arroz is the new miojo.

Tabela 2.2: Dificuldades encontradas na classificação de sentimento.

2.2 Twitter

O Twitter é um microblog e rede social no qual os usuários interagem a partir de mensagens, chamadas *tweets*. Microblogs são meios de comunicação que permitem usuários a compartilhar conteúdos e mídias. As principais características dos microblogs são a brevidade e instantaneidade. Os *tweets* são limitados a 140



Figura 2.1: Exemplo de *tweet* contendo *emoticon*, abreviação e neologismos.

As informações do usuário foram anonimizadas para sua privacidade.

caracteres e podem conter fotos, vídeos, links, localizações referências a usuários e *hashtags*. Criada em 2006, a rede social cresceu rapidamente a ponto de se tornar uma das redes com maior número de usuários no mundo. No ano de 2016 o Twitter a plataforma contava com a presença de 330 milhões de usuários ativos por mês. A Figura 2.1 mostra um exemplo de *tweet*.

Um ponto chave da comunicação por Twitter são as *hashtags*. *Hashtags* são palavras-chave ou frases que definem um tópico ou tema. Elas são caracterizadas por serem precedidas pelo símbolo da tralha (#) e por não conterem espaços ou pontuação. São, ainda, elementos clicáveis que apontam o leitor para outra página. Essa outra página, por sua vez, contém um mural com todas as citações contendo aquela *hashtag*. Desta forma, *hashtags* funcionam como agregadores de *tweets*.

O tamanho reduzido das mensagens veiculadas por Twitter e o meio pelo qual elas circulam levam a utilização de variações gramaticais da língua específicas para este ambiente. É comum encontrar, por exemplo, o uso excessivo de abreviações para reduzir a contagem total de caracteres, a repetição de pontuação para reforçar intensidade e o uso de neologismos.

Se tratando de classificação de sentimento, o fato do tamanho reduzido e linguagem própria apresentam uma dificuldade adicional. Apesar de não serem abordadas nesse trabalho, atributos não textuais como: republicação ou citação de mensagem, presença de foto ou vídeo, grafos de conexões entre usuários etc podem colaborar na realização da tarefa de análise de sentimento.

Capítulo 3

Processamento de Linguagem Natural

O processamento de linguagem natural (NLP), é um dos principais ramos da Inteligência Artificial. Um dos marcos iniciais desta área de conhecimento foi o teste de Turing [21] no qual a capacidade de comunicação é proposta como um critério de inteligência. Desde sua criação, no início dos anos 50, o NLP se consolidou de maneira a estar presente no dia a dia das pessoas como se observa na utilização de serviços automatizados de atendimento ao clientes, ferramentas de tradução, e outros. Grande parte destas aplicações provém da utilização de técnicas de aprendizado supervisionado que serão apresentadas no Capítulo 4.

As seções a seguir apresentam as transformações aplicadas a textos necessárias para viabilizar a aplicação de algoritmos de aprendizado.

3.1 Pré-processamentos

A primeira etapa na preparação de uma mensagem para utilização em algoritmos de aprendizado de máquina é a separação por palavras. Este processo é chamado de *tokenização*, denominando-se cada palavra resultante como um token. A *tokenização* costuma incluir a separação de contrações como a transformação de *dela* em *de* e *ela*.

Para aplicação de tarefas como classificação, é comum se remover as palavras mais comuns de um idioma e que não agregam informação discriminante para o ob-

jetivo desejado, essas palavras são chamadas de *stopwords*. A remoção de *stopwords* visa reduzir o ruído presente em dados textuais, aumentando a acurácia dos algoritmos de aprendizado de máquina e reduzindo a complexidade do problema [22]. Porém, a remoção de *stopwords* provindas de listas pré compiladas, técnica mais utilizada, apresenta desafios devido a dinamicidade de meios como redes sociais. Saif *et al.* [23] apresentam um estudo comparativo da remoção de *stopwords* por diferentes técnicas e seus efeitos na classificação de sentimento de *tweets*. Além da remoção de *stopwords*, em mensagens provindas de redes sociais também são costumeiramente removidos links, *hashtags* e menções a usuários.

Também é possível realizar a correção ortográfica das palavras e a lematização, processo de extração do radical da palavra. Estas técnicas são normalmente empregadas para reduzir o vocabulário e diminuir assim a complexidade do treinamento.

3.2 Representações Numéricas

Contudo, a utilização de algoritmos de aprendizado de máquina depende de uma representação numérica dos dados. Após a conversão de uma mensagem em uma uma sequência de tokens, há uma etapa de transformação destas cadeias de tokens em matrizes ou vetores. Serão apresentadas nas subseções seguintes as principais técnicas referentes a este processo.

3.2.1 Codificação One-Hot

Um dos métodos mais observados na literatura é a codificação *one-hot*, na qual cada token é representado de maneira maximamente esparsa. Ou seja, é definido um espaço cujo número de dimensão é dada pelo tamanho do vocabulário utilizado e cada palavra é substituída por um vetor unitário na direção que representa sua posição no vocabulário. Desta maneira, cada token é representado por um vetor unitário de dimensão igual a do vocabulário.

Outra forma possível de utilização é transformar cada par de palavras em um token. A utilização de múltiplas palavras por token é chamada de *n-gram*. A ideia da aplicação de *n-gram* é capturar expressões ou distinguir palavras em

diferentes contextos. Porém, sua utilização aumenta significativamente o tamanho do vocabulário, podendo dificultar o processo de treinamento.

3.2.2 Bag-of-Words

Uma vez que cada palavra é caracterizada por sua codificação *one-hot*, a forma natural de simbolizar uma mensagem é por uma matriz na qual cada coluna é composta pelo vetor correspondente a cada palavra da frase. Contudo, para evitar o aumento da dimensionalidade do problema é comum se utilizar da técnica *bag-of-words* na qual cada mensagem é representada pela soma dos vetores de seus tokens [24].

Esta técnica também é chamada na literatura como *term frequency* (tf), dado que a técnica é dada pela contagem de termos de um documento. Nesse caso as palavras são pesadas igualmente, outra abordagem possível é multiplicar a frequência do termo no documento pelo inverso do número de documentos no qual aquele termo aparece, técnica denominada *term frequency-inverse document frequency* (TF-IDF) [25]. A TF-IDF permite compensar a grande presença de palavras muito frequentes, que limitariam a influência de termos pouco usados.

É possível notar que a utilização de ambas as técnicas ignora a ordem das palavras na frase. Técnicas como Word2Vec, apresentada na Seção 3.2.3, permitem a utilização da posição do termo na mensagem por conter uma representação densa de cada token, reduzindo a dimensionalidade do problema.

3.2.3 Word2Vec

Word2Vec [26] (W2V) é uma técnica que visa representar uma palavra por um vetor de números reais, denso e de tamanho arbitrário. Os resultados obtidos por Mikolov *et al.* [26] mostram que a representação vetorial é capaz de capturar parte do sentido semântico dos termos. Na prática, isso significa que, por exemplo, palavras sinônimas ficam próximas entre si no *embedding* obtido.

Desenvolvida por um grupo de engenheiros do Google, estes vetores são aprendidos a partir de uma janela de contexto ao redor de cada palavra presente nos documentos de treinamento.

Foram desenvolvidos dois modelos de treinamento distintos, são eles o *Continuous Bag-of-Words* (CBOW) e o *Skipgram*. A diferença entre ambos, além da forma de treinamento, se dá principalmente pelo tempo de convergência e pela acurácia. No treinamento por CBOW, o W2V visa prever o termo central da janela a partir das palavras que a rodeiam. No modelo obtido por *Skipgram* o treinamento, por sua vez, é dado de forma a prever as palavras de contexto a partir do termo central da janela.

O modelo Word2Vec se constitui de uma rede neural de uma única camada escondida em que suas entradas e objetivos são a codificação *one-hot* dos termos.

O número de neurônios presentes na camada escondida equivale a dimensionalidade do *embedding*. No caso do treinamento por CBOW a entrada é dada por palavras de contexto, ou seja, as palavras ao redor do termo a ser treinado, sendo esse termo o objetivo da rede. A sua primeira camada possui ativação linear, com coeficiente $\frac{1}{c}$ no qual c representa o tamanho da janela utilizada. Enquanto a segunda camada é composta por ativação *softmax*. O treinamento do modelo *Skipgram* é análogo, porém com as entradas e saídas invertidas. Os pesos do modelo Word2Vec são os obtidos na primeira camada após o treinamento da rede.

Por não necessitar de anotação se faz possível a utilização de conjuntos massivos de textos no treinamento do Word2Vec. Entretanto, apesar do uso de grandes volumes de dados permitir um melhor desempenho do modelo esse fator também aumenta significativamente o tempo de treinamento. Para abordar esse problema, Mikolov *et al.* [27] apresentam uma técnica de treinamento com amostragem, acelerando o processo significativamente.

Apesar de ser o *embedding* mais conhecido para representação de texto, outros grupos de pesquisadores desenvolveram técnicas com a mesma finalidade. Dentre elas, destacam-se GloVe [28], algoritmo desenvolvido por um grupo de pesquisadores de Stanford e FastText [29], produzido por uma equipe do Facebook.

Capítulo 4

Aprendizado Supervisionado

O aprendizado supervisionado é o campo dentro de aprendizado de máquina que visa a gerar modelos preditivos a partir de um conjunto de dados de treinamento cujos resultados são previamente conhecidos.

4.1 Naïve Bayes

Naïve Bayes é uma das técnicas mais simples disponíveis nesse contexto. Ela se baseia no teorema de Bayes enquanto assume independência entre as características escolhidas para descrever o dado. Será demonstrada sua formulação matemática, como descrita por Schütze [24].

Tendo \mathbf{x} tal que $\mathbf{x} \in \mathbf{X}$ em que \mathbf{X} é o conjunto de dados de treinamento, a sua probabilidade de pertencer a classe $c_k \in \mathbf{c}$, sendo \mathbf{c} o conjunto de classes, é dada pelo teorema de Bayes:

$$p(c_k | \mathbf{x}) = \frac{p(c_k) p(\mathbf{x} | c_k)}{p(\mathbf{x})} \quad (4.1)$$

Cada observação \mathbf{x} é um vetor de n características, e ao se assumir independência entre elas obtém-se:

$$p(x_i | x_{i+1}, \dots, x_n, c_k) = p(x_i | c_k) \quad (4.2)$$

Logo, pode-se reescrever a equação 4.1 substituindo $p(\mathbf{x} | c_k)$ pelo produto de suas características:

$$p(c_k | \mathbf{x}) = \frac{p(c_k) \prod_{i=1}^n p(x_i | c_k)}{p(\mathbf{x})} \quad (4.3)$$

Como $p(\mathbf{x})$ será uma constante dado cada exemplo \mathbf{x} esta pode ser desprezada:

$$p(c_k | \mathbf{x}) \propto p(c_k) \prod_{i=1}^n p(x_i | c_k) \quad (4.4)$$

Portanto, tem-se que o estimador ótimo \hat{y} escolherá pela classe que atinja maior probabilidade:

$$\hat{y} = \max_{k \in \{1, \dots, K\}} p(c_k) \prod_{i=1}^n p(x_i | c_k) \quad (4.5)$$

Vê-se então que o modelo de *Naïve Bayes* depende apenas de $p(c_k)$ e $p(\mathbf{x} | c_k)$. Estes parâmetros serão extraídos do conjunto de treino por máxima verossimilhança.

Dado um vetor \mathbf{y} de tamanho m que representa as classificações referentes a \mathbf{X} , pode-se estimar $p(c_k)$ pela contagem de vezes que a classe c_k aparece no conjunto de treinamento:

$$\hat{p}(c_k) = \frac{\sum_{i=1}^m [y_i = c_k]}{m} \quad (4.6)$$

Por sua vez, $p(x_r | c_k)$ é estimado utilizando a contagem de vezes que uma característica aparece dividida pelo total de características presentes no subconjunto \mathbf{X}' , subconjunto de dados de treinamento pertencentes a classe c_k :

$$\hat{p}(x_r | c_k) = \frac{\sum_{j=1}^{m'} \sum_{i=1}^n [x_{ji} = x_r]}{|\mathbf{X}'|} \quad (4.7)$$

Vê-se que modelos montados a partir de *Naïve Bayes* são computacionalmente baratos dado que seus parâmetros são obtidos através de contagens sobre os dados de treinamento e que sua predição utiliza apenas multiplicações. Embora se baseie na independência entre características, seu baixo custo operacional leva esta técnica a ser utilizada mesmo em problemas com notória dependência entre características como a classificação de texto [30].

4.2 Support Vector Machine

O conceito fundamental do *Support Vector Machine* (SVM) se dá pela obtenção de vetores de suporte que melhor separe as classes. Esta separação é feita de maneira que se maximize a margem entre as classes. A Figura 4.1 demonstra dados de duas classes distintas, representadas pelas cores rosa e amarelo, pertencentes a um espaço de características de duas dimensões; vê-se na figura que a reta que define a maior separação é suportada pelos dados de cada classe mais próximos a ela.

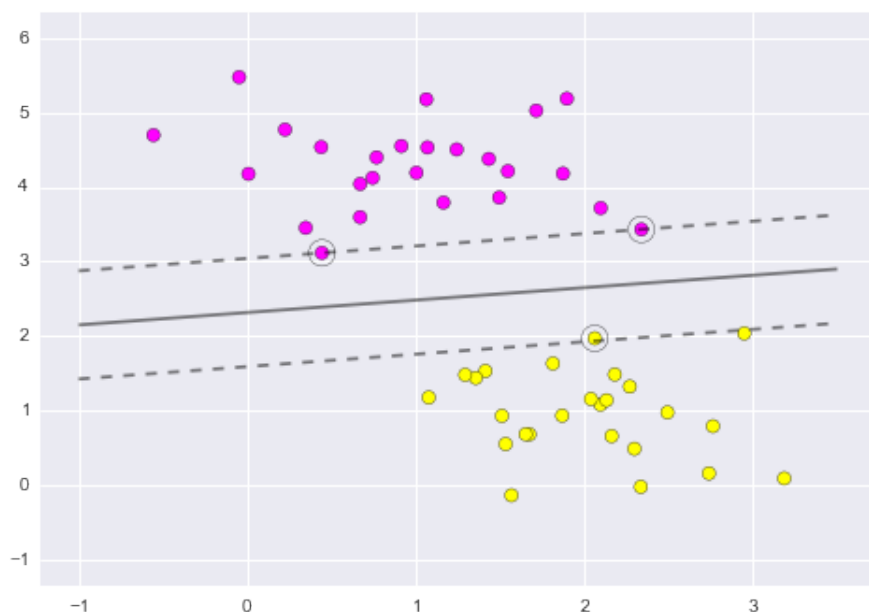


Figura 4.1: Reta de maior margem entre classes.

Imagem com direitos cedidos para uso não comercial, retirada de [31]

Como utilizam vetores de suporte para definir hiperplanos de separação, SVM não são capazes de segregar classes não linearmente distinguíveis. Podemos observar um exemplo deste caso na Figura 4.2, na qual uma SVM treinado tenta separar classes concêntricas. Para contornar esse impedimento, foi elaborado o que se chamou de *kernel trick*. Este se baseia em um mapeamento não linear dos dados para um espaço onde possam ser linearmente separáveis [32]. A Figura 4.3 mostra a representação dos dados apresentados na Figura 4.2 após seu mapeamento por uma função de base radial.

Neste novo espaço definido pela transformação, os dados são linearmente separáveis. Portanto, é possível achar um vetor de suporte que defina um hiperplano de separação das classes. Vemos na Figura 4.4 as margens encontradas.

Porém, por se basear nos dados próximos ao limiar de separação das classes, o algoritmo é incapaz de distinguir os casos de classes que não são separáveis sem erros. A solução desse problema foi a criação de uma variável de relaxamento que define um número máximo de erros de classificação permitido. Esta propriedade é descrita com mais detalhes por Cortes e Vapnik em [33]. Sua utilização permite o desenvolvimento de modelos mais robustos a *outliers* e melhora a generalização.

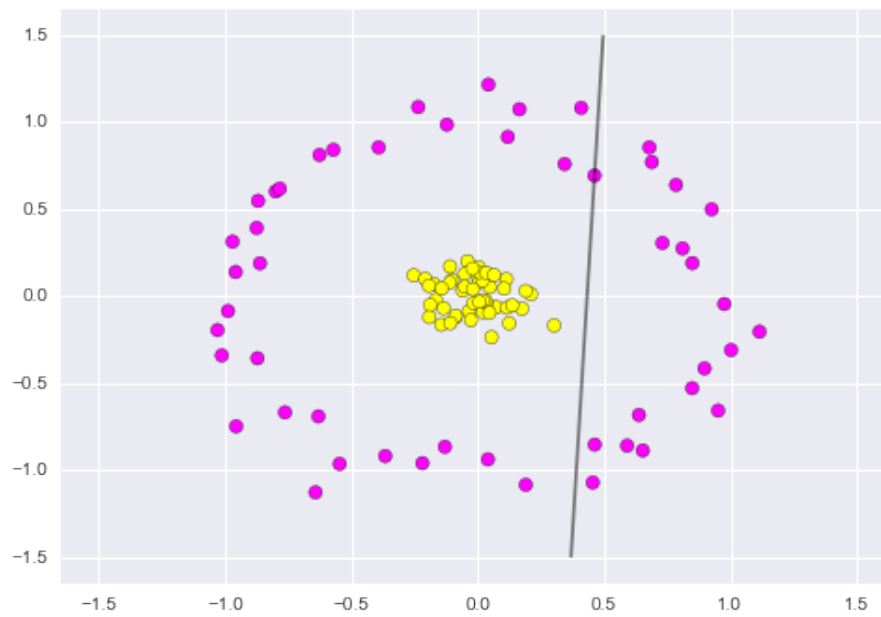


Figura 4.2: SVM em dados não linearmente separáveis.

Imagem com direitos cedidos para uso não comercial, retirada de [31]

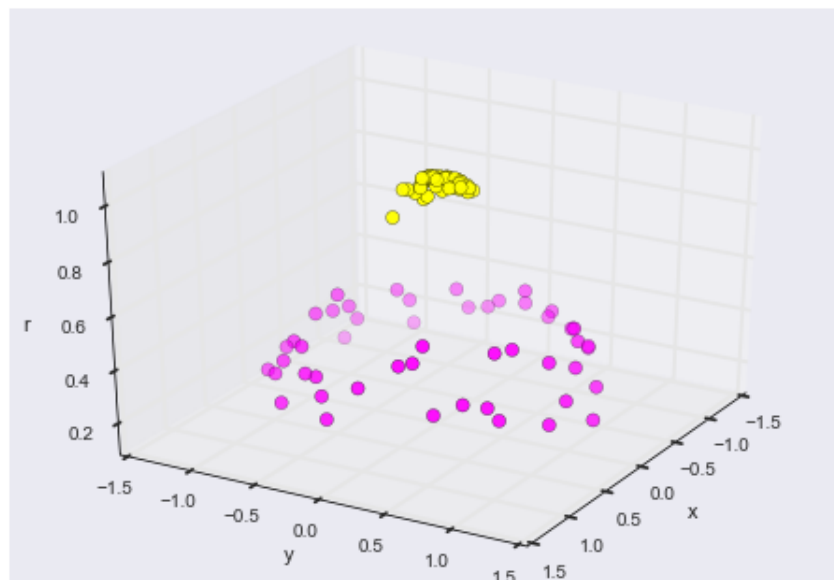


Figura 4.3: Transformação de dados por função de base radial.

Imagem com direitos cedidos para uso não comercial, retirada de [31]

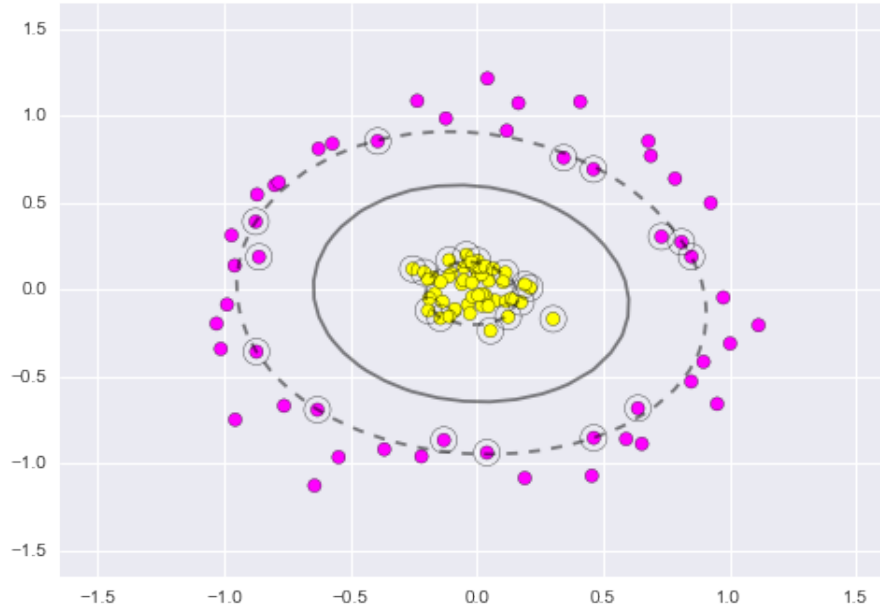


Figura 4.4: SVM com *kernel* de base radial.

Imagem com direitos cedidos para uso não comercial, retirada de [31]

Outros exemplos de regularizadores, como este, serão apresentados na Subseção 4.6.

Uma limitação na utilização deste algoritmo é seu tempo de treinamento. Sua complexidade computacional fica entre $O(n_{caracteristicas} \times n_{dados}^2)$ e $O(n_{caracteristicas} \times n_{dados}^3)$ [34]. Entretanto, Suykens e Vandewalle desenvolveram uma função custo através da qual tornou possível realizar o treinamento de SVM a partir da otimização pelo método do gradiente [35], viabilizando sua utilização em ambientes com limitações computacionais.

4.3 Redes Neurais

Redes neurais são sistemas computacionais que visam replicar o modelo de processamento do cérebro [36]. Inspirado pelo campo do conexionismo, em 1958 Rosenblatt [37] desenvolveu o algoritmo do perceptron. O perceptron foi uma das primeiras formas de redes neurais desenvolvidas. Eles são classificadores binários compostos por duas camadas de neurônios artificiais, uma de entrada e uma de saída, interligados por sinapses. As sinapses funcionam como um ponderador linear dos neurônios da camada anterior. Os neurônios, por sua vez, representam uma

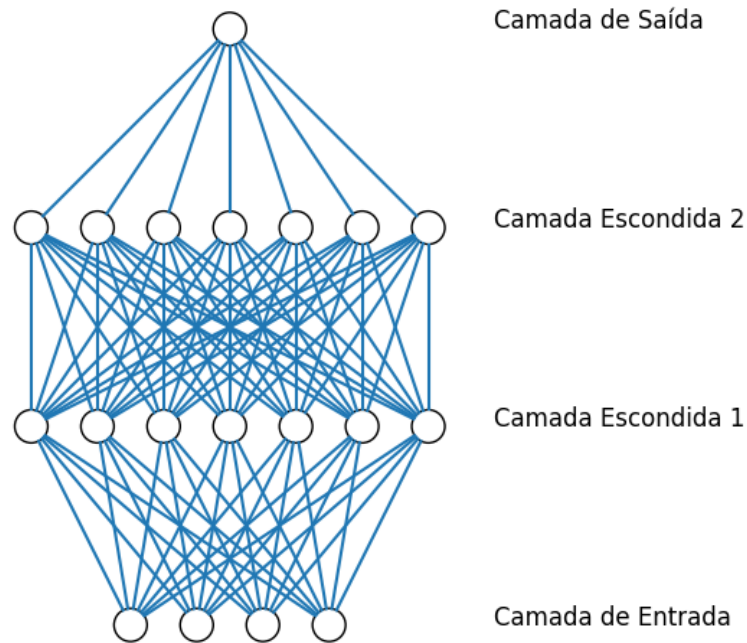


Figura 4.5: Rede neural multilayer perceptron

função de ativação, no caso do perceptron a função degrau, porém de maneira mais geral dispõem-se de funções como: tangente hiperbólica, logística etc.

A utilização de perceptrons com múltiplas camadas sucessivas de neurônios formam as redes neurais multilayer perceptron (MLP), como exemplificado na Figura 4.5, onde apresenta-se o diagrama de uma rede que conta com uma camada de entrada que representa os dados do problema, seguida de duas camadas intermediárias, comumente chamadas de escondidas, as quais são seguidas pela camada de saída, que apontará o resultado final de todo processamento da rede.

Uma de suas principais características de redes neurais é a capacidade de aproximar qualquer função, dado que a rede contenha pelo menos uma camada escondida com número suficientemente grande de neurônios com ativação não linear [38].

Porém, no final dos anos 60, Minsky e Papert [39] publicaram um estudo no qual prova que redes neurais com apenas duas camadas, como o perceptron, são incapazes de realizar classificações não lineares. Esta descoberta, junto com a inviabilidade computacional de se trabalhar com redes maiores na época resultou em uma estagnação dos estudos de redes neurais artificiais.

Apenas em 1982 quando Werbos [40] aplicou o método de otimização por auto diferenciação em redes neurais, facilitando o aprendizado de redes de múltiplas camadas, que o estudo de redes neurais voltou a receber atenção. Este método foi chamado de backpropagation. Hinton *et al.* [41] afirmam que este método é capaz de obter representações nas camadas intermediárias que são apropriadas para resolução do problema em questão.

Será descrito a seguir um exemplo de treinamento por backpropagation aplicado a uma rede de uma camadas escondida e uma saída.

A primeira etapa para utilização do backpropagation é computar o valor previsto \hat{y}_i pela rede neural para uma dada observação de índice i do conjunto de treinamento, como descrito nas equações 4.8a-4.8d, nas quais f representa a função de ativação; \mathbf{x}_i o conjunto de características que descreve esse exemplo; \mathbf{W} e \mathbf{b} os pesos e o *bias* respectivamente; e \mathbf{h}_i indica a camada escondida.

$$\mathbf{a}_i^{(1)} = \mathbf{W}^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)} \quad (4.8a)$$

$$\mathbf{h}_i = f(\mathbf{a}_i^{(1)}) \quad (4.8b)$$

$$a_i^{(2)} = \mathbf{W}^{(2)}\mathbf{h}_i + b^{(2)} \quad (4.8c)$$

$$\hat{y}_i = f(a_i^{(2)}) \quad (4.8d)$$

Em seguida, utiliza-se uma função custo L para calcular o erro J de representação obtido pela rede neural a partir das previsões $\hat{\mathbf{y}}$ e das anotações \mathbf{y} , como exposto na equação 4.9. A Seção 4.4 apresenta exemplos de funções custo.

$$J_i = L(\hat{y}_i, y_i) \quad (4.9)$$

A computação do custo J conclui a parte *forward* do algoritmo, no qual os valores das saídas de cada camada da rede são armazenados para execução da etapa seguinte, a *backward*. Na fase *backward* utiliza-se o gradiente do erro da rede neural em relação a cada camada para calcular ponderar quanto cada peso influi no erro de representação. Desta maneira, parâmetros que estejam menos adaptados ao problema receberão os maiores gradientes, e consequentemente os maiores ajustes, visto que são os mais responsáveis pelo erro.

As equações 4.10a-4.10g mostram a propagação do gradiente a partir da camada de saída até camada de entrada utilizando-se os valores de \mathbf{a}_i previamente obtidos. Dispondo do gradiente da função custo em relação a saída, $\frac{\partial J_i}{\partial y_i}$, que é dependente da função escolhida, calcula-se por regra da cadeia o gradiente do erro em relação aos pesos e *bias* da camada de saída como observa-se nas equações 4.10b e 4.10c. Computa-se também o gradiente da camada escondida como apresentado na equação 4.10d. Uma vez apurado o gradiente da camada escondida, pode-se calcular os gradientes dos pesos e *bias* desta camada como presentes nas equações 4.10f e 4.10g.

$$\frac{\partial J_i}{\partial a_i^{(2)}} = \frac{\partial J_i}{\partial y_i} \frac{\partial y_i}{\partial a_i^{(2)}} = \frac{\partial J_i}{\partial y_i} f'(a_i^{(2)}) \quad (4.10a)$$

$$\frac{\partial J_i}{\partial \mathbf{W}^{(2)}} = \frac{\partial J_i}{\partial a_i^{(2)}} \frac{\partial a_i^{(2)}}{\partial \mathbf{W}^{(2)}} = \frac{\partial J_i}{\partial a_i^{(2)}} \mathbf{h}_i \quad (4.10b)$$

$$\frac{\partial J_i}{\partial b^{(2)}} = \frac{\partial J_i}{\partial a_i^{(2)}} \frac{\partial a_i^{(2)}}{\partial b^{(2)}} = \frac{\partial J_i}{\partial a_i^{(2)}} \quad (4.10c)$$

$$\frac{\partial J_i}{\partial \mathbf{h}_i} = \frac{\partial J_i}{\partial a_i^{(2)}} \frac{\partial a_i^{(2)}}{\partial \mathbf{h}_i} = \frac{\partial J_i}{\partial a_i^{(2)}} \mathbf{W}^{(2)} \quad (4.10d)$$

$$\frac{\partial J_i}{\partial \mathbf{a}_i^{(1)}} = \frac{\partial J_i}{\partial \mathbf{h}_i} \frac{\partial \mathbf{h}_i}{\partial \mathbf{a}_i^{(1)}} = \frac{\partial J_i}{\partial \mathbf{h}_i} f'(\mathbf{a}_i^{(1)}) \quad (4.10e)$$

$$\frac{\partial J_i}{\partial \mathbf{W}^{(1)}} = \frac{\partial J_i}{\partial \mathbf{a}_i^{(1)}} \frac{\partial \mathbf{a}_i^{(1)}}{\partial \mathbf{W}^{(1)}} = \frac{\partial J_i}{\partial \mathbf{a}_i^{(1)}} \mathbf{x}_i \quad (4.10f)$$

$$\frac{\partial J_i}{\partial \mathbf{b}^{(1)}} = \frac{\partial J_i}{\partial \mathbf{a}_i^{(1)}} \frac{\partial \mathbf{a}_i^{(1)}}{\partial \mathbf{b}^{(1)}} = \frac{\partial J_i}{\partial \mathbf{a}_i^{(1)}} \quad (4.10g)$$

Os gradientes dos pesos e *bias* serão utilizados para sua otimização pelo método do gradiente visando minimizar a função custo. A Seção 4.5 aborda o tema de otimizadores, focando em rede neurais.

4.3.1 Deep Learning

Com o aumento do poder computacional ao longo dos anos levou a utilização de redes neurais mais robustas. Todavia, a adição de múltiplas camadas a redes neurais feedforward apresenta dificuldades no treinamento por *backpropagation* pelo

efeito do *vanishing gradient* [42]. O *vanishing gradient* acontece quando os pesos das camadas mais externas da rede são treinados até o ponto que o gradiente dos seus erros sejam pequenos. A partir desse momento, as camadas mais próximas da entrada recebem atualizações menores em seus pesos visto que o fator de atualização depende da multiplicação dos gradientes das camadas mais externas. Este processo resulta no treinamento efetivo apenas das camadas próximas a saída, estabelecendo uma dificuldade exponencial no treinamento com relação ao número de camadas.

Somente quando Hinton *et al.* desenvolveram o pré-treinamento camada a camada [43] que começou a se contornar o problema do *vanishing gradiente*, viabilizando a aplicação de redes neurais feedforward com muitas camadas. Essa técnica consiste em se obter um conjunto de pesos de cada camada de neurônios individualmente de maneira a melhor representar a sua entrada, posteriormente esse conjunto de pesos serve como inicialização para o treinamento da rede. Outros métodos de redução do *vanishing gradiente* são utilizados, como a função de ativação ReLU (*rectified linear unit*) [44] e mais recentemente, as arquiteturas de redes neurais residuais [45].

Chamou-se de *deep learning* o campo de estudo de redes neurais com muitas camadas. Redes neurais profundas romperam barreiras de performance em problemas como reconhecimento de fala, detecção de objetos, tradução etc [46].

Cada camada de uma rede neural gera uma abstração que representa a sua entrada de modo a facilitar a tarefa a ser realizada. O encadeamento de camadas resulta em representações mais complexas dos dados, revelando relações previamente não observáveis. O aprofundamento das redes tem como consequência a redução ou eliminação da chamada *feature engineering*, processo de seleção de características, que usualmente requer expertise no domínio do problema. Considerando por exemplo o caso do diagnóstico de câncer de pele, técnicas tradicionais de classificação dependiam de extração de informações de imagens de lesões, tais como: formato, tamanho, cor etc. Modelos baseados em redes neurais profundas são capaz de obter resultados superiores, eliminando completamente esta etapa [47].

Contudo, as técnicas de *deep learning* não se limitam apenas a arquitetura multilayer perceptron. Dentre as arquitetura frequentemente utilizadas temos: redes neurais recursivas (RNN) [48], redes neurais convolucionais, *autoencoders* [49], e

redes neurais geradoras adversárias (GAN) [50]. A Seção 4.3.2 apresenta detalhes das redes neurais convolucionais, técnica aplicada neste trabalho.

4.3.2 Redes Neurais Convolucionais

A utilização de redes neurais multilayer perceptron em imagens sofre limitações, por cada pixel estar diretamente ligado aos neurônios, rotações ou translações na imagem afetam fortemente a capacidade da rede. Pretendendo solucionar estes problemas, foram desenvolvidas redes neurais convolucionais, cujas conexões são baseadas no funcionamento do córtex visual.

Elas são compostas primariamente de dois tipos de camadas: convolucionais e *pooling*. Camadas convolucionais são formadas por um conjunto de filtros espaciais compostos das mesmas funções não-lineares utilizadas em redes MLP [51]. Para cada exemplo do conjunto de dados, o filtro, cujo tamanho é menor que a entrada, é aplicado no início do dado e deslocado até o final. Desta forma, os filtros dividem os pesos entre todo espaço do dado, permitindo uma insensibilidade a deslocamentos e rotações. Por sua vez, as camadas de *pooling* visam reduzir o número total de sinapses da rede. Para tal, reduz-se o tamanho dos filtros, obtendo-se valores máximos ou médias de dimensões desses filtros. *Pooling* é fundamental quando o objetivo a ser atingido depende mais da presença de uma característica do que sua posição no dado [51].

Apesar de Redes Convolucionais terem sido desenvolvidas com intuito de solucionar problemas de visão computacional, elas também obtiveram êxito nas mais diversas áreas, como no reconhecimento de voz e em séries temporais [52]. Tal sucesso pode ser atribuído à insensibilidade da rede a deslocamentos.

Kim [2], por sua vez, propôs o uso de redes convolucionais aplicadas a texto. Para tal, os documentos são representados em uma matriz na qual cada coluna é composta pelos pesos de um *embedding*, como o Word2Vec, correspondente a cada palavra. Por cada mensagem poder ter tamanho variado, escolhe-se um número de tokens n , para representar cada mensagem. Dos documentos que contém número de palavras maior do que o escolhido selecionam-se os n primeiros tokens. Nos casos em que a mensagem é menor, ela é completada com tokens de preenchimento, os quais são alocados na origem do espaço do *embedding*. O número n é escolhido com

base nos dados disponíveis de treinamento, de maneira a não se desperdiçar grande quantidade de informação nem se fazer uso excessivo dos tokens.

A principal diferença na utilização de redes convolucionais em texto é que neste caso as convoluções e *poolings* são operadas em apenas uma dimensão, a dimensão que representa a posição de cada palavra na mensagem. Desta maneira, cada filtro se desloca apenas entre as janelas de tokens, englobando toda a representação do *embedding* de cada token.

4.4 Funções Custo

Dá-se o nome de função custo às funções que caracterizam a distância entre o resultado obtido pela rede e seu objetivo. Assim, as sinapses de redes neurais são otimizadas a partir da minimização de tais funções. Nesta seção, serão apresentadas as funções custo mais relevantes no contexto de redes neurais.

4.4.1 Erro Médio Quadrático

A função custo mais utilizada é o erro médio quadrático (EMQ). Seu fator quadrático pune mais severamente grandes erros, acelerando o processo de otimização.

A fórmula 4.11 mostra a composição do EMQ sendo n o número de pares de exemplos e \hat{y} a predição da rede neural.

$$\frac{\sum^n |\hat{y}_i - y_i|^2}{n} \quad (4.11)$$

4.4.2 Erro Médio Absoluto

O erro médio absoluto se assemelha ao EMQ, tendo como diferença a falta do fator quadrático, conforme observa-se na fórmula 4.12.

$$\frac{\sum^n |\hat{y}_i - y_i|}{n} \quad (4.12)$$

4.4.3 Erro Médio Absoluto Percentual

Há situações em que se pretende mapear um objetivo com extensão de diferentes ordens de grandeza. O uso de erro médio quadrático ou absoluto nestes casos irá dar menos relevância a entradas pequenas. Para solucionar este problema, o erro médio absoluto percentual é composto conforme apresentado na fórmula 4.13

$$\frac{100}{n} \sum^n \left| \frac{\hat{\mathbf{y}}_i - \mathbf{y}_i}{\mathbf{y}_i} \right| \quad (4.13)$$

4.4.4 Entropia Cruzada

A entropia cruzada é uma medida que contém a distância entre duas distribuições de probabilidades. Sua formula é disposta de maneira que sua otimização encontre os parâmetros de máxima verossimilhança. As equações 4.14 e 4.15 mostram a dedução da função de verossimilhança \mathcal{L} a partir da probabilidade a posteriori, na qual k representa cada classe presente no conjunto de dados.

$$\mathcal{L}(\theta) = \prod_k P(\mathbf{y}_k | \mathbf{x}) \quad (4.14)$$

$$\mathcal{L}(\theta) = \prod_k \hat{\mathbf{y}}_k^{\mathbf{y}_k} (1 - \hat{\mathbf{y}}_k)^{(1-\mathbf{y}_k)} \quad (4.15)$$

Para simplificar o custo computacional, o valor máximo da função verossimilhança é obtido minimizando-se o negativo de seu logaritmo, como presente na fórmula 4.16.

$$-\ln(\mathcal{L}(\theta)) = - \sum_k \mathbf{y}_k \ln(\hat{\mathbf{y}}_k) + (1 - \mathbf{y}_k) \ln(1 - \hat{\mathbf{y}}_k) \quad (4.16)$$

A expressão 4.17 mostra o uso da entropia cruzada como função custo. A segunda parte do somatório determina que a acurácia de não classe pode ser ignorada para determinada classe k , visto que esta já será considerada na sua classe correta. É necessário notar que a sua utilização depende de \mathbf{y}_i e $\hat{\mathbf{y}}_i$ serem limitados entre 0 e 1. Para adequar-se a esta limitação, caso a rede neural possua apenas uma saída, utiliza-se ativação Sigmoid na última camada. Em caso de múltiplas saídas, utiliza-se ativação Softmax.

$$\frac{-\sum_k^n y_k \ln(\hat{y}_k)}{n} \quad (4.17)$$

4.5 Otimizadores

Definidas as funções custo, nesta seção serão abordados os algoritmos responsáveis por sua minimização. Embora a área de otimização apresente uma vasta gama de opções, sua utilização em aprendizado de máquina costuma se restringir a algoritmos de primeira ordem, como o gradiente descendente, principalmente por seu menor custo computacional. O método do gradiente, primeiramente apresentado por Cauchy em 1847 [53], consiste em minimizar a função custo por meio de atualizações iterativas de parâmetros do modelo, a partir da derivada da própria função custo em relação aos parâmetros.

Redes neurais, em geral, possuem funções custo não convexas. Assim sendo, o algoritmo de gradiente descendente não garante convergência ao mínimo global. A convergência a mínimos locais torna o modelo sensível a parâmetros, como funções de ativação, número de neurônios, etc., e o torna sensível também à inicialização dos pesos, como mostra Glorot no artigo em que apresenta um método de inicialização de pesos homônimo [54]. Entretanto, o problema de convergência não se limita a mínimos locais. Dauphin [55] mostra que quanto maior o número de variáveis latentes do modelo, maior a proporção de pontos de sela em relação a mínimos locais. Mostra também que os mínimos locais estão mais propensos a aparecer em pontos críticos com baixo custo, enquanto pontos de sela costumam aparecer em pontos de alto custo. Goodfellow [56] apresenta resultados empíricos da efetividade do método do gradiente na fuga dos pontos de sela.

O Algoritmo 4.1 descreve a forma tradicional do gradiente descendente, dado que L é a função custo adotada e que $\hat{\mathbf{y}}$ é o mapeamento obtido pela rede neural a partir dos dados \mathbf{x} e pesos $\boldsymbol{\theta}$. Observa-se que $\hat{\mathbf{g}}$ é a estimativa do gradiente dos pesos obtida a partir dos dados de treinamento.

Serão apresentadas nas próximas subseções as variações do método do gradiente mais notórias no treinamento de redes neurais.

Algoritmo 4.1 Gradiente Descendente

Parâmetro Taxa de treinamento ϵ

Parâmetro Pesos iniciais $\boldsymbol{\theta}_0$

Parâmetro Número de iterações T

```
1:  $t \leftarrow 0$ 
2: while  $t < T$  do
3:    $t \leftarrow t + 1$ 
4:    $\hat{\mathbf{y}} \leftarrow f(\mathbf{x}, \boldsymbol{\theta}_{t-1})$ 
5:    $\hat{\mathbf{g}} \leftarrow \frac{1}{n} \nabla_{\boldsymbol{\theta}} \sum_i^n L(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$ 
6:    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \epsilon \hat{\mathbf{g}}$ 
7: end while
```

4.5.1 Gradiente Descendente Estocástico

A forma canônica do algoritmo do gradiente descendente começa a apresentar dificuldades de utilização em conjuntos de dados de tamanho considerável. Como observa-se no Algoritmo 4.1, para cada atualização dos pesos $\boldsymbol{\theta}$ é necessária a estimação do gradiente $\hat{\mathbf{g}}$, a qual, por sua vez, depende da computação da função custo sobre todos os dados. Além do alto custo computacional, bases de dados grandes tendem a conter várias instâncias de dados muito próximos, não adicionando significativamente informação à estimação do gradiente.

Uma das soluções deste problema, abordada pelo *Gradiente Descendente Estocástico*, é a divisão do conjunto de treinamento em lotes, nos quais serão estimados os gradientes. Cada interação sobre o conjunto inteiro de dados é chamada de época e entre cada época costuma-se embaralhar os dados. Já as outras partes do algoritmo permanecem iguais à implementação tradicional como apresentado em 4.1.

4.5.2 Gradiente Descendente com Momento

A escolha da taxa de treinamento envolve uma troca entre tempo de treinamento e grau de convergência. Altas taxas de treinamento podem apresentar dificuldades de convergência ou comportamentos oscilatórios em torno de um ponto de custo mínimo, enquanto baixas taxas de treinamento consomem muito tempo de treinamento em regiões de alto custo, diminuindo assim sua viabilidade. Objetivando-se

obter um melhor compromisso, foi acrescentado o momento ao método do gradiente, como formulado no Algoritmo 4.2. O termo adicional \mathbf{v} representa a "velocidade" do gradiente e a constante de momento α condiz com a taxa de decaimento exponencial do termo \mathbf{v} .

Algoritmo 4.2 Gradiente Descendente com Momento

Parâmetro Taxa de treinamento ϵ

Parâmetro Constante de momento α

Parâmetro Pesos iniciais $\boldsymbol{\theta}_0$

Parâmetro Número de iterações T

```

1:  $t \leftarrow 0$ 
2:  $\mathbf{v}_0 \leftarrow \vec{0}$ 
3: while  $t < T$  do
4:    $t \leftarrow t + 1$ 
5:    $\hat{\mathbf{y}} \leftarrow f(\mathbf{x}, \boldsymbol{\theta}_{t-1})$ 
6:    $\hat{\mathbf{g}} \leftarrow \frac{1}{n} \nabla_{\boldsymbol{\theta}} \sum_i^n L(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$ 
7:    $\mathbf{v}_t \leftarrow \alpha \mathbf{v}_{t-1} - \epsilon \hat{\mathbf{g}}$ 
8:    $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} + \mathbf{v}_t$ 
9: end while

```

O momento permite que, enquanto o gradiente $\hat{\mathbf{g}}$ mantiver o mesmo sinal entre iterações, o módulo da velocidade \mathbf{v} cresça, acelerando o processo de aprendizado. Quando o sinal do gradiente for invertido, ou seja, passar de um ponto de custo mínimo, o módulo da velocidade será reduzido até seu sentido ser revertido, voltando ao mínimo. Segundo Haykin [57], o uso do momento tem um efeito estabilizador no algoritmo de aprendizado.

4.5.3 Adam

Outra abordagem adotada nos algoritmos de otimização é o ajuste da taxa de aprendizado de maneira independente para cada dimensão do espaço. Duchi *et al.* [58] propuseram o algoritmo *AdaGrad* (*Adaptative Gradient*), no qual a regra de atualização dos pesos é inversamente proporcional à soma do gradiente ao quadrado em todas iterações anteriores. Dessa forma, direções que tenham grandes e frequentes atualizações serão amortecidas rapidamente, enquanto dimensões

com raras atualizações mantêm seu treinamento significativo por mais iterações. Algoritmos com taxa de treinamento adaptativo, como o proposto por Duchi *et al.*, atingem bons resultados em problemas com dimensões esparsas, como o exemplo de Pennington *et al.* [28] no treinamento de *embeddings* de palavras, técnica que será apresentada na Seção 3.2.3.

Kingma e Ba [59] projetaram o algoritmo *Adam*, *adaptive moment estimation* baseado nos princípios do *AdaGrad*. *Adam* estima o primeiro e segundo momentos do gradiente, média e variância não centralizada, a partir da média móvel de ambos. A regra de atualização de pesos é controlada pela taxa de treinamento ϵ , e pelas estimativas de média e variância, $\hat{\mathbf{m}}$ e $\hat{\mathbf{v}}$ respectivamente, dada a fórmula $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \epsilon \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}}$. Portanto, gradientes grandes por seguidas iterações aumentam a média e, por consequência, o tamanho do passo. Por sua vez, altas variâncias forçam a diminuição da taxa efetiva de aprendizado.

O Algoritmo 4.3 possui a implementação de *Adam* em pseudocódigo. Os valores dos decaimentos betas costumam ser escolhidos em torno de 0.9 para β_1 e 0.999 para β_2 . Pelos momentos serem inicializados em zero, é necessário corrigir a tendência para se obter a verdadeira estimativa.

Adam apresenta robustez à escolha da taxa de treinamento ϵ , visto que esta funciona como um limite superior da taxa efetiva de atualização dos pesos, como provado em seu artigo [59], juntamente com sua análise de convergência e com seus resultados experimentais. Por se utilizar de médias móveis na estimação dos momentos, o algoritmo apresenta outra vantagem, sua capacidade de se adaptar a funções objetivos não estacionárias.

Por sua robustez à escolha de hiperparâmetros e por sua aceleração do processo de aprendizado *Adam* se tornou uma das principais técnicas de otimização de redes neurais profundas.

4.6 Generalização

Um dos principais fatores de um sistema de aprendizado de máquina é sua capacidade de generalizar, isto é, quando seu desempenho em dados de teste, ou seja, dados que não foram utilizados na fase de treinamento, é equivalente ou sufi-

Algoritmo 4.3 Adam

Parâmetro Taxa de treinamento ϵ

Parâmetro Constante de decaimento β_1, β_2

Parâmetro Pesos iniciais θ_0

Parâmetro Número de iterações T

```
1:  $t \leftarrow 0$ 
2:  $\mathbf{m}_0 \leftarrow \vec{0}$ 
3:  $\mathbf{v}_0 \leftarrow \vec{0}$ 
4: while  $t < T$  do
5:    $t \leftarrow t + 1$ 
6:    $\hat{\mathbf{y}} \leftarrow f(\mathbf{x}, \theta_{t-1})$ 
7:    $\hat{\mathbf{g}} \leftarrow \frac{1}{n} \nabla_{\theta} \sum_i^n L(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$ 
8:    $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \hat{\mathbf{g}}$ 
9:    $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \hat{\mathbf{g}}^2$ 
10:   $\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{(1 - \beta_1^t)}$ 
11:   $\hat{\mathbf{v}}_t \leftarrow \frac{\mathbf{v}_t}{(1 - \beta_2^t)}$ 
12:   $\theta_t \leftarrow \theta_{t-1} - \epsilon \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}}$ 
13: end while
```

cientemente próximo ao obtido durante aprendizado.

A fase de aprendizado do sistema pode ser vista como o processo de ajuste de parâmetros. Desta forma, ao tentar representar a função, o modelo pode sofrer com sobreajuste, *overfitting*, adequando-se demais aos dados de treinamento. O oposto também acontece quando o modelo é incapaz de mapear a complexidade entre as entradas e a saída; a este evento dá-se o nome de subajuste, *underfit*. Tais eventos estão exemplificados na Figura 4.6.

O uso de modelos robustos como os de *Deep Learning* implica em uma maior propensão de se obter *overfitting* durante o treinamento, resultado de seu crescente número de parâmetros livres. Abordando esse problema, foi desenvolvido um conjunto de técnicas que visam a reduzir o *overfitting*, conjunto esse que recebe o nome de regularização. Nas subseções seguintes serão apresentadas as duas principais técnicas de regularização.

4.6.1 Regularização de Tikhonov

As regularizações de norma compõem um dos grupos mais simples de regularização. Seu funcionamento é possível pois se adiciona à função custo um termo proporcional aos pesos, penalizando neurônios com grandes normas. Assim, limita-se a capacidade do modelo e, conseqüentemente, diminui-se o potencial de *overfitting*.

Também conhecida como regularização L_2 , a regularização de Tikhonov utiliza-se da norma quadrática dos pesos. A formula 4.18 exemplifica a função custo erro médio quadrático com o termo adicional de regularização L_2 , que por sua vez é controlado por uma constante λ .

$$\frac{\sum^n |\hat{\mathbf{y}}_i - \mathbf{y}_i|^2}{n} + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 \quad (4.18)$$

Krogh e Hertz [60] provam que o uso de regularização L_2 é capaz de suprimir parâmetros irrelevantes, diminuindo o número de parâmetros do modelo, e de reduzir o efeito de ruídos presentes na anotação dos dados.

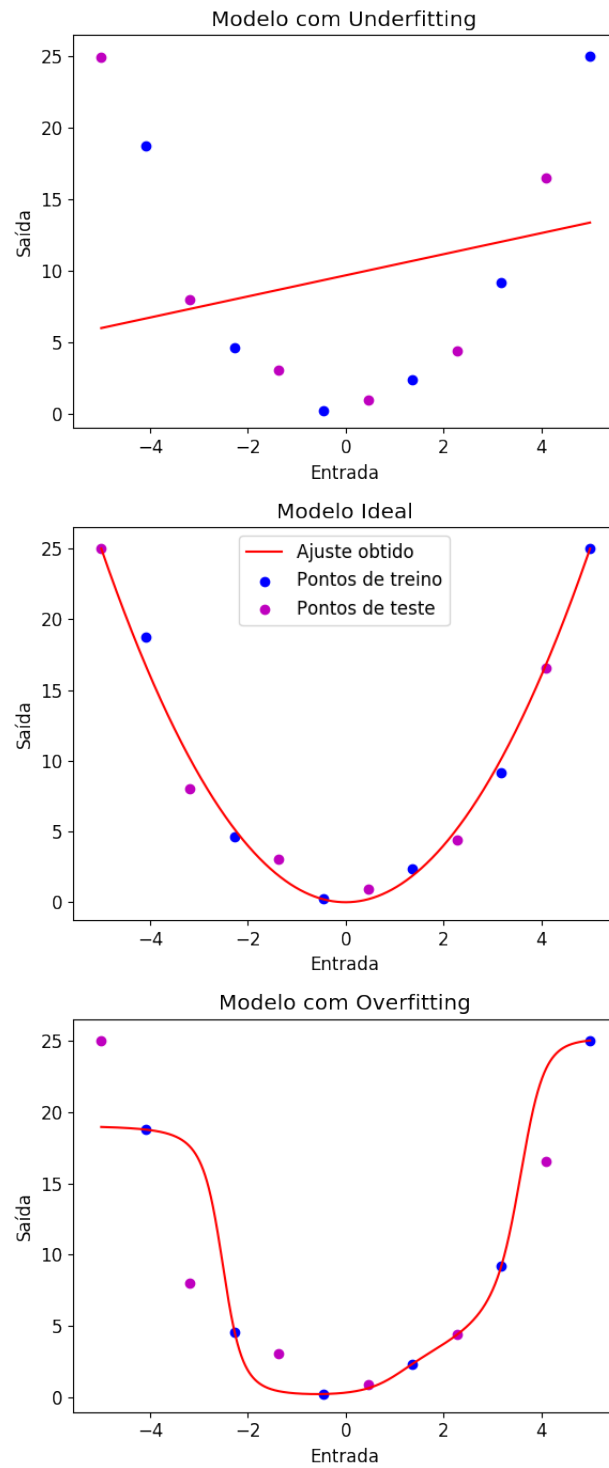


Figura 4.6: Diferentes pontos de ajuste dos algoritmos de aprendizado de máquina.

4.6.2 Dropout

Apresentado por Srivastava *et al.* [61], *Dropout* é um método que propõe reduzir o *overfitting* por meio do desligamento temporário e aleatório de neurônios durante a etapa de treinamento. Esse processo dá-se pela escolha de um parâmetro fixo p que define a probabilidade de manutenção de cada neurônio, a partir desse ponto se desconectam os neurônios sorteados, efetivamente multiplicando seu pesos por 0, resultando em uma rede neural composta por um subconjunto de neurônios da rede original os quais serão treinados com *Backpropagation* como habitual. Os sorteios ocorrem a cada seleção de lote do algoritmo do gradiente, levando a se treinar um número exponencial de redes neurais durante a fase de aprendizado. *Dropout* pode ser aplicado tanto em neurônios de camadas escondidas como em neurônios de entrada, valores de p são costumeiramente escolhidos por volta de 0,5 para camadas escondidas e 0,8 para camadas de entrada.

Dropout se deriva de um conjunto de técnicas de combinação de modelos, também conhecidas como *ensembling*. Como mostra Dietterich *et al.* [62], modelos significativamente diferentes quando combinados costumam resultar em melhoras na performance quando comparados à utilização de cada modelo isoladamente. No escopo de *Deep Learning* o custo computacional do treinamento de múltiplos modelos reduz a viabilidade desse processo. Neste sentido, *Dropout* aproxima-se de técnicas tradicionais de *ensembling*, como *bagging*, tendo como principal diferença o compartilhamento de pesos entre as redes neurais obtidas pelos desligamentos de neurônios, que permite o treinamento de um grande número de modelos sem um aumento computacional proporcional. Ressaltando-se que o apagamento de neurônios resulta em redes de tamanho reduzido, diminuindo assim seu poder de mapeamento, logo, a utilização dessa técnica exige um número superior de parâmetros para uma mesma capacidade de mapeamento.

Essa ferramenta se inspira na reprodução sexuada, que envolve trocas entre genes de dois indivíduos distintos, Srivastava *et al.* acreditam que assim como na evolução biológica, técnicas que reduzam a co-adaptação entre neurônios tornam redes neurais mais resistentes a destruição de informação, fazendo com que cada neurônio extraia características relevantes por si só, aumentando a redundância e melhorando assim o desempenho da rede como um todo.

A utilização de modelos com *Dropout* na fase de inferência requer um cuidado adicional. Por não haver desligamento nessa fase é necessário normalizar o valor esperado de cada neurônio para que este seja igual ao do período de treinamento. Há diferentes métodos, sendo a abordagem mais simples a multiplicação de cada neurônio por sua probabilidade de manutenção p .

4.7 Supervisão Distante

O sucesso de algoritmos de *Deep Learning* depende da disponibilidade de grandes volumes de dados de treinamento. Apesar da crescente produção de dados, para muitos problemas o processo de anotação é uma tarefa manual, tornando-a custosa e limitando assim o conjunto de dados supervisionados.

Supervisão Distante visa contornar essa dificuldade extraíndo anotação de maneira automatizada. Essa técnica consiste encontrar uma característica que tenha forte correlação com a saída desejada e considerá-la como uma anotação ruidosa [1].

Esse procedimento obteve êxito em algumas aplicações de processamento de linguagem natural [63][1] pela abundância de informações presentes de forma desestruturada. Craven *et al.* [63] apresenta uma das primeiras aplicações da técnica extraíndo a partir de registros textuais informações sobre proteínas e suas relações.

Capítulo 5

Métodos

Este capítulo visa descrever a elaboração de um modelo de *deep learning* capaz de classificar sentimento, positivo ou negativo, de *tweets*. A produção desse modelo será feita a partir de base de dados anotada de maneira automatizada, permitindo a sua fácil replicação.

5.1 Bases de Dados

O trabalho Sentiment140, desenvolvido por Go *et al.* [1] visa definir um sistema de classificação de sentimento sem a necessidade de anotação manual dos dados para treinamento, utilizando a técnica de supervisão distante. Conjuntamente com o artigo, Go *et al.* disponibilizaram um base de dados formada por anotação ruidosa composta de 800 mil *tweets*, divididos igualmente entre as classes positivas e negativas, e uma pequena base de testes de 360 *tweets* anotados manualmente, também divididos igualmente entre as classes.

A não dependência de anotação dos dados e a facilidade de coleta dos *tweets*, por meio de interface programável, viabilizam a formação de uma grande base de treinamento. Para a realização deste trabalho foram coletados cerca de 40 milhões de mensagens do Twitter filtrando-as apenas pelo idioma inglês. Posteriormente, se aplica o método de classificação ruidosa de sentimento desenvolvido por Read [64] como especificado na Seção 4.7 nos *tweets* coletados. Nesse método, são definidos grupos de *emoticons* positivos e negativos. Mensagens que possuírem *emoticons* pertencentes a algum destes grupos serão anotadas com as respectivas classes, caso

emoticons de ambos os grupos estejam presentes em uma mesma mensagem esta será descartada. Os *emoticons* utilizados para anotação da base de dados foram removidos das mensagens para evitar introduzir viés ao classificador.

A tabela 5.1 mostra os *emoticons* escolhidos para compor a anotação ruidosa. Em adição aos *emoticons* presentes no Sentment140, foram classificados manualmente os emoticons mais presentes na base de dados. A escolha dos *emoticons* definirá quão ruidosa será a anotação automática. Durante este processo, observou-se um grande desbalanceamento nas classes, na qual os *tweets* positivos compõem 80% do volume total de anotações.

















Negativos	Positivos
: (:)
= (=)
: - (: -)
: ' (: D
	
	
	
	
	
	
	
	
	
	
	

Tabela 5.1: Emoticons selecionados para aplicação de supervisão distante.

A base de teste por sua vez é composta de *tweets* anotados manualmente. Essa base é formada pela coletânea de dados disponibilizados pela conferencia anual *Semantic Evaluation* (SemEval) [65] entre os anos de 2013 a 2017. Esta base é composta de cerca de 70 mil *tweets* e processo de anotação destes dados foi feito manualmente através da plataforma CrowdFlower, mais detalhes sobre a formação

da base de dados são apresentados por Rosenthal *et al.* [66], organizadores do evento. Novamente foram removidos os *emoticons* presentes na anotação ruidosa.

5.2 Métricas

Uma vez definidos os dados e as técnicas de classificação, se faz necessário medir a qualidade dos modelos obtidos. Esta seção apresentará as métricas utilizadas para seleção de parâmetros e avaliação dos algoritmos.

A forma mais direta de se medir a performance de um modelo é considerar a sua acurácia. Todavia, a presença de desbalanceamento de classes nos dados pode favorecer a acurácia da classe mais presente enquanto perde eficiência na classificação da classe com menos exemplos.

A curva ROC (*receiver operating characteristic*) [67] é uma forma de visualizar a performance de um classificador binário para seleção de um ponto de operação. Nela se apresenta a variação da probabilidade de detecção e de falso alarme variam com a escolha do patamar de decisão. A área sob a curva (AUC) pode ser utilizada como métrica de eficiência para um classificador.

O índice SP é outro método de avaliação de classificadores binários. A equação 5.1 descreve a formação desse índice na qual P_c representa a probabilidade de detecção da classe positiva enquanto P_{nc} caracteriza a probabilidade de não obtenção de falso alarme [68].

$$SP = \sqrt{\sqrt{P_c P_{nc}} \left(\frac{P_c + P_{nc}}{2} \right)} \quad (5.1)$$

O patamar de decisão do classificador será escolhido de maneira a maximizar o índice SP.

Uma maneira de garantir que os resultados obtidos pelas métricas apresentadas sejam consistentes independente do conjunto de dados na qual será aplicado é utilizar a validação cruzada. Neste trabalho será utilizado o método de validação cruzada de K-partições. Este processo [69] consiste na separação aleatória da base de dados em k conjuntos mutualmente exclusivos de tamanhos similares. Posteriormente são executadas k rodadas, na qual o conjunto correspondente a cada rodada será utilizado para validação do modelo enquanto os outros $k - 1$ conjuntos servirão

para seu treinamento. O treinamento pode ser considerado consistente se obtiver pouca variação nos resultados das k rodadas.

5.3 Desenvolvimento

O presente trabalho será dividido em três etapas. Na primeira fase utilizou-se a base de dados provida por Go *et al.* [1] replicando as técnicas abordadas em seu artigo visando validar os pré-processamentos e algoritmos aplicados. Para tal, serão utilizadas as bases de dados tanto de treinamento como de testes disponibilizadas no Sentiment140 [1].

Começou-se tokenizando os *tweets*, durante esse processo removeram-se *stopwords*; links; referências a usuários; e cada token foi transformado para forma minúscula. Replicou-se como entrada do algoritmo de aprendizado apenas a representação dos tokens por unigrama, essa escolha foi feita por ser a representação mais simples e por Go *et al.* [1] mostrar que há pouca variação de resultado entre as diferentes representações como apresentado na Tabela 5.2.

Representação	Naïve Bayes	Máxima Entropia	SVM
Unigrama	81,3%	80,5%	82,2%
Bigrama	82,6%	79,1%	78,8%
Unigrama + Bigrama	82,7%	83,0%	81,6%
Unigrama + <i>Part of Speech</i>	82,7%	83,0%	81,6%

Tabela 5.2: Acurácia obtida pelos classificadores do Sentiment140.

Em seguida, foram treinados os algoritmos de Naïve Bayes e SVM, estes foram escolhidos por existirem implementações de código aberto disponíveis e de fácil acesso. Utilizou-se Naïve Bayes com distribuição multinomial, por se adequar ao pré processamento utilizado. Variou-se o parâmetro de suavização de distribuição de laplace, de maneira a otimizar a acurácia. Para treinamento do modelo por *support vector machines*, dada a grande quantidade de dados e as limitações computacionais, foi empregado o treinamento a partir do método do gradiente como descrito por Suykens e Vandewalle [35]. A função *kernel* utilizada foi linear. Neste caso, variou-se parâmetro de regularização L2 também visando maximizar a acurácia.

Empregou-se validação cruzada por K-partições, com 10 partições, no treinamento dos dois algoritmos. Findada a seleção dos modelos, se avaliou o desempenho dos classificadores comparados aos resultados apresentados por Go *et al.*

A segunda etapa consiste em treinar os mesmos algoritmos de Naïve Bayes e SVM utilizados anteriormente porém utilizando a base de dados anotada desenvolvida com supervisão distante, como elucidado na Seção 5.1. Os resultados obtidos nesta fase servirão como base de comparação entre os modelos lineares e modelos formados técnicas de *deep learning*. Adicionalmente, será validado o processo de captação de dados e anotação por supervisão distante. Nesta fase replicou-se os mesmos procedimentos práticos da etapa anterior.

Entretanto, devido a base de dados coletada ser desbalanceada, durante o treinamento do modelo por *support vector machines* foi necessário dar pesos diferentes a cada exemplo de maneira que cada classe pesasse igualmente durante o treinamento. Tal correção não se faz necessária no modelo por Naïve Bayes pois este considera a probabilidade de presença das classes, como descrito na Seção 4.1. Por sua vez, a métrica utilizada para seleção do modelo neste caso será a área sob a curva ROC visto que a seleção por acurácia favorece modelos enviesados a classe mais presente nos dados. Uma vez selecionados, os modelos foram analisados com base em sua performance na classificação das bases de testes tanto do Sentiment140 quanto do SemEval.

Por fim, a terceira e última etapa é formada pela aplicação de redes convolucionais em textos como descrito por Kim [2]. Nesta fase será medido o impacto da utilização desta técnica na classificação de mensagens. Serão variados diferentes pré-processamentos e parâmetros das redes para analisar sua influência na eficiência do modelo.

Para utilização de modelos de redes convolucionais, precisa-se primeiro representar os dados por um *embedding*. Neste sentido, utilizou-se o *embedding* Word2Vec pré-treinado a partir de notícias em inglês, disponibilizado pelo Google. Este Word2Vec é treinado de maneira a representar cada token como um vetor de 300 dimensões.

Posteriormente, geraram-se modelos de redes convolucionais nos quais a representação obtida pelo Word2Vec foi utilizada como entrada da rede. A escolha dos hiperparâmetros a serem testados foi baseada na submissão ao SemEval de De-

rius *et al.* [70], a qual obteve o melhor resultado da SemEval 2016. Compõem os hiperparâmetros variados: número de camadas, número de filtros convolucionais por camada, tamanho dos filtros convolucionais, tamanho do filtro de *pooling*. Os parâmetros de regularização L2 e probabilidade de Dropout se mantiveram fixos, sendo seus valores também iguais aos propostos por Derius *et al.*

Para o treinamento das redes convolucionais sortearam-se os dados anotados de maneira ruidosa em dois grupos: treinamento, contendo 80% do volume total, e validação com os 20% restante. Não se aplicou a validação cruzada, como nas etapas anteriores, visto o alto custo computacional de treinamento de cada rede. O treinamento se deu até que o valor da função custo sob os dados de treinamento se estabiliza-se em um valor arbitrariamente pequeno, ϵ . Entretanto, para reduzir o efeito do *overfitting*, foi aplicado *early stopping* [71], técnica que consiste em utilizar os pesos obtidos na época de treinamento que corresponde ao menor valor de função custo aplicada ao conjunto de validação. A função custo aplicada foi a entropia cruzada e utilizou-se o otimizador por método do gradiente *Adam*.

Assim como aplicado na modelagem por SVM na segunda etapa, foi necessário compensar o desbalanceamento das classes através de pesos diferentes para cada classe. A seleção do classificador de melhor performance foi feita a partir daquele que obteve maior valor de área sob a curva ROC. Por fim, se verificou se a utilização de redes convolucionais se desempenhou como o esperado na análise de sentimento de *tweets*.

Capítulo 6

Resultados

Este capítulo tem como objetivo apresentar os resultados obtidos pelos procedimentos descritos no Capítulo 5. Serão avaliados se os objetivos foram atingidos e serão ressaltados problemas encontrados no desenvolvimento.

6.1 Primeira Etapa

A primeira etapa consiste no treinamento de algoritmos de aprendizado de máquina a partir de base de dados anotada ruidosamente, replicando assim o trabalho de Go *et al.* Nesta fase foram utilizados os dados disponibilizados no Sentiment140 para treinamento e teste do algoritmo. Os dados de treinamento são compostos por supervisão distante, totalizando 800 mil exemplos, metade positivos e metade negativos. Entretanto, a base de teste dispõe apenas de 360 observações, também divididas igualmente entre as classes. O baixo número de exemplos pode causar pequenas variações nos resultados obtidos.

Para seleção de hiperparâmetros foram treinados modelos de Naïve Bayes com diferentes fatores de regularização por suavização de distribuição de laplace. O modelo selecionado foi o que obteve maior valor médio de acurácia entre 10 partições de validação cruzada. A Figura 6.1 mostra a resposta da acurácia dos diferentes modelos nos grupos de validação, as barras de erro correspondem a um desvio padrão. A linha vertical vermelha presente na Figura 6.1 indica o melhor parâmetro obtido, aproximadamente 4,1, resultando em uma acurácia média sobre o conjunto de validação de 78,2%. Aplicando-se o modelo de melhor desempenho no conjunto de

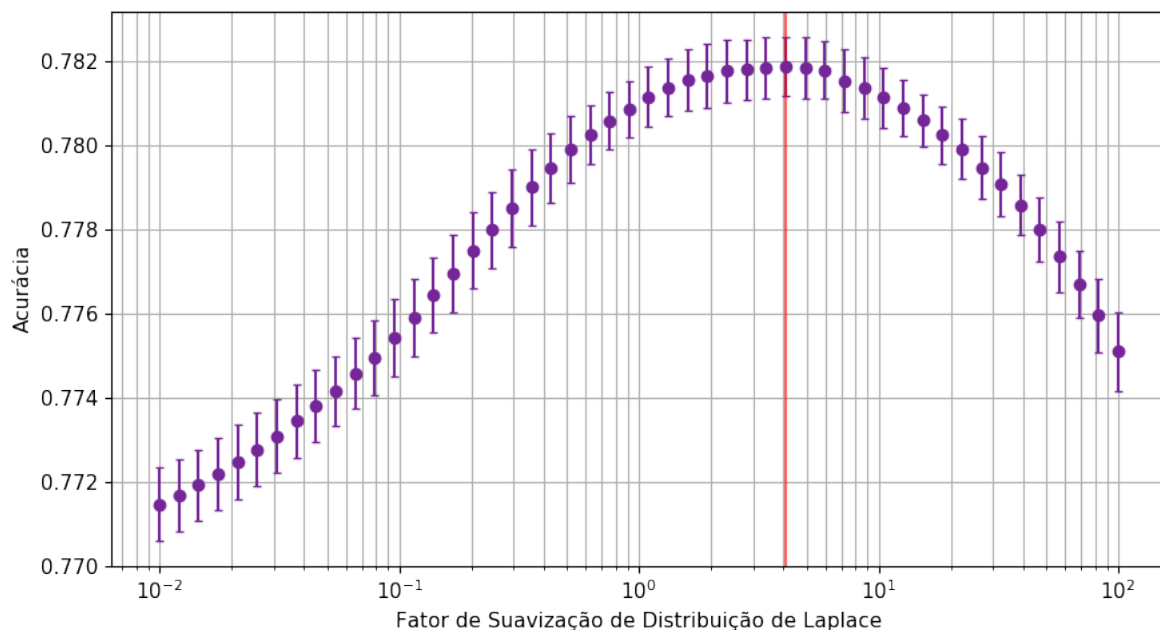


Figura 6.1: Seleção de hiperparâmetros de Naïve Bayes - Etapa 1.

testes foi obtida acurácia de 83,3%, este valor se aproxima ao obtido originalmente: 81,3%.

A seleção do parâmetros do modelo formado por *support vector machines* foi feita por processo semelhante ao anterior. Foram variados o fator de regularização L2, selecionando o modelo de maior percentual de acurácia. A Figura 6.2 apresenta os resultados obtidos, no qual o fator de 6×10^{-6} resultou na acurácia média de 80.0% sobre o conjunto de validação. O modelo selecionado apresenta 83,0% de acurácia no conjunto de testes.

A Tabela 6.1 apresenta os resultados obtidos tanto no Sentiment140 quanto pela replicação de seu método. Pode se observar que foram atingidos valores próximos a referência, validando assim os pré-processamentos e as implementações dos algoritmos aplicados.

6.2 Segunda Etapa

A segunda etapa consiste na validação do processo de formação de base de treinamento por supervisão distante. A aplicação de técnicas de pré-processamento e algoritmos previamente validados neste novo conjunto de dados visa tanto comparar

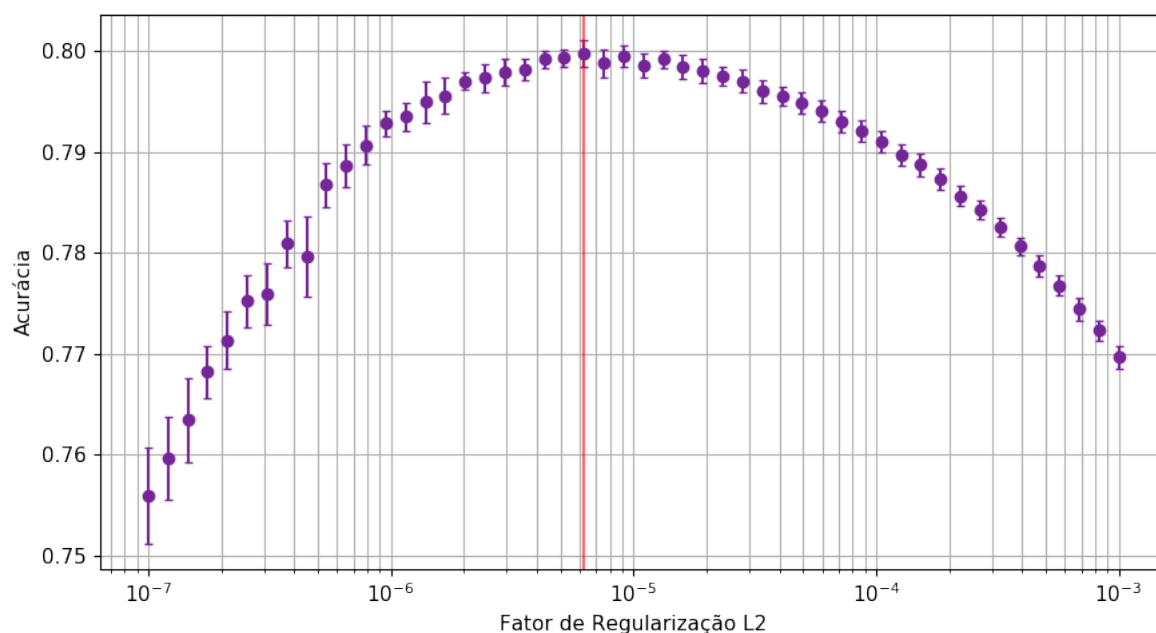


Figura 6.2: Seleção de hiperparâmetros de SVM - Etapa 1.

Algoritmo	Original	Replicação
Naïve Bayes	81,3%	83,3%
SVM	82,2%	83,0%

Tabela 6.1: Comparação de resultados da replicação dos classificadores do Senti-ment140.

o processo de elaboração por anotação ruidosa quanto servir como referência para a aplicação de algoritmos de *deep learning*. Nesta etapa, os modelos serão avaliados tanto pelo seu desempenho na base de teste disponibilizada por Go *et al.* quanto quando aplicado na base de testes formada pela coletânea de *tweets* oferecida pelas conferências SemEval, como descrito na Seção 5.1

Assim como na fase anterior, foram treinados modelos por Naïve Bayes e *support vector machines*. Neste caso, a seleção dos parâmetros de regularização foi feita de maneira a maximizar a área sob a curva ROC. As Figuras 6.3 e 6.4 mostram a resposta dos modelos, respectivamente, Naïve Bayes e SVM a mudanças nos parâmetros de regularização, as linhas verticais vermelhas apontam os parâmetros que obtiveram maior média de área sob a curva ROC dentre os grupos de validação

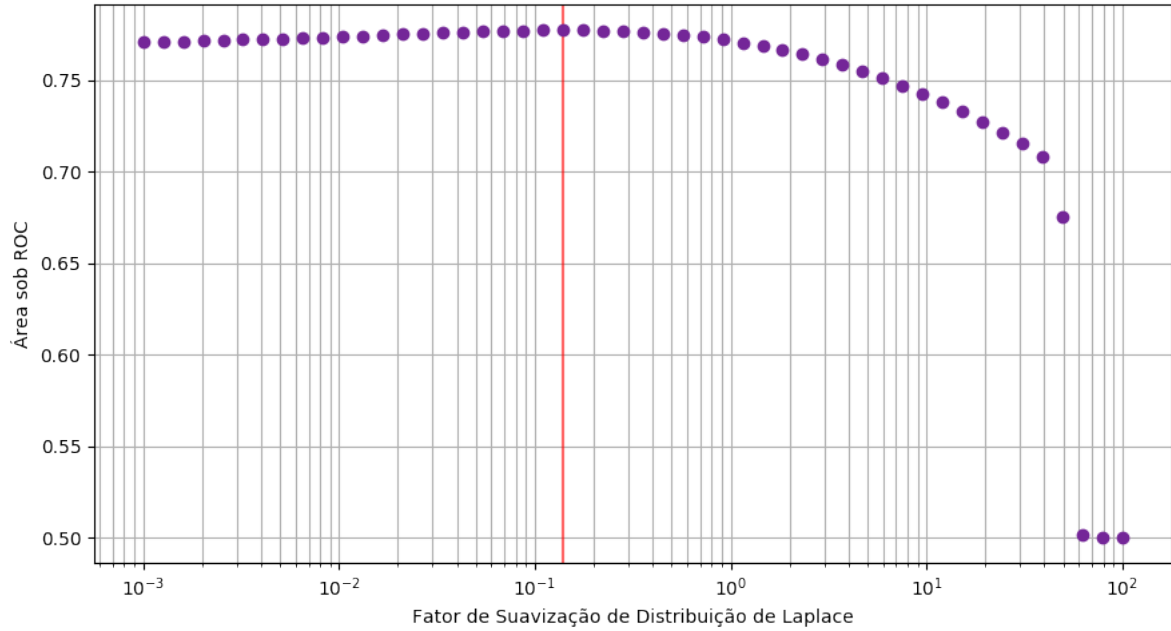


Figura 6.3: Seleção de hiperparâmetros de Naïve Bayes - Etapa 2.

quando aplicada validação cruzada com 10 partições.

Uma vez selecionado os melhores hiperparâmetros, a Figura 6.5 compara a curva ROC que caracteriza a performance dos modelos treinados tanto na primeira quanto na segunda etapa sob os dados de testes disponibilizados por Go *et al.* A Figura 6.6 por sua vez apresenta a curva ROC dos modelos quando aplicados nos dados manualmente anotados coletados do SemEval.

Observa-se que os modelos treinados pelo conjunto de dados de anotação ruidosa disponibilizados por Go *et al.* apresentou desempenho consideravelmente melhor nos dois conjuntos de testes. Desta maneira, se percebe que o processo de anotação dos dados foi mais ruidoso do que o apresentado pelo Sentiment140.

Uma hipótese a ser feita é que evoluções no idioma e na plataforma durante o intervalo entre a criação de ambas as bases anotadas por supervisão distante tenha dificultado este processo, visto que a desenvolvida por Go *et al.* foi coletada em 2009, e a base de treinamento coletada por este trabalho conta com *tweets* de 2017. Constatou-se, por exemplo, que a base desenvolvida neste trabalho apresentou 1,1 milhões de palavras únicas após o processo de tokenização, enquanto os dados de treinamento do Sentiment140 contam com 330 mil palavras únicas.

Outro fator relevante para definição de ruído no processo de anotação é a

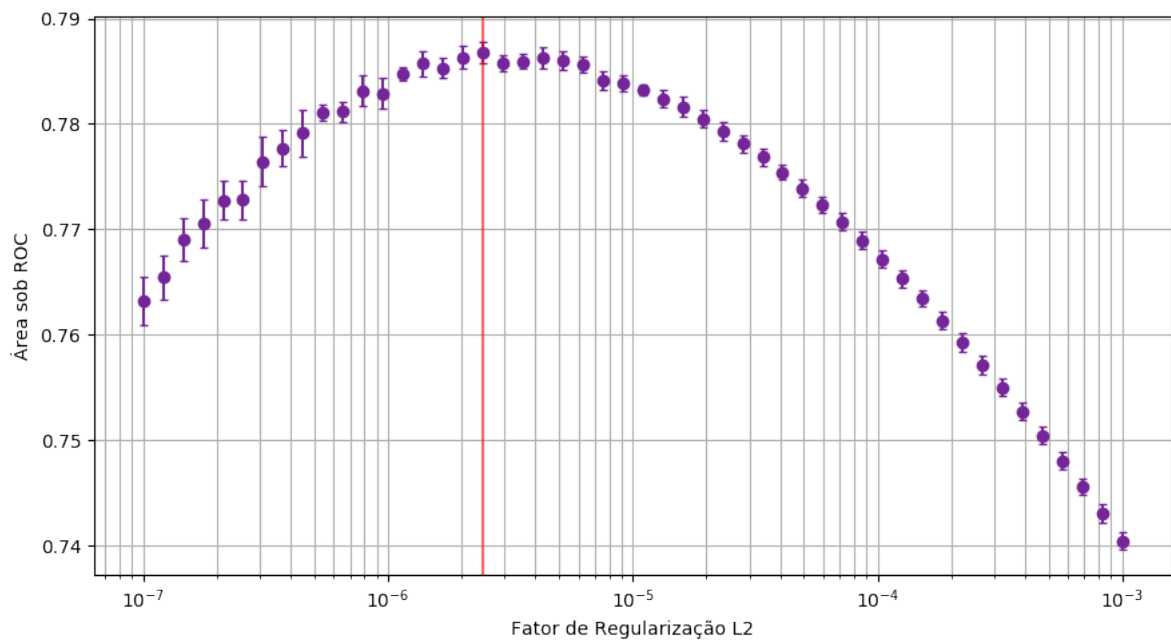


Figura 6.4: Seleção de hiperparâmetros de SVM - Etapa 2.

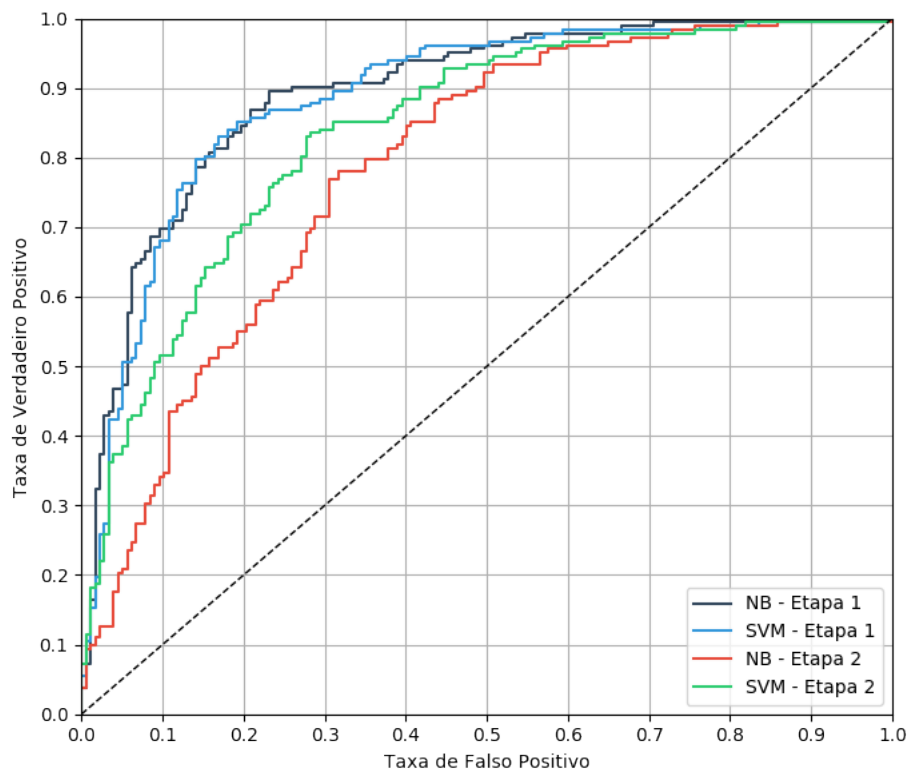


Figura 6.5: Curva ROC dos modelos aplicados aos dados de teste do Sentiment140.

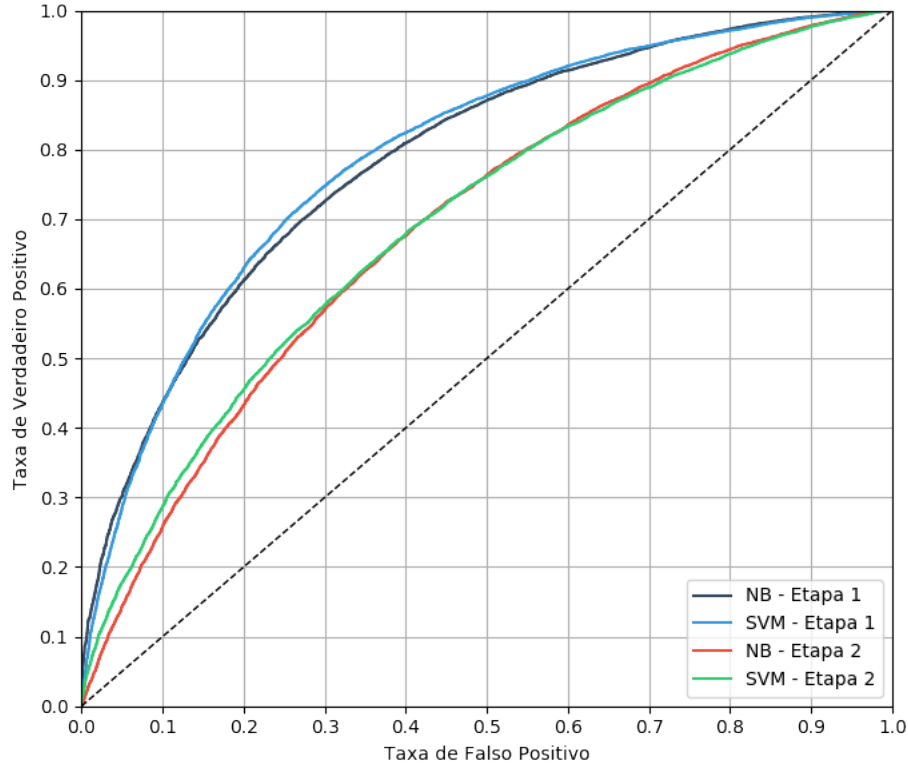


Figura 6.6: Curva ROC dos modelos aplicados aos dados de teste do SemEval.

escolha dos *emoticons*. Neste trabalho, além dos *emoticons* utilizados pelo Sentiment140, os *emoticons* mais presentes nos dados foram selecionados para classificação manual. A escolha de *emoticons* de maneira a atingir maior correlação com as classes pode ser uma solução para reduzir a diferença de performance dos modelos entre os resultados de treinamento e os resultados de teste.

Apesar dos resultados obtidos na base de treinamento anotada para este trabalho serem inferiores aos apresentados pelos classificadores treinados com a base disponibilizada por Go *et al.*, a técnica de supervisão distante por *emoticons* para anotação ruidosa de mensagens de redes sociais se mantém como boa alternativa ao processo custoso de anotação manual.

A Tabela 6.2 resume os resultados dos modelos treinados na primeira e na segunda etapas quando aplicados sob ambos os conjuntos de testes. O ponto de operação dos modelos foi escolhido de maneira a maximizar o índice SP. É válido ressaltar que a acurácia é uma métrica inconsistente quando consideras bases de testes com classes desbalanceadas, como é o caso da base de *tweets* coletados dos

SemEval, sendo a área sob a curva ROC, AUC, ou o índice SP, melhores opções para comparação destes casos.

Dados de teste	Modelo		Acurácia	AUC	SP
Sentiment140	Etapa 1	NB	83.3%	0.893	0.831
		SVM	83.0%	0.887	0.830
	Etapa 2	NB	73.3%	0.785	0.732
		SVM	77,7%	0,839	0,776
SemEval	Etapa 1	NB	70,9%	0,786	0,713
		SVM	72,8%	0,791	0,724
	Etapa 2	NB	64,9%	0,688	0,639
		SVM	64,2%	0,693	0,640

Tabela 6.2: Resultados obtidos pelos classificadores.

6.3 Terceira Etapa

A terceira fase do projeto visa a reproduzir redes neurais convolucionais aplicadas a texto e comparar seu desempenho a algoritmos de aprendizado de máquina consolidados no processamento de linguagem natural, como presentes na etapa anterior. Neste estágio, as redes serão treinadas com dados provindos da base anotada por supervisão distante, mesmos dados utilizados para treinamento dos algoritmos de Naïve Bayes e *support vector machines* da segunda etapa. Após feita a seleção dos hiperparâmetros, o modelo será avaliado aos treinados na segunda etapa com base no seu desempenho obtido na base de testes disponibilizada pelas conferências SemEval.

Foram variados diversos hiperparâmetros, são eles: número de camadas, número de filtros convolucionais por camada, tamanho dos filtros convolucionais, tamanho do filtro de *pooling*. Outros hiperparâmetros como: fator de regularização L2 e probabilidade de Dropout foram mantidos fixos em respectivamente 10^{-3} e 0,5. Os modelos foram treinados até que o valor da função custo estabilizar, não se modificando mais do que um valor arbitrariamente pequeno, ϵ , entre cada época de treinamento. O valor escolhido de ϵ foi 10^{-3} , o qual foi selecionado após a análise

de algumas curvas de treinamento. Por sua vez, aplicou-se *early stopping*, no qual os pesos da rede selecionados foram os correspondentes a época de menor valor de função custo aplicada ao conjunto de validação.

A escolha dos hiperparâmetros foi feita de maneira a maximizar a área sob a curva ROC nos dados de validação. A Tabela 6.3 lista os resultados obtidos por cada configuração treinada.

Hiperparâmetros				Resultado
Número de Camadas	Número Filtros Conv.	Tamanho Filtros Conv.	Tamanho Filtros Pooling	AUC
1	100	2	2	0.???
1	100	2	3	0.???
1	100	2	5	0.???
1	100	3	2	0.???
1	100	3	3	0.???
1	100	3	5	0.???
1	100	5	2	0.???
1	100	5	3	0.???
1	100	5	5	0.???
1	200	2	2	0.???
1	200	2	3	0.???
1	200	2	5	0.???
1	200	3	2	0.???
1	200	3	3	0.???
1	200	3	5	0.???
1	200	5	2	0.???
1	200	5	3	0.???
1	200	5	5	0.???
1	400	2	2	0.???
1	400	2	3	0.???

Hiperparâmetros				Resultado
Número de Camadas	Número Filtros Conv.	Tamanho Filtros Conv.	Tamanho Filtros Pooling	AUC
1	400	2	5	0.???
1	400	3	2	0.???
1	400	3	3	0.???
1	400	3	5	0.???
1	400	5	2	0.???
1	400	5	3	0.???
1	400	5	5	0.???
2	100	2	2	0.???
2	100	2	3	0.???
2	100	2	5	0.???
2	100	3	2	0.???
2	100	3	3	0.???
2	100	3	5	0.???
2	100	5	2	0.???
2	100	5	3	0.???
2	100	5	5	0.???
2	200	2	2	0.???
2	200	2	3	0.???
2	200	2	5	0.???
2	200	3	2	0.???
2	200	3	3	0.???
2	200	3	5	0.???
2	200	5	2	0.???
2	200	5	3	0.???
2	200	5	5	0.???
2	400	2	2	0.???

Hiperparâmetros				Resultado
Número de Camadas	Número Filtros Conv.	Tamanho Filtros Conv.	Tamanho Filtros Pooling	AUC
2	400	2	3	0.???
2	400	2	5	0.???
2	400	3	2	0.???
2	400	3	3	0.???
2	400	3	5	0.???
2	400	5	2	0.???
2	400	5	3	0.???
2	400	5	5	0.???

Tabela 6.3: Seleção de hiperparâmetros de CNN.

De acordo com a tabela, podemos observar que o conjunto de parâmetros com melhor desempenho no conjunto de validação foram: X camadas compostas por Y filtros convolucionais de tamanho Z seguidas de *pooling* com filtro de tamanho W. A Figura XYZ mostra a curva de treinamento deste modelo.

Uma vez selecionados os hiperparâmetros da rede neural convolucional, a Figura XYZ apresenta a curva ROC comparativa entre a rede e os modelos treinados na segunda etapa quando aplicados a base de testes formada por *tweets* fornecido pelas conferências SemEval. A Tabela XYZ resume os resultados destes classificadores, novamente considerando que o limiar de decisão foi selecionado de maneira a maximizar o índice SP.

Pode se observar que a utilização de redes neurais convolucionais em conjunto com representações obtidas por *embeddings* propiciou resultados melhores do que os obtidos com modelos tradicionais, como Naïve Bayes e *support vector machines* sobre representações *one-hot* dos textos. Ressalta-se que a representação Word2Vec utilizada foi desenvolvida a partir de notícias. O treinamento de modelos próprios para o meio no qual serão utilizados, neste caso, *tweets*, pode melhorar o desempenho do classificador a ser utilizado.

Capítulo 7

Conclusões

A massificação do uso das redes sociais gera uma crescente produção de dados. Entretanto, as informações que estão sendo geradas são dispostas de forma não estruturada, dificultando sua extração. Dado o volume de dados produzidos, se torna cada vez mais custoso realizar esse processo manualmente. Portanto, é fundamental o desenvolvimento de técnicas de processamento de linguagem natural capazes de auxiliar neste procedimento.

O presente trabalho teve como objetivo o desenvolvimento de um método de gerar classificadores de análise de sentimento para redes sociais sem dependência da anotação manual de bases de dados. Classificadores formados por algoritmos de aprendizado de máquina vem obtendo bons resultados na mineração de opinião. Todavia, estes dependem de dados de treinamento, os quais tem produção custosa visto que sua criação depende da anotação manual dos casos, dificultando, por exemplo, a reprodução destes classificadores para diferentes idiomas ou redes sociais.

Com o desenvolvimento de classificadores cada vez mais complexos, como os englobados pelas técnicas de *deep learning*, cresce a dependência de grandes volumes de dados de treinamento. O método da anotação ruidosa por supervisão distante, apesar de apresentar dificuldades decorrentes da seleção das características correlacionada com as classes, se mostrou eficiente na formação de bases de treinamento.

Observou-se que embora algoritmos de aprendizado de máquina tradicionalmente aplicadas ao processamento de linguagem natural atinjam resultados positivos nesta tarefa, a utilização de técnicas de *deep learning* é capaz elevar o nível de desempenho obtido.

Portanto, foi desenvolvido um método de produção de classificadores eficazes e que não dependem de anotação manual de dados de treinamento. Isto se fez possível pela utilização de representações de texto por algoritmos de aprendizado de máquina não supervisionados e por classificadores compostos por redes neurais convolucionais, treinadas a partir de dados anotados por supervisão distante.

7.1 Trabalhos Futuros

Referências Bibliográficas

- [1] GO, A., BHAYANI, R., HUANG, L., “Twitter sentiment classification using distant supervision”, *CS224N Project Report, Stanford*, v. 1, n. 12, 2009.
- [2] KIM, Y., “Convolutional neural networks for sentence classification”, *arXiv preprint arXiv:1408.5882*, , 2014.
- [3] SOCIAL, W. A., “Digital in 2017: Global Overview”, <https://wearesocial.com/special-reports/digital-in-2017-global-overview>, acessado em 10 de Outubro de 2017.
- [4] MOURTADA, R., SALEM, F., “Civil movements: The impact of Facebook and Twitter”, *Arab Social Media Report*, v. 1, n. 2, pp. 1–30, 2011.
- [5] YU, J., “A New Era Of Digital Content: Engagement And Content Metrics That Matter”, <https://marketingland.com/new-era-digital-content-engagement-content-metrics-matter-136147>, acessado em 5 de Janeiro de 2018.
- [6] NEWMAN, D., “Big Data And The Power Of Sentiment”, <https://www.forbes.com/sites/danielnewman/2016/05/10/big-data-and-the-power-of-sentiment/#715be21015a5>, acessado em 5 de Janeiro de 2018.
- [7] ALLEN, R., “What happens online in 60 seconds?”, <https://www.smartinsights.com/internet-marketing-statistics/happens-online-60-seconds/>, acessado em 5 de Janeiro de 2018.

- [8] CAMBRIA, E., SCHULLER, B., XIA, Y., *et al.*, “New avenues in opinion mining and sentiment analysis”, *IEEE Intelligent Systems*, v. 28, n. 2, pp. 15–21, 2013.
- [9] SOON, W. M., NG, H. T., LIM, D. C. Y., “A machine learning approach to coreference resolution of noun phrases”, *Computational linguistics*, v. 27, n. 4, pp. 521–544, 2001.
- [10] LAPPIN, S., LEASS, H. J., “An algorithm for pronominal anaphora resolution”, *Computational linguistics*, v. 20, n. 4, pp. 535–561, 1994.
- [11] NADEAU, D., SEKINE, S., “A survey of named entity recognition and classification”, *Linguisticae Investigationes*, v. 30, n. 1, pp. 3–26, 2007.
- [12] YAROWSKY, D., “Unsupervised word sense disambiguation rivaling supervised methods”. In: *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pp. 189–196, Association for Computational Linguistics, 1995.
- [13] LIU, B., “Sentiment analysis and opinion mining”, *Synthesis lectures on human language technologies*, v. 5, n. 1, pp. 1–167, 2012.
- [14] WIEBE, J., RILOFF, E., “Creating Subjective and Objective Sentence Classifiers from Unannotated Texts.” In: *CICLing*, v. 5, pp. 486–497, Springer, 2005.
- [15] BOLLEN, J., MAO, H., PEPE, A., “Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena.”, *Icwsn*, v. 11, pp. 450–453, 2011.
- [16] ZHUANG, L., JING, F., ZHU, X.-Y., “Movie review mining and summarization”. In: *Proceedings of the 15th ACM international conference on Information and knowledge management*, pp. 43–50, ACM, 2006.
- [17] HU, M., LIU, B., “Mining and summarizing customer reviews”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177, ACM, 2004.

- [18] BOLLEN, J., MAO, H., ZENG, X., “Twitter mood predicts the stock market”, *Journal of computational science*, v. 2, n. 1, pp. 1–8, 2011.
- [19] TUMASJAN, A., SPRENGER, T. O., SANDNER, P. G., *et al.*, “Predicting elections with twitter: What 140 characters reveal about political sentiment.”, *Icwsm*, v. 10, n. 1, pp. 178–185, 2010.
- [20] DU, J., XU, H., HUANG, X., “Box office prediction based on microblog”, *Expert Systems with Applications*, v. 41, n. 4, pp. 1680–1689, 2014.
- [21] TURING, A. M., “Computing machinery and intelligence”, *Mind*, v. 59, n. 236, pp. 433–460, 1950.
- [22] SILVA, C., RIBEIRO, B., “The importance of stop word removal on recall values in text categorization”. In: *Neural Networks, 2003. Proceedings of the International Joint Conference on*, v. 3, pp. 1661–1666, IEEE, 2003.
- [23] SAIF, H., FERNÁNDEZ, M., HE, Y., *et al.*, “On stopwords, filtering and data sparsity for sentiment analysis of twitter”, , 2014.
- [24] SCHÜTZE, H., “Introduction to information retrieval”. In: *Proceedings of the international communication of association for computing machinery conference*, 2008.
- [25] SALTON, G., BUCKLEY, C., “Term-weighting approaches in automatic text retrieval”, *Information processing & management*, v. 24, n. 5, pp. 513–523, 1988.
- [26] MIKOLOV, T., CHEN, K., CORRADO, G., *et al.*, “Efficient estimation of word representations in vector space”, *arXiv preprint arXiv:1301.3781*, , 2013.
- [27] MIKOLOV, T., SUTSKEVER, I., CHEN, K., *et al.*, “Distributed Representations of Words and Phrases and their Compositionality”. In: Burges, C. J. C., Bottou, L., Welling, M., *et al.* (eds.), *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pp. 3111–3119, 2013.
- [28] PENNINGTON, J., SOCHER, R., MANNING, C., “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

- [29] BOJANOWSKI, P., GRAVE, E., JOULIN, A., *et al.*, “Enriching Word Vectors with Subword Information”, *arXiv preprint arXiv:1607.04606*, , 2016.
- [30] MCCALLUM, A., NIGAM, K., OTHERS, “A comparison of event models for naive bayes text classification”. Citeseer.
- [31] VANDERPLAS, J., https://github.com/jakevdp/sklearn_pycon2015/blob/master/notebooks/03.1-Classification-SVMs.ipynb, acessado em 28 de Maio 2017.
- [32] SCHÖLKOPF, B., SMOLA, A., “Support Vector Machines and Kernel Algorithms”, 2002.
- [33] CORTES, C., VAPNIK, V., “Support-vector networks”, *Machine Learning*, v. 20, n. 3, pp. 273–297, 1995.
- [34] LIST, N., SIMON, H. U., “SVM-optimization and steepest-descent line search”. In: *Proceedings of the 22nd Annual Conference on Computational Learning Theory*, 2009.
- [35] SUYKENS, J. A., VANDEWALLE, J., “Least squares support vector machine classifiers”, *Neural processing letters*, v. 9, n. 3, pp. 293–300, 1999.
- [36] WIENER, N., *Cybernetics or Control and Communication in the Animal and the Machine*, v. 25. MIT press, 1961.
- [37] ROSENBLATT, F., “The perceptron: A probabilistic model for information storage and organization in the brain.”, *Psychological review*, v. 65, n. 6, pp. 386, 1958.
- [38] HORNIK, K., STINCHCOMBE, M., WHITE, H., “Multilayer feedforward networks are universal approximators”, *Neural networks*, v. 2, n. 5, pp. 359–366, 1989.
- [39] MINSKY, M., PAPERT, S., *Perceptrons: An Introduction to Computational Geometry*. Mit Press, 1972.
- [40] WERBOS, P. J., “Applications of advances in nonlinear sensitivity analysis”. In: *System modeling and optimization*, Springer, pp. 762–770, 1982.

- [41] WILLIAMS, D., HINTON, G., “Learning representations by back-propagating errors”, *Nature*, v. 323, n. 6088, pp. 533–538, 1986.
- [42] HOCHREITER, S., “The vanishing gradient problem during learning recurrent neural nets and problem solutions”, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, v. 6, n. 02, pp. 107–116, 1998.
- [43] HINTON, G. E., OSINDERO, S., TEH, Y.-W., “A fast learning algorithm for deep belief nets”, *Neural computation*, v. 18, n. 7, pp. 1527–1554, 2006.
- [44] NAIR, V., HINTON, G. E., “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.
- [45] HE, K., ZHANG, X., REN, S., *et al.*, “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [46] LECUN, Y., BENGIO, Y., HINTON, G., “Deep learning”, *Nature*, v. 521, n. 7553, pp. 436–444, 2015.
- [47] ESTEVA, A., KUPREL, B., NOVOA, R. A., *et al.*, “Dermatologist-level classification of skin cancer with deep neural networks”, *Nature*, v. 542, n. 7639, pp. 115–118, 2017.
- [48] HOPFIELD, J. J., “Neural networks and physical systems with emergent collective computational abilities”. In: *Spin Glass Theory and Beyond: An Introduction to the Replica Method and Its Applications*, World Scientific, pp. 411–415, 1987.
- [49] HINTON, G. E., SALAKHUTDINOV, R. R., “Reducing the dimensionality of data with neural networks”, *science*, v. 313, n. 5786, pp. 504–507, 2006.
- [50] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., *et al.*, “Generative Adversarial Nets”. In: Ghahramani, Z., Welling, M., Cortes, C., *et al.* (eds.), *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., pp. 2672–2680, 2014.

- [51] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [52] LECUN, Y., BENGIO, Y., OTHERS, “Convolutional networks for images, speech, and time series”, *The handbook of brain theory and neural networks*, v. 3361, n. 10, 1995.
- [53] CAUCHY, A., “Méthode générale pour la résolution des systemes d’équations simultanées”, *Comp. Rend. Sci. Paris*, v. 25, n. 1847, pp. 536–538, 1847.
- [54] GLOROT, X., BENGIO, Y., “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.
- [55] DAUPHIN, Y. N., PASCANU, R., GULCEHRE, C., *et al.*, “Identifying and attacking the saddle point problem in high-dimensional non-convex optimization”. In: *Advances in neural information processing systems*, pp. 2933–2941, 2014.
- [56] GOODFELLOW, I. J., VINYALS, O., SAXE, A. M., “Qualitatively characterizing neural network optimization problems”, *arXiv preprint arXiv:1412.6544*, , 2014.
- [57] HAYKIN, S., *Neural networks and learning machines*, v. 3. Pearson Upper Saddle River, NJ, USA:, 2009.
- [58] DUCHI, J., HAZAN, E., SINGER, Y., “Adaptive subgradient methods for online learning and stochastic optimization”, *Journal of Machine Learning Research*, v. 12, n. Jul, pp. 2121–2159, 2011.
- [59] KINGMA, D., BA, J., “Adam: A method for stochastic optimization”, *arXiv preprint arXiv:1412.6980*, , 2014.
- [60] KROGH, A., HERTZ, J. A., “A simple weight decay can improve generalization”. In: *Advances in neural information processing systems*, pp. 950–957, 1992.

- [61] SRIVASTAVA, N., HINTON, G. E., KRIZHEVSKY, A., *et al.*, “Dropout: a simple way to prevent neural networks from overfitting.”, *Journal of machine learning research*, v. 15, n. 1, pp. 1929–1958, 2014.
- [62] DIETTERICH, T. G., OTHERS, “Ensemble methods in machine learning”, *Multiple classifier systems*, v. 1857, pp. 1–15, 2000.
- [63] CRAVEN, M., KUMLIEN, J., OTHERS, “Constructing biological knowledge bases by extracting information from text sources.” In: *ISMB*, v. 1999, pp. 77–86, 1999.
- [64] READ, J., “Using emoticons to reduce dependency in machine learning techniques for sentiment classification”. In: *Proceedings of the ACL student research workshop*, pp. 43–48, Association for Computational Linguistics, 2005.
- [65] “International Workshop on Semantic Evaluation 2017”, <http://alt.qcri.org/semeval2017/>, acessado em 28 de Maio de 2017.
- [66] ROSENTHAL, S., FARRA, N., NAKOV, P., “SemEval-2017 task 4: Sentiment analysis in Twitter”. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 502–518, 2017.
- [67] BRADLEY, A. P., “The use of the area under the ROC curve in the evaluation of machine learning algorithms”, *Pattern recognition*, v. 30, n. 7, pp. 1145–1159, 1997.
- [68] CIODARO, T., DEVA, D., DE SEIXAS, J., *et al.*, “Online particle detection with neural networks based on topological calorimetry information”. In: *Journal of physics: conference series*, v. 368, p. 012030, IOP Publishing, 2012.
- [69] KOHAVI, R., OTHERS, “A study of cross-validation and bootstrap for accuracy estimation and model selection”. In: *Ijcai*, v. 14, pp. 1137–1145, Stanford, CA, 1995.
- [70] DERIU, J., GONZENBACH, M., UZDILLI, F., *et al.*, “SwissCheese at SemEval-2016 Task 4: Sentiment Classification Using an Ensemble of Convolutional Neural Networks with Distant Supervision.”, 2016.

- [71] CARUANA, R., LAWRENCE, S., GILES, C. L., “Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping”. In: *Advances in neural information processing systems*, pp. 402–408, 2001.