



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 idéias, 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

Inserir o resumo do seu trabalho aqui. O objetivo é apresentar ao pretendo leitor do seu Projeto Final uma descrição genérica do seu trabalho. Você também deve tentar despertar no leitor o interesse pelo conteúdo deste documento.

Palavras-Chave: trabalho, resumo, interesse, projeto final.

ABSTRACT

Insert your abstract here. Insert your abstract here. Insert your abstract here.
Insert your abstract here. Insert your abstract here.

Key-words: word, word, word.

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	1
1.3	Justificativa	2
1.4	Objetivos	2
1.5	Metodologia	3
1.6	Descrição	3
2	Contexto	4
2.1	Análise de Sentimento	4
3	Processamento de Linguagem Natural	5
3.1	Pré-Processamentos	5
3.2	Representações Numéricas	6
3.2.1	Codificação One-Hot	6
3.2.2	Bag-of-Words	6
3.2.3	Word2Vec	7
4	Aprendizado Supervisionado	9
4.1	Naive Bayes	9
4.2	Support Vector Machine	10
4.3	Redes Neurais	13
4.3.1	Deep Learning	15
4.3.2	Redes Neurais Convolucionais	15
4.4	Funções Custo	16
4.4.1	Erro Médio Quadrático	16

4.4.2	Erro Médio Absoluto	16
4.4.3	Erro Médio Absoluto Percentual	17
4.4.4	Entropia Cruzada	17
4.5	Otimizadores	18
4.5.1	Gradiente Descendente Estocástico	19
4.5.2	Gradiente Descendente com Momento	19
4.5.3	Adam	20
4.6	Generalização	21
4.6.1	Regularização de Tikhonov	23
4.6.2	Dropout	25
4.7	Supervisão Distante	26
5	Metodologia	27
6	Resultados	28
7	Conclusões	29
	Bibliografia	30
A	O que é um apêndice	34
B	Encadernação do Projeto de Graduação	35
C	O que é um anexo	37

Lista de Figuras

4.1	Reta de maior margem entre classes.	11
4.2	SVM em dados não linearmente separáveis.	12
4.3	Transformação de dados por função de base radial.	12
4.4	SVM com <i>kernel</i> de base radial.	13
4.5	Rede neural multilayer perceptron	14
4.6	Diferentes pontos de ajuste.	24
B.1	Encadernação do projeto de graduação.	36

Lista de Tabelas

Lista de Algoritmos

1	Gradiente Descendente	19
2	Gradiente Descendente com Momento	20
3	Adam	22

Capítulo 1

Introdução

1.1 Tema

O tema do projeto é o estudo de técnicas de aprendizado de máquina para classificação de sentimento em mensagens de redes sociais, em especial o Twitter. Portanto, o projeto visa criar um modelo capaz de distinguir entre a polaridade, positiva ou negativa, de uma mensagem.

1.2 Delimitação

O objeto de estudo são *tweets*, mensagens publicadas no Twitter. Serão utilizadas duas bases de dados: a primeira, uma base anotada automaticamente pelo *Sentiment140* [1], o qual é formado por um grupo de alunos de *Stanford University*; e a segunda, uma base com anotação manual, formada pela coletânea de dados disponibilizados anualmente entre 2013 e 2017 pelo *International Workshop on Semantic Evaluation* [2]. A análise de sentimento a ser aplicada terá seu enfoque 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.

1.3 Justificativa

Acompanhamos ao longo da última década a massificação do uso de redes sociais. O Twitter, objeto de nosso estudo, conta com 310 milhões de usuários ativos, gerando um total de meio bilhão de *tweets* por dia. Tais estatísticas provam ser cada vez mais necessário ferramentas capazes de automatizar o processo extração de informação deste mar de dados.

A análise de sentimento neste contexto visa resgatar opinião sobre um assunto, evento ou produto. Os primeiros estudos deste tópico aplicados ao Twitter foram desenvolvidos em 2009 por Go et al., [3]. Esse grupo de pesquisadores utiliza, em seu artigo, as melhores técnicas de classificação de texto disponíveis na época e ressalta as diferenças de seus resultados comparados a quando aplicadas em textos jornalísticos, resenhas etc.

O campo do processamento de linguagem natural foi fortemente impactado pelo crescimento do *Deep Learning*. Técnicas como Redes Neurais Convolutivas (CNN), a princípio desenvolvidas para processamento de imagens, e *Long Short-Term Memory* (LSTM) propulsionaram o salto de desempenho obtido nos últimos anos. O impacto deste avanço é notório no nosso dia-a-dia. Frequentemente utilizamos essas técnicas como em serviços automatizados de atendimento ao cliente ou em ferramentas de tradução simultânea.

Neste sentido, o presente projeto visa aplicar em *tweets* os procedimentos que compõe o estado da arte em classificação de sentimento.

1.4 Objetivos

O objetivo deste projeto, portanto, consiste em gerar um modelo computacional capaz de sistematizar a classificação de sentimento de *tweets*. Este modelo deve ser independente de uma base de dados anotada manualmente, visto o alto custo desta ser reproduzida.

1.5 Metodologia

Para alcançar esse objetivo, as seguintes etapas serão necessárias: (1) replicar técnicas consolidadas de análise de sentimento para *tweets* e utilizá-las como referência; (2) aplicar nos *tweets* técnicas de *Deep Learning* que vêm obtendo sucesso em processamento de linguagem natural.

A primeira etapa do trabalho será a replicação de estudos desenvolvidos por Go et al., [3] que aplica técnicas de *Naive Bayes* e *SVM* na classificação de polaridade de *tweets*, sendo seu treinamento feito em cima de uma base de dados com anotação automática, conforme apresentado por Read [4]. Os resultados obtidos por estas técnicas serão utilizados como patamar para comparação dos modelos a serem gerados.

Posteriormente, serão aplicadas técnicas de *Deep Learning*, como apresentadas por Kim [5], em que se utiliza CNNs para classificação de texto. O treinamento continua sendo feito a partir do banco de dados com anotação semi-supervisionada, o qual foi apresentado anteriormente. Será avaliado o desempenho desta técnica quando aplicada em mensagens do Twitter.

1.6 Descrição

No capítulo 2 será

O capítulo 3 apresenta ...

Os são apresentados no capítulo 4. Nele será explicitado ...

E assim vai até chegar na conclusão.

Capítulo 2

Contexto

O crescimento da internet acontece de forma exponencial. Estima-se que no ano de 2016 cerca de 66% da população brasileira tenha acesso a rede [6]. A internet modificou a forma A criação de conteúdo digital acompanha...

No Brasil, 58% da população, ou seja, 120 milhões de pessoas, participam de pelo menos uma rede social [6].

2.1 Análise de Sentimento

Capítulo 3

Processamento de Linguagem Natural

3.1 Pré-Processamentos

A primeira etapa na preparação de um documento e/ou 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 token. A *tokenização* costuma incluir a separação de contrações como a transformação de *dela* em *de* e *ela*.

Para aplicação de tarefas como a de classificação é comum se remover as palavras mais comuns de um idioma e que não agregam informação discriminantes para o objetivo 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 [7]. Porém, a remoção de *stopwords* provindas de listas pré compiladas, técnica mais utilizada, apresenta desafios principalmente devido a dinamicidade de meios como redes sociais. Saif *et al.* [8] apresentam um estudo comparativo do efeito 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 lema-

tização, 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. Portanto, após a conversão de uma mensagem em 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 [9].

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) [10]. 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 [11] (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.* [11] 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 da rede 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 *Skip-gram* 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.* [12] 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 [13], algoritmo desenvolvido por um grupo de pesquisadores de Stanford e FastText [14], 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 preditores a partir de um conjunto de dados de treinamento cujos resultados são previamente conhecidos.

4.1 Naive Bayes

Naive 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 apresentado a seguir sua formulação matemática como descrita por Schütze [9].

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:

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

No qual \mathbf{x} é um vetor de n características, 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 produtório 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 *Naive 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 *Naive 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 [15].

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.

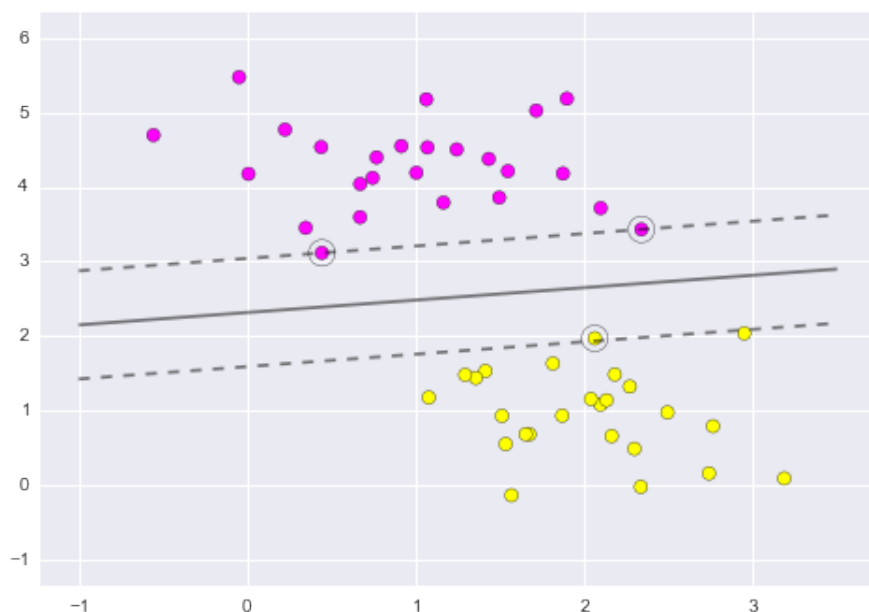


Figura 4.1: Reta de maior margem entre classes.

Imagem com direitos cedidos para uso não comercial, retirada de [16]

Por se basear nos dados próximos ao limiar de separação das classes, o algoritmo passa a ser 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 [17]. Sua utilização permite o desenvolvimento de modelos mais robustos a *outliers* e melhora a generalização. Outros exemplos de regularizadores, como este, serão apresentados na subseção 4.6.

Como utilizam vetores de suporte para definir hiperplanos de separação, SVMs não são capazes de segregar classes não linearmente distinguíveis. Podemos observar um exemplo deste caso na figura 4.2, na qual um 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 [18]. 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

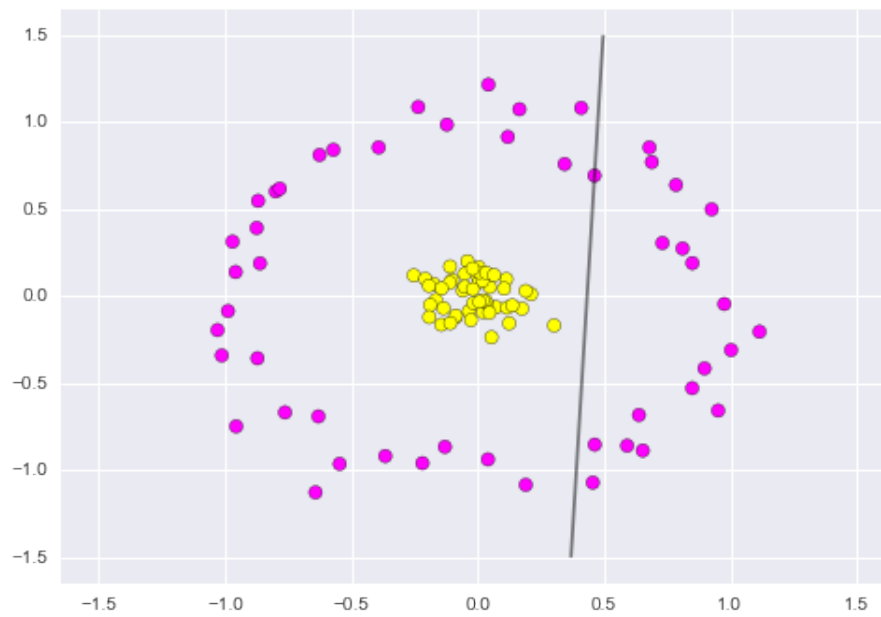


Figura 4.2: SVM em dados não linearmente separáveis.

Imagem com direitos cedidos para uso não comercial, retirada de [16]

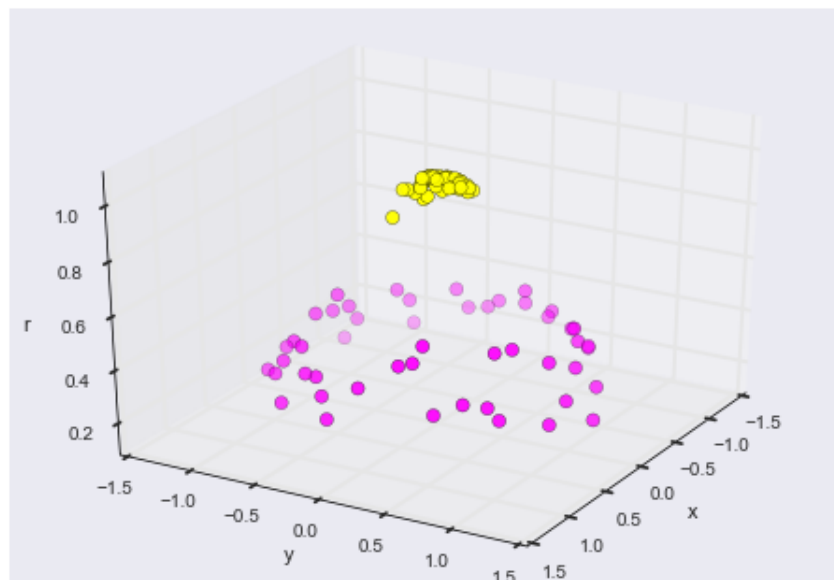


Figura 4.3: Transformação de dados por função de base radial.

Imagem com direitos cedidos para uso não comercial, retirada de [16]

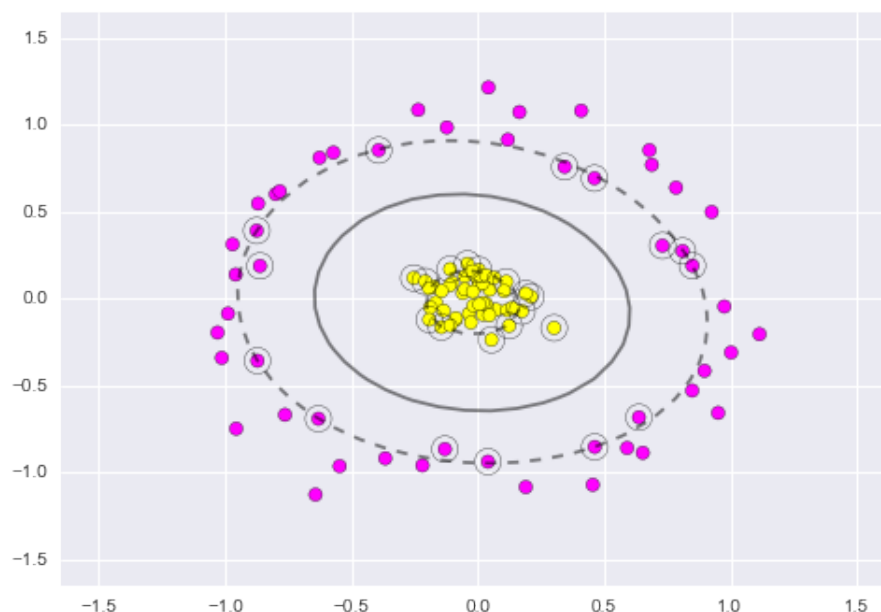


Figura 4.4: SVM com *kernel* de base radial.

Imagem com direitos cedidos para uso não comercial, retirada de [16]

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.

Uma limitação na utilização deste algoritmo é seu tempo de treinamento. Sua complexidade computacional fica entre $O(n_{caracteristicas} \times n_{dados}^2)$ e $O(n_{caracteristicas} \times n_{dados}^3)$ [19]. Porém, Suykens e Vandewalle desenvolveram uma função custo através da qual tornou possível realizar o treinamento de SVMs a partir da otimização pelo método do gradiente [20].

4.3 Redes Neurais

Redes neurais são sistemas computacionais que visam replicar o modelo de processamento do cérebro [21]. Inspirado pelo campo do conexionismo criado por Rosenblatt [22] desenvolve

Redes neurais MLP são compostas de camadas de neurônios sucessivamente interligadas por sinapses. As sinapses funcionam como um ponderador linear dos neurônios da camada anterior. Os neurônios, por sua vez, representam uma função de ativação, normalmente não linear, tais como: tangente hiperbólica, logística etc.

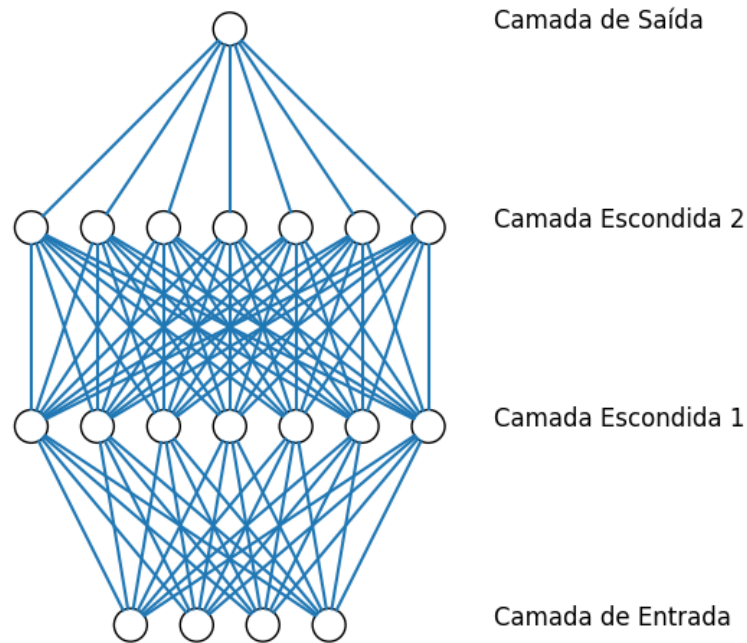


Figura 4.5: Rede neural multilayer perceptron

Como exemplificação, há a 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 [23].

Para uma rede neural simular uma função, precisa-se obter o conjunto de pesos, ou sinapses, que torne esta rede uma boa representação para a função escolhida. Em outras palavras, precisa-se minimizar o erro da representação da rede dado que este erro é uma função das sinapses. Uma prática comum é inicializar a rede com pesos selecionados de uma distribuição uniforme com média zero e desvio padrão inversamente proporcional ao número de dados de treinamento [24]. Posteriormente, é feita uma otimização da representação em função dos pesos. Para realizar esta otimização, utiliza-se normalmente *Backpropagation* para se encontrar o ajuste ideal

de cada sinapse baseada, em sua contribuição no erro. Uma explicação deste método é apresentada por Williams e Hinton em [25].

4.3.1 Deep Learning

O sucesso na aplicação de redes neurais nas mais diversas áreas levou à utilização de redes cada vez mais robustas. Assim, redes neurais profundas romperam barreiras de performance em problemas como reconhecimento de fala, detecção de objetos, tradução etc [26]. Chamou-se de *Deep Learning* o campo de estudo de redes neurais com muitas camadas.

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 visa a reduzir ou eliminar a 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 [27].

4.3.2 Redes Neurais Convolucionais

A utilização de redes neurais feedforward 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 feedforward [28]. 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 sua posição no dado [28].

Por fim, 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 [29]. Tal sucesso pode ser atribuído à insensibilidade da rede a deslocamentos.

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

$$\frac{\sum^n |\hat{y}_i - y_i|^2}{n} \quad (4.8)$$

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 4.9.

$$\frac{\sum^n |\hat{y}_i - y_i|}{n} \quad (4.9)$$

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 4.10

$$\frac{100}{n} \sum^n \left| \frac{\hat{\mathbf{y}}_i - \mathbf{y}_i}{\mathbf{y}_i} \right| \quad (4.10)$$

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 4.11 e 4.12 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.

$$\mathcal{L}(\theta) = \prod_k P(\mathbf{y}_k | \mathbf{x}) \quad (4.11)$$

$$\mathcal{L}(\theta) = \prod_k \hat{\mathbf{y}}_k^{\mathbf{y}_k} (1 - \hat{\mathbf{y}}_k)^{(1-\mathbf{y}_k)} \quad (4.12)$$

Para simplificar o custo computacional, o valor máximo da função verossimilhança é obtido minimizando-se o negativo de seu logaritmo, como presente na fórmula 4.13.

$$-\ln(\mathcal{L}(\theta)) = - \sum_k \mathbf{y}_k \ln(\hat{\mathbf{y}}_k) + (1 - \mathbf{y}_k) \ln(1 - \hat{\mathbf{y}}_k) \quad (4.13)$$

A expressão 4.14 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.

$$\frac{-\sum_k^n y_k \ln(\hat{y}_k)}{n} \quad (4.14)$$

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 [30], 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 [31]. Entretanto, o problema de convergência não se limita a mínimos locais. Dauphin [32] 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 [33] apresenta resultados empíricos da efetividade do método do gradiente na fuga dos pontos de sela.

O algoritmo 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 1 Gradiente Descendente

Parâmetro Taxa de treinamento ϵ

Parâmetro Pesos iniciais θ_0

Parâmetro Número de iterações T

$t \leftarrow 0$

while $t < T$ **do**

$t \leftarrow t + 1$

$\hat{\mathbf{y}} \leftarrow f(\mathbf{x}, \theta_{t-1})$

$\hat{\mathbf{g}} \leftarrow \frac{1}{n} \nabla_{\theta} \sum_i^n L(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$

$\theta_t \leftarrow \theta_{t-1} - \epsilon \hat{\mathbf{g}}$

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 1, para cada atualização dos pesos θ 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 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 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 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

$t \leftarrow 0$

$\mathbf{v}_0 \leftarrow \vec{0}$

while $t < T$ **do**

$t \leftarrow t + 1$

$\hat{\mathbf{y}} \leftarrow f(\mathbf{x}, \boldsymbol{\theta}_{t-1})$

$\hat{\mathbf{g}} \leftarrow \frac{1}{n} \nabla_{\boldsymbol{\theta}} \sum_i^n L(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$

$\mathbf{v}_t \leftarrow \alpha \mathbf{v}_{t-1} - \epsilon \hat{\mathbf{g}}$

$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} + \mathbf{v}_t$

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 [34], 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.* [35] propuseram o algoritmo *AdaGrad* (*Adaptive 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.* [13] no treinamento de *embeddings* de palavras, técnica que será apresentada na seção 3.2.3.

Kingma e Ba [36] 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 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 [36], 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 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

$t \leftarrow 0$

$\mathbf{m}_0 \leftarrow \vec{0}$

$\mathbf{v}_0 \leftarrow \vec{0}$

while $t < T$ **do**

$t \leftarrow t + 1$

$\hat{\mathbf{y}} \leftarrow f(\mathbf{x}, \theta_{t-1})$

$\hat{\mathbf{g}} \leftarrow \frac{1}{n} \nabla_{\theta} \sum_i^n L(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$

$\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \hat{\mathbf{g}}$

$\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \hat{\mathbf{g}}^2$

$\hat{\mathbf{m}}_t \leftarrow \frac{\mathbf{m}_t}{(1 - \beta_1^t)}$

$\hat{\mathbf{v}}_t \leftarrow \frac{\mathbf{v}_t}{(1 - \beta_2^t)}$

$\theta_t \leftarrow \theta_{t-1} - \epsilon \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t}}$

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 4.15 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 .

$$\frac{\sum^n |\hat{\mathbf{y}}_i - \mathbf{y}_i|^2}{n} + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 \quad (4.15)$$

Krogh e Hertz [37] 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.

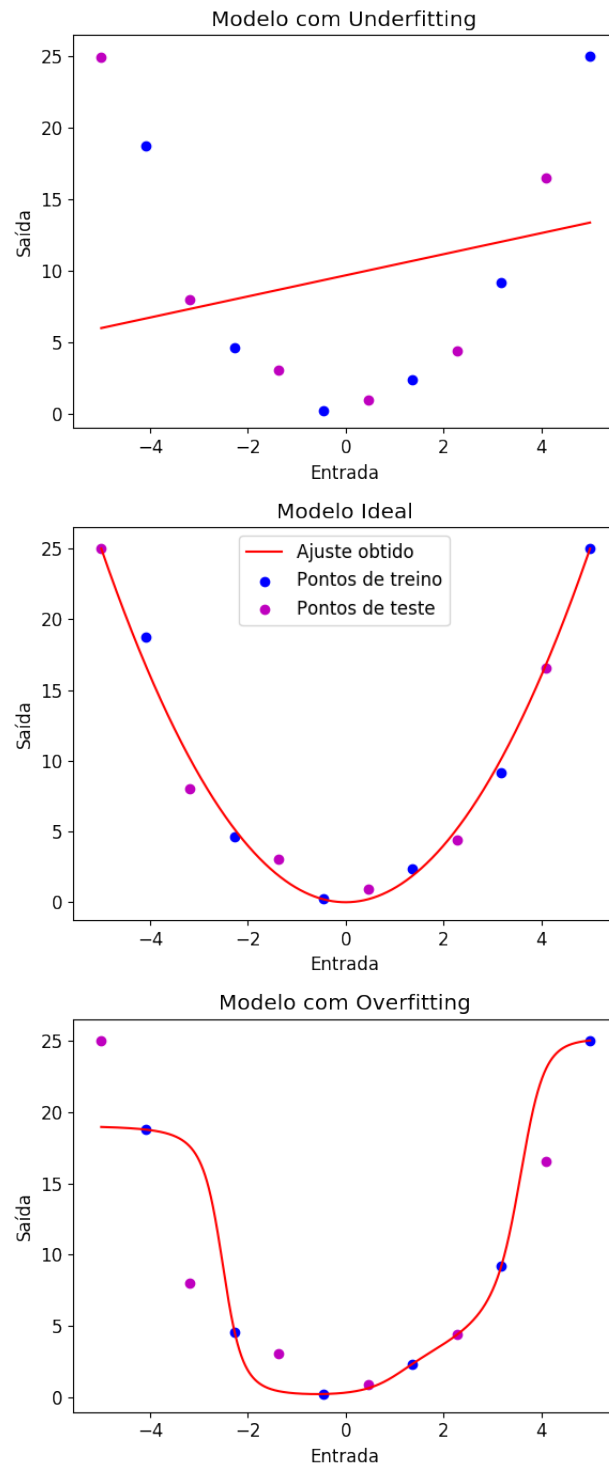


Figura 4.6: Diferentes pontos de ajuste.

4.6.2 Dropout

Apresentado por Srivastava *et al.* [38], *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.* [39], 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 extraindo anotação de maneira automatizada. Essa técnica consiste encontrar uma característica que tenha forte correlação com a saída desejada e considera-la como uma anotação ruidosa [3].

Esse procedimento obteve êxito em algumas aplicações de processamento de linguagem natural [40][3] pela abundância de informações presentes de forma desestruturada. Craven *et al.* [40] apresenta uma das primeiras aplicações da técnica extraindo a partir de registros textuais informações sobre proteínas e suas relações.

Capítulo 5

Metodologia

Capítulo 6

Resultados

Capítulo 7

Conclusões

Tratam-se das considerações finais do trabalho, mostrando que os objetivos foram cumpridos e enfatizando as descobertas feitas durante o projeto. Em geral reserva-se um ou dois parágrafos para sugerir trabalhos futuros.

Observe que neste modelo a conclusão é numerada pelo numeral 3, mas o projeto não tem a obrigatoriedade de possuir apenas 3 capítulos. Alias, espera-se que tenha mais que isso.

Referências Bibliográficas

- [1] “Sentiment140 - A Twitter Sentiment Analysis Tool”, <http://www.sentiment140.com/>, acessado em 28 de Maio de 2017.
- [2] “International Workshop on Semantic Evaluation 2017”, <http://alt.qcri.org/semeval2017/>, acessado em 28 de Maio de 2017.
- [3] GO, A., BHAYANI, R., HUANG, L., “Twitter sentiment classification using distant supervision”, *CS224N Project Report, Stanford*, v. 1, n. 12, 2009.
- [4] 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.
- [5] KIM, Y., “Convolutional neural networks for sentence classification”, *arXiv preprint arXiv:1408.5882*, , 2014.
- [6] 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.
- [7] 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.
- [8] SAIF, H., FERNÁNDEZ, M., HE, Y., *et al.*, “On stopwords, filtering and data sparsity for sentiment analysis of twitter”, , 2014.
- [9] SCHÜTZE, H., “Introduction to information retrieval”. In: *Proceedings of the international communication of association for computing machinery conference*, 2008.

- [10] SALTON, G., BUCKLEY, C., “Term-weighting approaches in automatic text retrieval”, *Information processing & management*, v. 24, n. 5, pp. 513–523, 1988.
- [11] MIKOLOV, T., CHEN, K., CORRADO, G., *et al.*, “Efficient estimation of word representations in vector space”, *arXiv preprint arXiv:1301.3781*, , 2013.
- [12] 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.
- [13] 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.
- [14] BOJANOWSKI, P., GRAVE, E., JOULIN, A., *et al.*, “Enriching Word Vectors with Subword Information”, *arXiv preprint arXiv:1607.04606*, , 2016.
- [15] MCCALLUM, A., NIGAM, K., OTHERS, “A comparison of event models for naive bayes text classification”. Citeseer.
- [16] VANDERPLAS, J., https://github.com/jakevdp/sklearn_pycon2015/blob/master/notebooks/03.1-Classification-SVMs.ipynb, acessado em 28 de Maio 2017.
- [17] CORTES, C., VAPNIK, V., “Support-vector networks”, *Machine Learning*, v. 20, n. 3, pp. 273–297, 1995.
- [18] SCHÖLKOPF, B., SMOLA, A., “Support Vector Machines and Kernel Algorithms”, 2002.
- [19] LIST, N., SIMON, H. U., “SVM-optimization and steepest-descent line search”. In: *Proceedings of the 22nd Annual Conference on Computational Learning Theory*, 2009.
- [20] SUYKENS, J. A., VANDEWALLE, J., “Least squares support vector machine classifiers”, *Neural processing letters*, v. 9, n. 3, pp. 293–300, 1999.

- [21] WIENER, N., *Cybernetics or Control and Communication in the Animal and the Machine*, v. 25. MIT press, 1961.
- [22] ROSENBLATT, F., “The perceptron: A probabilistic model for information storage and organization in the brain.”, *Psychological review*, v. 65, n. 6, pp. 386, 1958.
- [23] HORNIK, K., STINCHCOMBE, M., WHITE, H., “Multilayer feedforward networks are universal approximators”, *Neural networks*, v. 2, n. 5, pp. 359–366, 1989.
- [24] LECUN, Y. A., BOTTOU, L., ORR, G. B., *et al.*, “Efficient backprop”. In: *Neural networks: Tricks of the trade*, Springer, pp. 9–48, 2012.
- [25] WILLIAMS, D., HINTON, G., “Learning representations by back-propagating errors”, *Nature*, v. 323, n. 6088, pp. 533–538, 1986.
- [26] LECUN, Y., BENGIO, Y., HINTON, G., “Deep learning”, *Nature*, v. 521, n. 7553, pp. 436–444, 2015.
- [27] 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.
- [28] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [29] 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.
- [30] CAUCHY, A., “Méthode générale pour la résolution des systemes d’équations simultanées”, *Comp. Rend. Sci. Paris*, v. 25, n. 1847, pp. 536–538, 1847.
- [31] 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.

- [32] 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.
- [33] GOODFELLOW, I. J., VINYALS, O., SAXE, A. M., “Qualitatively characterizing neural network optimization problems”, *arXiv preprint arXiv:1412.6544*, , 2014.
- [34] HAYKIN, S., *Neural networks and learning machines*, v. 3. Pearson Upper Saddle River, NJ, USA:, 2009.
- [35] 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.
- [36] KINGMA, D., BA, J., “Adam: A method for stochastic optimization”, *arXiv preprint arXiv:1412.6980*, , 2014.
- [37] KROGH, A., HERTZ, J. A., “A simple weight decay can improve generalization”. In: *Advances in neural information processing systems*, pp. 950–957, 1992.
- [38] 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.
- [39] DIETTERICH, T. G., OTHERS, “Ensemble methods in machine learning”, *Multiple classifier systems*, v. 1857, pp. 1–15, 2000.
- [40] CRAVEN, M., KUMLIEN, J., OTHERS, “Constructing biological knowledge bases by extracting information from text sources.” In: *ISMB*, v. 1999, pp. 77–86, 1999.

Apêndice A

O que é um apêndice

Elemento que consiste em um texto ou documento elaborado pelo autor, com o intuito de complementar sua argumentação, sem prejuízo do trabalho. São identificados por letras maiúsculas consecutivas e pelos respectivos títulos.

Apêndice B

Encadernação do Projeto de Graduação

<div>Número</div> <div>Nome do Aluno</div> <div>Título do Projeto*</div> <div>Ano</div>	<div>UNIVERSIDADE FEDERAL DO RIO DE JANEIRO</div> <div>Escola Politécnica</div> <div>Departamento de Eletrônica e de Computação</div> <div>Título do Projeto</div> <div>Nome do Aluno</div> <div>Projeto de Graduação</div> <div>Mês / Ano</div>

*** Título resumido caso necessário**
Capa na cor preta, inscrições em dourado

Figura B.1: Encadernação do projeto de graduação.

Apêndice C

O que é um anexo

Documentação não elaborada pelo autor, ou elaborada pelo autor mas constituindo parte de outro projeto.