UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Lúcia Corrêa da Silva

Detecção e classificação de padrões sonoros utilizando inteligência artificial

Florianópolis-SC
2021

Lúcia Corrêa da Silva

Detecção e classificação de padrões sonoros utilizando inteligência artificial

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal de Santa Catarina para a obtenção do título de Mestre em Engenharia Mecânica.
Orientador: Prof. Arcanjo Lenzi, PhD

Florianópolis-SC
2021

Lúcia Corrêa da Silva

Detecção e classificação de padrões sonoros utilizando inteligência artificial

O presente trabalho em nível de Mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Mestre em Engenharia Mecânica.

Coordenador do Programa

Prof. Arcanjo Lenzi, PhD
Orientador

Florianópolis-SC, 31 de março de 2021.

AGRADECIMENTOS

RESUMO

apresentar rapidamente os objetivos, justificativa, uma ideia da metodologia (citar redes usadas) e resultados **Palavras-chave**:

ABSTRACT

Keywords:

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Os padrões sonoros permitem identificar características importantes do comportamento de sistemas dinâmicos possibilitando ao ser humano reconhecer os mais variados fenômenos. Com as habilidades auditivas e vocais, um individuo é capaz de gerar e discernir sinais sonoros sem significados, fonemas, palavra, frases, e até em uma conversa conseguir encontrar o local de origem de um locutor pelo seu sotaque e regionalismos. Também é possível criar sons com ritmos, timbres, estilos diferentes assimilando assim sues contrastes. Essa capacidade também pode ser utilizada para dar significado a alertas sonoros, ruídos causados por animais, batidas, entre outros.

As aptidões humanas de classificação de sinais foram relevantes no seu desenvolvimento e na forma de interagir com o mundo (**jung2020human**). Logo, as descrição de comportamento de sistemas são estudadas para desenvolver técnicas que possibilitam aperfeiçoar ou adicionar recursos em produtos e serviços, como também orientar a criação de ferramentas e a detecção de problemas. A identificação de falhas ou condições atípicas de operação em máquinas é uma aplicação interessante (**yang2019machine**), pois pode ser verificada diariamente nos produtos, por exemplo, ruídos incomuns produzido por automóveis podem avisar o próprio condutor ou mecânico de algum tipo de problema. Logo, alguns sinais relatam tanto a alertas de avaria como também da qualidade do dispositivo.

No contexto industrial, essas ferramentas podem ser de grande utilidade para prever falhas tanto nas máquinas da linha de fabricação (**purohit2019mimii**) quanto no produto final (**yang2019machine**), elas auxiliam no desenvolvimento de equipamento em baixo custos de forma automatizada e otimizada (**zahid2015optimized**). Outra prática relevante é no monitoramento a partir de sistemas embarcados, o que é vantajoso em situações de grande prejuízo caso a falha no equipamento não seja detectada imediatamente. Nestes casos, o interessante é obter uma análise prévia, se não imediata do problema, para uma abordagem simples e efetiva, gerando menos perda de material e tempo.

Do ponto de vista técnico, a classificação consiste em extrair características tanto físicas quanto perceptivas de um sinal sonoro e, a partir disso, definir uma classe na qual o evento em questão melhor se enquadra (**alias2016review**). Para isso, pode-se utilizar algoritmos de extração e classificação de recursos que são bastante diversificados. Uma forma de simplificar essa escolha é ter em mente que ela depende do domínio no qual a análise será realizada.

Atualmente, novos métodos estão sendo estudados, entre eles a classificação automática de sinais sonoros utilizando Inteligência Artificial (AI)[1]. Essa é uma área crescente de pesquisa com inúmeras aplicações no mundo real existindo uma grande quantidade de trabalhos em campos relacionados ao áudio(**unknown**), como fala (**abdel2014convolutional**) e música (**boddapati2017classifying**), e alguns trabalhos sobre a classificação de sons

[1] Acrônimo do inglês para *Artificial Intelligence*.

ambientais (**boddapati2017classifying**). O conteúdo desses estudos são empregados em produtos e serviços tais como: aplicativos de reconhecimento de fala ou musica, assistente residenciais, próteses auditivas, sistemas de automatização, carros, entre outros.

É importante averiguar essas tecnologias emergentes tanto no sentido de inovação de produtos quanto na análise preditiva para detecção de prováveis avarias de amostras. Uma aplicação que ainda carece de estudos aprofundados é a classificação de padrões sonoros de compressores hermético utilizando algoritmos de inteligência artificial.

O compressor é um dos responsável pela circulação de fluído ao longo do sistema de refrigeração junto com o condensador, válvula de expansão e evaporador, este dispositivo torna possível o ciclo de refrigeração (**boabaid2017estudo**). Ele também é uma das fonte sonoras mais importantes desse conjunto, pois as energias acústicas e vibratórias produzidas no seu interior são transmitidas dele ao ambiente com níveis de ruído consideráveis. Dessa forma, identificar os ruídos anômalos dos típicos é de suma importância para indicar falhas de operação ou descrever a qualidade sonora do produto final.

Portanto, de todos os padrões sonoros de compressores herméticos, foram escolhidos alguns para treinar redes neurais para classifica-los. Essa será uma primeira análise para que em trabalho futuros seja possível reconhecer anomalias no sistema. Entre os sons escolhidos estão: *dripping* (gotejamento), *knock*, *start stop*, batida de mola, operação estacionária.

NO FINAL DO TRABALHO VOLTAR PARA DESCREVER TODOS OS SINAIS ESCOLHIDOS.

1.1 OBJETIVOS

A partir do exposto anteriormente e da necessidade de identificar padrões sonoros de compressores, o objetivo geral deste trabalho é determinar a eficácia diferentes algoritmos de inteligência artificial para reconhecer e classificar padrões de falhas a partir de sinais pressão produzidos por compressores herméticos de sistemas de refrigeração.

Objetivos específicos

- Investigar o funcionamento de diferentes algoritmos de redes neurais para detecção de padrões em sinais de sistemas de refrigeração;
- Implementar rotinas de treinamento e classificação de maneira organizada e intuitiva para utilização posterior do Laboratório de Vibrações e Acústica (LVA) e empresas parceiras;
- Investigar o comportamento dos algoritmos com diferentes tipos de entradas, tais como sinais de pressão sonora e aceleração.

1.2 ORGANIZAÇÃO DO TRABALHO

A estrutura de trabalho está organizada da seguinte forma, há (TANTOS) capítulos. O primeiro é uma introdução dos trabalho de padrão sonoros, das áreas que são beneficiadas por estas técnicas e do motivo de utilizar algoritmos de inteligência artificial para classificar em classes sinais sonoros de compressores.

No Capítulo 2 é apresentado o referencial teórico, no qual são descritas as principais teorias utilizadas para o trabalho, tendo como base a literatura e as normas referentes a cada área necessária para o desenvolvimento do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os principais aspectos teóricos utilizados neste trabalho.

2.1 DIFERENÇA ENTRE INTELIGÊNCIA ARTIFICIAL, APRENDIZADO DE MÁQUINASE APRENDIZADO PROFUNDO

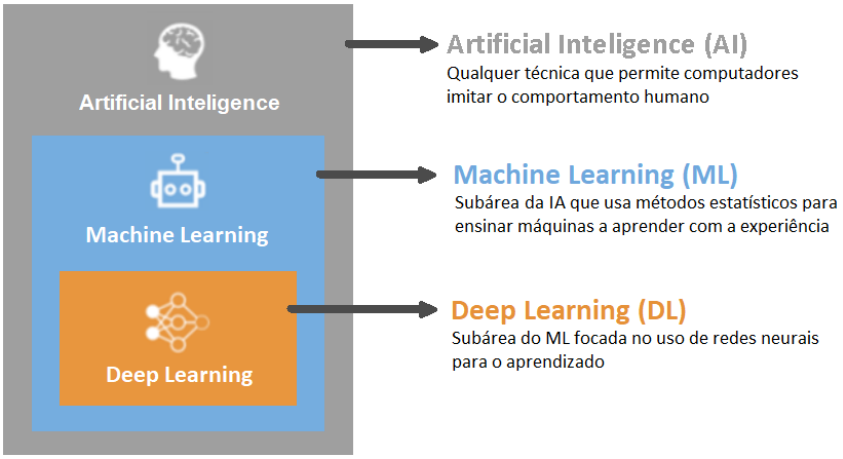
Os temas Inteligência Artificial, Aprendizado de Máquinas (ML)[1] e Aprendizado Profundo (DL)[2] são discutidos assiduamente em estudos e vendidos como diferencial de produtos. Portanto nesta seção será definida cada uma dessas áreas e subáreas, apresentando suas diferenças.

A Inteligência Artificial, se propõe elaborar dispositivos que simulem a capacidade humana de raciocinar, perceber, tomar decisões e resolver problemas. Logo, qualquer técnica que permita computadores simular o comportamento humano pode ser definida com uma AI (**DeepLearningArtigo**).

Aprendizado de Máquinasé um subárea de AI. Seu intuito é fazer que as máquinas aprendam sozinhas usando dados fornecidos e assim, consigam realizar previsões com uma dada incerteza. Dentre os algorítimos de ML existe um outro subconjunto conhecido como Aprendizado Profundo focado em Rede Neural (NN)[3] (**kim2017matlab**) .

As Redes Neurais são uma forma de modelar matematicamente sistemas de neurônios biológicos, e utiliza-los para resolver tarefas que outros tipos de algoritmos não conseguem realizar, por exemplo, classificação de imagens. Os subconjuntos de que compõem a AI podem ser visualizados na Figura **??**.

Figura 1 – Esquemático dos subconjuntos da AI.



Fonte – adaptado de (**site:RevolucaoIA**).

[1] Acrônimo do inglês para *Machine Learning*.
[2] Acrônimo do inglês para *Deep Learning*.
[3] Acrônimo do inglês para *Neural Network*.

A ideia de Aprendizado Profundo ser um "tipo" de Aprendizado de Máquinas é muito importante, e este é o motivo de realizar uma revisão sobre como AI, ML e DL estão relacionados. O Aprendizado Profundo, por ter resolvido com proficiência alguns problemas que desafiaram a Inteligência Artificial esta em destaque recentemente. Seu desempenho, a pesar de ser excepcional em muitos campos, também enfrenta limitações que derivam de seus conceitos fundamentais os quais foram herdados de seu antecessor o ML. Como uma subárea do aprendizado de máquina, o aprendizado profundo não pode evitar os problemas base que o ML enfrenta. Por isso que é necessário apresentar o Aprendizado de Máquinas antes de discutir o conceito de Aprendizado Profundo, o que será feito na seção **??**, porque o ML é alicerce do DL.

2.2 APRENDIZADO DE MÁQUINAS

Uma outra forma de compreender as técnicas de Aprendizado de Máquinas, é entendê-lo como método de análise de dados que visa automatizar a construção de modelos analíticos. Seus algoritmos são capazes de aprender iterativamente com os dados, possibilitando que computadores encontrem *insights* sem serem explicitamente programados sobre onde e o que procurar nas informações (**CollegeL30:online**).

Posto isso, nesta seção serão apresentadas as terminologias básicas necessárias em Aprendizado de Máquinas, como se dá o aprendizado e de que forma seus erros são reduzidos. Também será discutida a diferença entre tarefas supervisionadas e não supervisionadas, assim como a distinção entre classificação e regressão. No final, serão descritas as métricas de avaliação de performance de modelos de classificação.

2.2.1 Terminologias Fundamentais de ML e Introdução a Conceitos Básicos

Nesta subseção serão definidas terminologias fundamentais de Aprendizado de Máquinase Aprendizado Profundoobtidas em (**MachineL6:online**). A partir disso, é possível determinar conceitos como: tipos de tarefas, diferenças entre treinamento e inferência e assim por diante.

Há dois tipos de tarefas em Aprendizado de Máquinas, as supervisionadas e as não supervisionadas. As tarefas supervisionadas aprendem como a combinar inputs para produzir previsões úteis sobre dados nunca antes vistos, ou seja, se tem conhecimento sobre qual é a saída para uma dada entrada (**singh2016review**). Já no caso de não ter noção prévia do que espera-se como inferência das redes, utiliza-se modelos não supervisionados.

Duas terminologias muito importantes em Aprendizado de Máquinas supervisionada são os rótulos[4] e os atributos[5]). Os rótulos, são as classes e também o que se espera

[4] Em inglês *labels*.
[5] Em inglês *features*.

como predição do modelo, representado pela variável y. As atributos, são as variáveis de entrada (valores que caracterizam as amostras), representado por x.

Uma das formas de indicar o quão sofisticado é um projeto de aprendizado de máquinas é através da quantidade de atributos. Ou seja, redes que recebem como entradas poucos recursos são mais simples que as com múltiplas entradas.

Do conjunto de amostras, um exemplos é uma instância particular de dados do vetor de atributos $\overline{\mathbf{x}}$ e são classificados em duas categorias: exemplo rotulado[6] e exemplo não rotulado[7].

Um exemplos rotulados incluí tanto atributo(s) como o rótulo(s). Ou seja:

exemplos rotulados: {atributo, rótulo}: (x, y)

Já um exemplos não rotulados contem apenas as atributo(s) sem o rótulo(s). Como representado a seguir:

exemplos não rotulados: {atributo, ?}: (x, ?)

Os problemas mais comuns resolvidos com este tipo de ML são os de regressão e classificação. Uma diferença entre eles, é que a regressão prediz valores contínuos, enquanto a classificação prediz valores discretos. Outra característica importante ocorre na forma de avaliar performance, com métricas específicas para cada um. Elas serão descritas na seção **??**.

De tudo o que foi exposto até agora, é possível concluir que modelos de classificação e regressão necessitam de um banco de dados prévio, visto que utilizam exemplos com e sem rótulos, ou seja, são tarefas supervisionadas. Já no caso de aprendizado não supervisionado, que não se sabe a saída, exemplos comuns de aplicação são: problemas como detecção de anomalias (não se tem conhecimento do problema); agrupamentos individuais [8]; e redução de dimensionalidade (**gentleman2008unsupervised**). Como este trabalho será sobre classificação de padrões sonoros, as técnicas de aprendizado supervisionado não serão abordadas.

Para finalizar as definições básicas, necessita-se determinar o que é modelo, isso do ponto de vista de tarefas supervisionadas. Ele tem duas fases, que são:

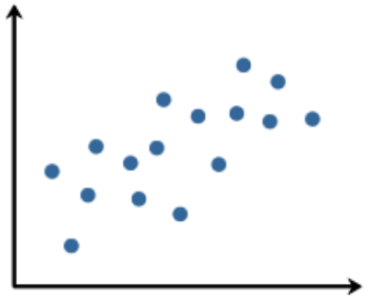
- Treinamento: fase na qual o modelo aprende, ou seja, ele receberá exemplos rotulados e será permitido que ele gradualmente encontre a relação entre x e y;
- Inferência: momento que será aplicado o modelo treinado em exemplos não rotulados, isto é, ele faz predições uteis (y').

[6] Em inglês *labeled examples*.
[7] Em inglês *unlabeled examples*.
[8] Em inglês *cluster*.

2.2.2 Treinando modelo simples para compreender Aprendizado de Máquinas

A Figura **??** apresenta uma pequena amostra de dados observados, e a partir delas deseja-se encontrar uma relação entre y, variável de interesse dependente que se quer predizer, e x a variável independente.

Figura 2 – Pequena amostra de dados quaisquer.



Fonte – (**MachineL6:online**)

Pela análise da Figura **??** é possível dizer que, uma reta consegue inferir o comportamento entre x e y, mesmo que não passe exatamente por todos os pontos, como a Figura **??** demonstra. Matematicamente é descrita pela equação **??** na convenção estabelecida em ML e adicionando o subíndice que indica a possibilidade de ter mais de uma dimensão. Para inferir y, basta substituir os valores de x neste modelo (reta) (**CollegeL30:online**).

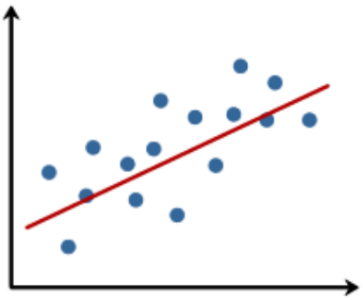
$$y' = b + w_1 x_1; \tag{1}$$

em que:

- y' é o rótulo predito [9] (saída desejado);
- b é o bias (intercepção no eixo y);
- w_1 é o peso da atributo 1. Peso tem o mesmo conceito que a inclinação (m) tem na equação tradicional da reta;
- x é o atributo (entrada).

[9] Em inglês *predicted label*.

Figura 3 – Regressão linear que prever o comportamento da pequena amostra de dados quaisquer.



Fonte – (**MachineL6:online**)

O exemplo proposto considerar somente uma dimensão de atributos como entrada, ao contrário de modelo mais sofisticado que utilizam muito mais, e com isso, possuem mais pesos (w). Logo, há um peso para cada atributo em separado, como descreve a equação **??**.

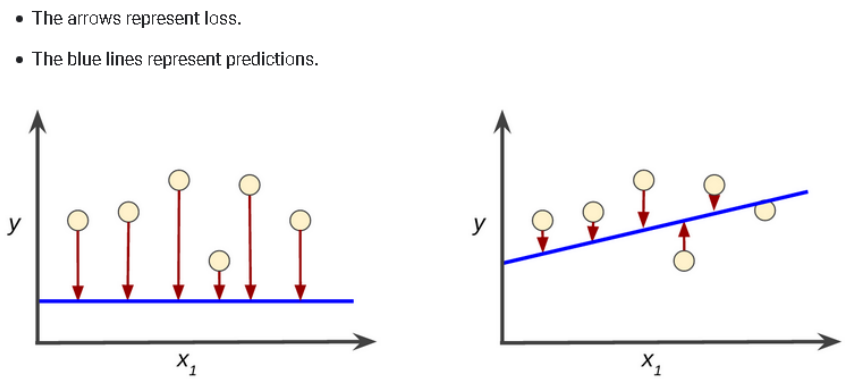
$$y' = b + w_1 x_1 + w_2 x_2 + ... + w_n x_n. \tag{2}$$

Para sintetizar, os dados foram aplicados em um algoritmo de Aprendizado de Máquinase como resultado obteve-se um modelo, que neste caso é uma reta. Mas é gerada uma a pergunta: como o treinamento realmente acontece? Ele ocorre ao determinar bons valores para o bias e todos os pesos utilizando exemplos rotulados. Para isso, o algoritmo examina varias amostras, e compara a predição feita pela equação criada, com o valor real, buscando encontrar um modelo que minimize a perda entre y e y'.

A perda é também conhecida como erro, que é dado por um número que indica o quão boa foi a inferência de um exemplo único. Ele pode ser visto como uma penalidade dada ao modelo por fazer uma predição ruim. Se y' for igual a y, a perda será zero, caso contrário, será um valor maior que zero e grande.

A Figura **??** apresenta dois modelos, sendo que o da esquerda possui um erro alto. Uma forma de concluir isso, é através da distância muito maior entre a linha azul (equação) e o ponto (valor verdadeiro). No gráfico da direta esse espaço é pequeno, mostrando que a perda é menor.

Figura 4 – O gráfico da esquerda ilustrar um modelo com alta perda e o da direita com baixa perda.



Fonte – (**MachineL6:online**)

Portanto o objetivo dos algoritmos de ML e DL é treinar o modelo para encontrar os valores de w_i e b que minimize essa diferença ao longo de todos os exemplos. A equação mais comum usada para esse cálculo é erro quadrático[10] , entretanto, ela é usada a partir do valor médio, denominado Erro Médio Quadrático (MSE)[11], e descrito pela equação **??**(**TotalTen29:online**).

$$MSE = \frac{1}{N} \sum_{(x,y)\in D} (y - y')^2 \tag{3}$$

em que:

- (x, y) é um exemplo no qual
  - x é o conjunto de atributos que o modelo usa para realizar predições;
  - y é o rótulo do exemplo.
- y' é a função de pesos e bias em combinação com o conjunto de atributos x;
- D é um conjunto de dados contendo muitos exemplos rotulados, nos quais (x, y) são pares;
- N é o número de exemplos em D.

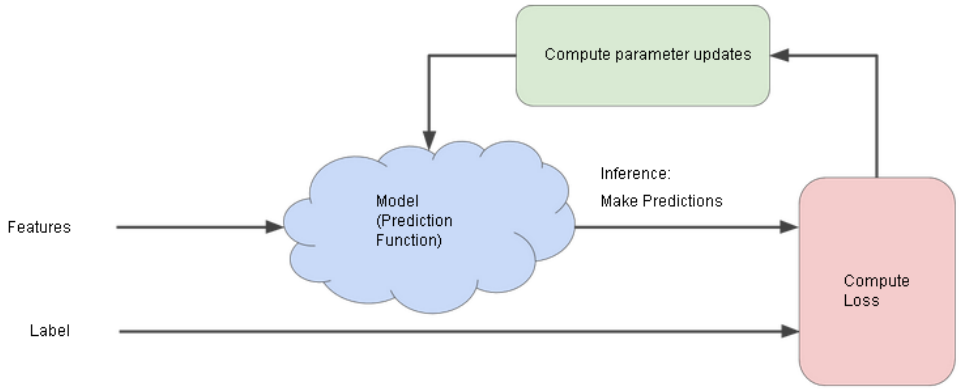
2.2.3 Redução de Perda

O diagrama apresentado na Figura **??**, demonstra como o modelo de aprendizado de máquina reduz a perda iterativamente para aprender. Esse processo começa com o algoritmo gerando um "palpite", ou seja, um valor para b e w_1. Então, o erro é calculado através da equação **??**.

[10] Em inglês *squared loss*.
[11] Acrônimo do inglês para *Mean square Error*.

As entradas, na estimativa da perda, são o conjunto de rótulos, y, e a inferência, y', realizada pelo primeiro modelo utilizando o conjunto de atributos. Após é repetido este processo, entretanto com um novo palpite e da comparação entre os resultado obtidos, o algoritmo escolhe qual das tentativa obteve o menor erro.

Figura 5 – Abordagem iterativa para o treinamento do modelo através da redução do erro.



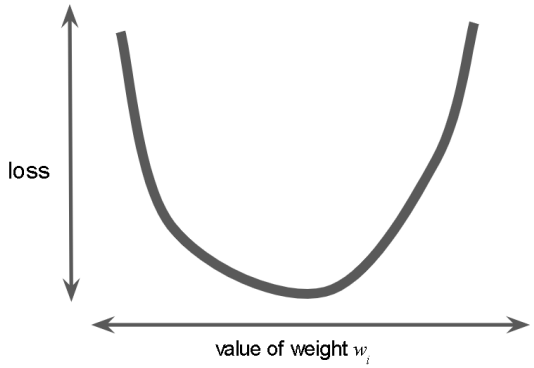
Fonte – (**MachineL6:online**)

O diagrama **??** tem uma etapa chamada de *Compute Loss*, que representa o momento no qual o algoritmo determina o erro. Na sequência, há o processo *Compute Parameter Updates*, que caracteriza a rede e examina os valores calculados pela função de perda e escolhe como será a atualização deles. Esse procedimento de aprendizado continua a cada iteração até que o algoritmo descubra os parâmetros do modelo com a menor perda possível.

O ponto no qual o treinamento é interrompido é denominado ponto de corte, e é estabelecido a partir de um critério de parada. Normalmente determina-se esse momento na época na qual a perda geral para de mudar ou pelo menos mude de forma extremamente lenta, quando isso acontece, o modelo convergiu.

Foi usado como exemplo um modelo de regressão linear, se para este caso forem calculados todos os valores possíveis de perda para w_1, o resultado seria sempre uma parábola com concavidade para cima. Ela é apresentada no gráfico convexo da Figura **??**.

Figura 6 – Problemas de regressão linear geram gráficos de perda vs w_1 (peso) convexo.

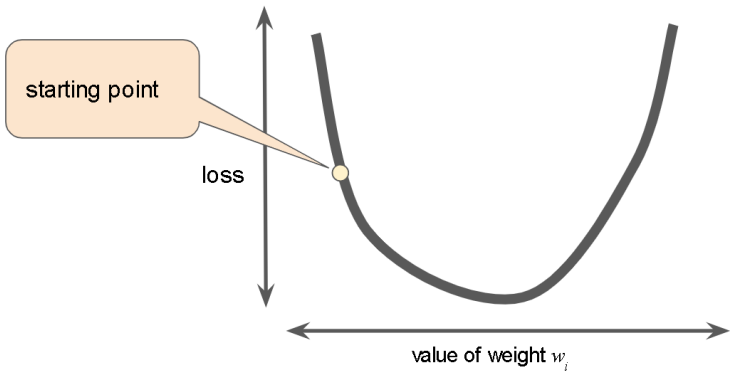


Fonte – (**MachineL6:online**)

Esse problema é muito simples, pois uma parábola possuí apenas um mínimo, ou seja, um ponto cuja inclinação é exatamente igual a 0. Entretanto, por mais fácil que seja visualizar o ponto onde a função perda converge em uma parábola, ainda é um processo inviável e ineficiente levantar todos os valores para posteriormente encontrar o mínimo.

Um método melhor e mais popular, no aprendizado de máquinas, é o gradiente descendente. Essa técnica também começa com um palpite para b e w_1. O ponto de partida não importa muito, assim, muitos algoritmos simplesmente configuram para iniciar em 0 ou em um valor aleatório, como exemplo, a Figura **??**.

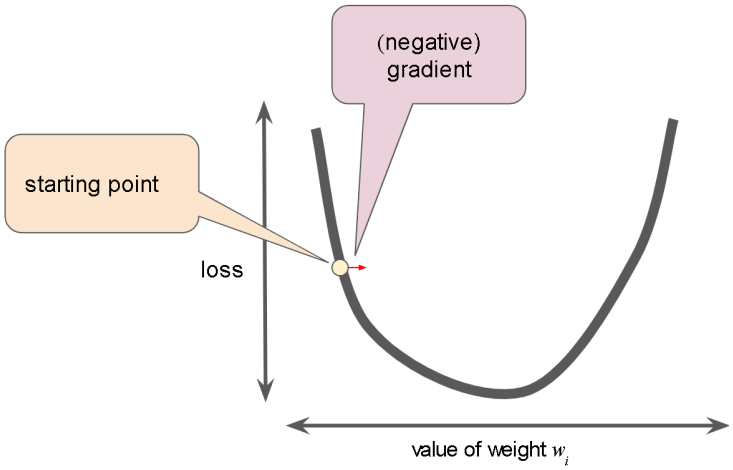
Figura 7 – Ponto de partida aleatório utilizado pelo gradiente descente para minimizar a perda.



Fonte – (**MachineL6:online**)

O algoritmo calcula o erro no ponto inicial pela derivada parcial, que informa o quanto a função muda quando é feita uma pequena perturbação em uma variável com dado peso. Com base nesta variação, o gradiente consegue predizer se a inclinação da derivada é positiva ou negativa. Como o intuito é encontrar o mínimo, a rede seguirá o gradiente negativo, conhecido como gradiente descendente. Quando há vários pesos, o gradiente é um vetor de derivadas parciais em relação aos pesos. Como o gradiente é um vetor, ele tem uma direção e uma magnitude e seu resultado sempre aponta na direção do gradiente negativo para reduzir a perda o mais rápido possível, como mostra a Figura **??**.

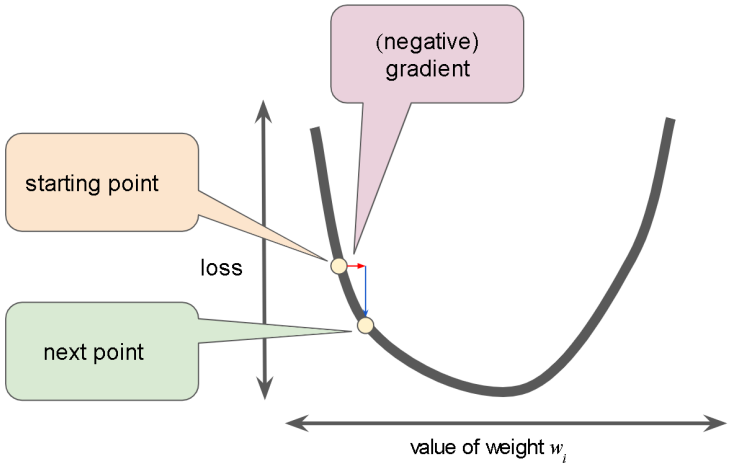
Figura 8 – A descida do gradiente depende de gradientes negativos.



Fonte – (**MachineL6:online**)

Para determinar o próximo ponto ao longo da curva de perda, o algoritmo adiciona uma fração da magnitude do gradiente ao ponto inicial, conforme mostrado na Figura **??**. Esse processo é repetido, chegando cada vez mais perto do mínimo.

Figura 9 – Acréscimo da magnitude do gradiente no valor do ponto inicial move para o próximo ponto da curva de perda.



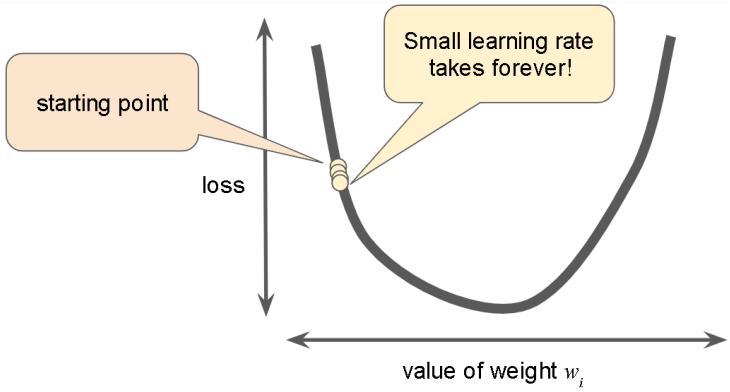
Fonte – (**MachineL6:online**)

Entretanto, não é somente somado ao ponto inicial o valor da magnitude. Antes disso, ela é multiplicada pala taxa de aprendizado, também chamado de tamanho do passo. A taxa de aprendizado é um hiper parâmetro, ou seja, uma das variáveis que controla o próprio processo de treinamento. Esse conjunto soma e multiplicação que realmente dita o próximo ponto de análise na função de perda. Para exemplificar, pode-se considerar uma magnitude de 2,5 e uma taxa de 0,01. O algoritmo escolherá o próximo ponto a uma distancia de 0,025 do valor anterior.

A taxa de aprendizado trás algumas informações importantes: se ela for muito pequena, o treinamento demorará muito como mostra a Figura **??**. Por outro lado, se ela

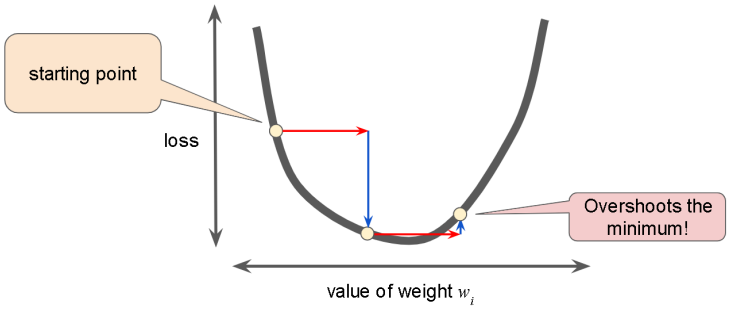
for muito grande, o ponto irá saltar perpetuamente ao acaso perto do vale sem convergir, demonstrado pela Figura **??**.

Figura 10 – Se a taxa de aprendizagem for muito pequena a proximidade entre os passos é maior.



Fonte – (**MachineL6:online**)

Figura 11 – A taxa de aprendizagem é muito grande.



Fonte – (**MachineL6:online**)

Existe uma taxa de aprendizagem *Goldilocks*[12] para cada problema de regressão, que descreve o quão plana é a função de perda. Essa informação permite aumentar com segurança a taxa de aprendizado, se for de conhecimento prévio que o gradiente é pequeno. Com essa compreensão que o valor da magnitude do gradiente é pequeno, é possível configurar um tamanho de passo maior, e dessa forma, que a perda encontre o mínimo no menor número de etapas. Já a taxa de aprendizagem ideal em uma dimensão é o inverso da segunda derivada de $f(x)$ em x e para 2 ou mais dimensões é o inverso da matriz das derivadas parciais secundárias.

Uma terminologia importante, no gradiente descendente, é lote[13] definido como o número total de exemplos usados para calcular o gradiente em uma única iteração. Quando é dado como entrada para o cálculo da inclinação o conjunto inteiro de dados, considera-se que foi utilizado um lote. Entretanto, eles geralmente contêm um grande

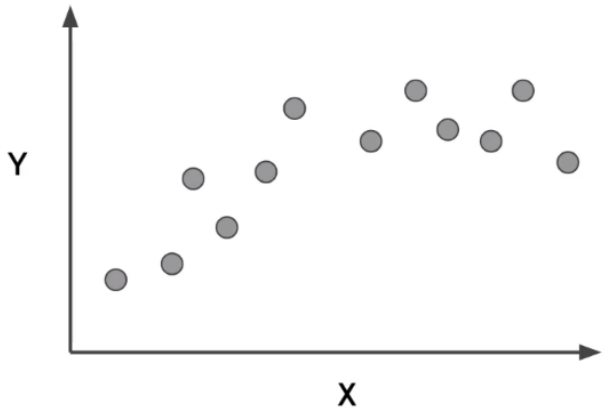
[12] Em Redes Neurais é a taxa de aprendizado ideal.
[13] Em inglês *batch*.

número de atributos, ou seja, um lote pode ser enorme, fazendo com que até mesmo uma única iteração demore muito para ser computada.

2.2.4 Generalização em modelos

Para descrever o que significa um modelo sobre-ajustado[14], um modelo sub-ajustado[15] e o porquê de separar os dados em subconjuntos, serão usadas outras amostras com somente um atributo, apresentadas na Figura **??**. Assim como no exemplo da seção anterior **??**, o objetivo é desenvolver um modelo para prever os rótulos e a partir dele discutir esses processos (**kim2017matlab**).

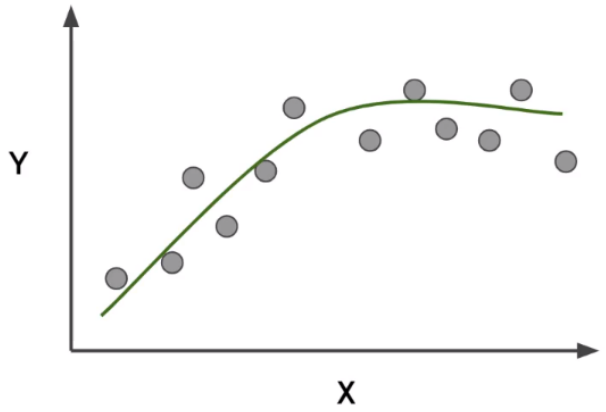
Figura 12 – Pequena amostra de dados com somente uma dimensão atributos.



Fonte – (**Complete79:online**)

É considerado como um bom modelo aquele que tenta se ajustar à tendência geral do conjunto de dados real, e não se encaixar perfeitamente neles. A Figura **??** exibe esse comportamento de generalizar seus resultados.

Figura 13 – Comportamento esperado para um bom modelo.

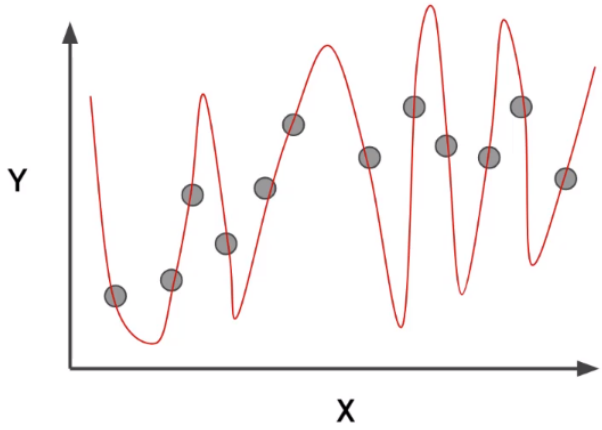


Fonte – (**Complete79:online**)

[14] Em inglês *overfitting*
[15] Em inglês *underfitting*

A Figura **??** apresenta o comportamento de um modelo com sobre-ajuste utilizando as amostras da Figura **??**. Nela é possível ver que a curva passa exatamente em todos os pontos do gráfico. Quando isso acontece, ele se torna mais complexo do que necessário, seu erro de treinamento é extremamente baixo ou, como neste caso específico, igual à zero.

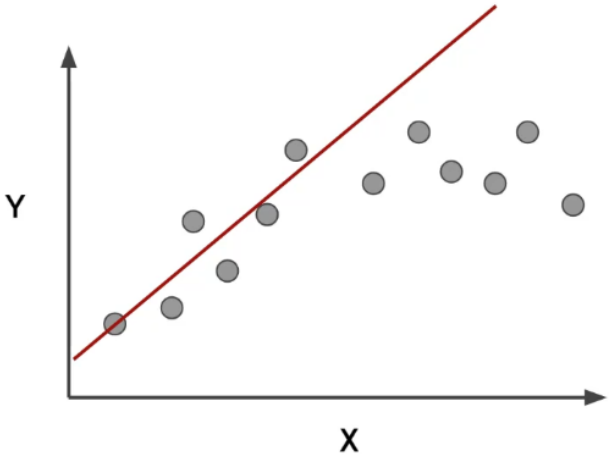
Figura 14 – Modelo considerado ruim com a presença de sobre-ajuste.



Fonte – (**Complete79:online**)

O oposto do sobre-ajuste é o sub-ajuste, que ocorre em modelos excessivamente simples os quais não conseguem capturar a tendência dos dados para prever seu comportamento. Como resultado, frequentemente, apresentam baixa variança e alto bias. Um exemplo é apresentado na Figura **??**, ele também foi baseados nas amostras da Figura **??**.

Figura 15 – Exemplo de um modelo considerado ruim por apresentar sub-ajuste.



Fonte – (**Complete79:online**)

Infelizmente, o modelo não consegue considerar todas as informações possíveis, uma vez que ele só recebe uma amostra de dados, denominado conjunto de treinamento. Isso acarreta em um problema, se ocorrer sobre-ajuste ou sub-ajuste nos exemplos atuais, como confiar que o modelo também fará boas previsões sobre exemplos nunca antes vistos? Esse

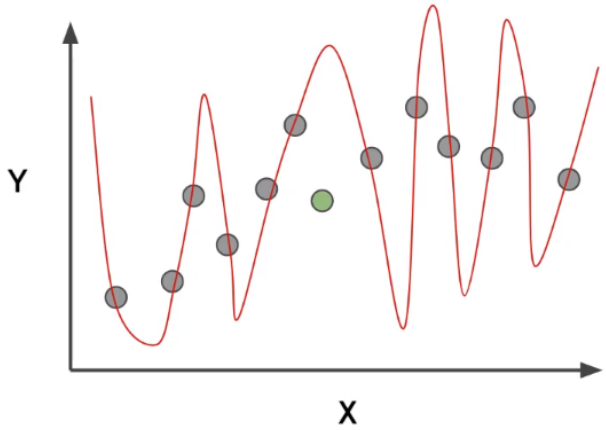
é o motivo de separar os conjuntos de dados em mais de um subconjunto. As separações mais utilizadas em ML são:

- Em duas divisões denominadas: treino e teste;
- Em três divisões denominadas: treino, teste e validação.

O conjunto de teste deve atender à duas condições: ser grande o suficiente para produzir resultados estatisticamente significativos e ser representativo para o conjunto de dados como um todo. Em outras palavras, não pode ser escolhido um conjunto de teste com características diferentes do conjunto de treinamento.

Supondo que o conjunto de teste atenda às duas condições anteriores, independente de qual divisão for escolhida, haverá um subconjunto para testar o modelo já treinado. Dessa forma, é possível avaliar se há sobre-ajuste ou sub-ajuste a partir dos resultados do cálculo de erro. Isso porque esse valor será pequeno para o conjunto de treinamento, mas para as novas amostras de teste será muito maior. A Figura **??** apresenta a um ponto de teste.

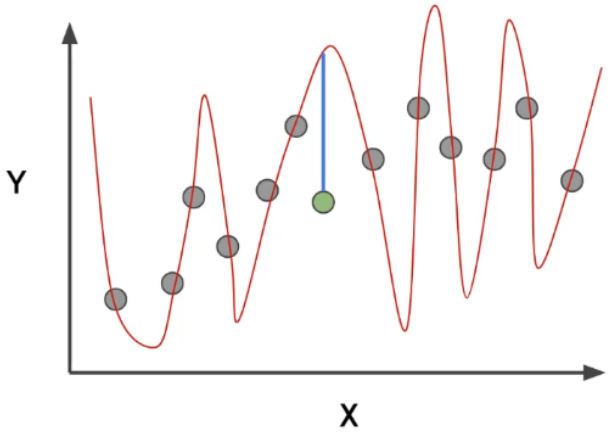
Figura 16 – Novo valor teste aplicado a um modelo com sobre-ajuste.



Fonte – (**Complete79:online**)

A noção da grandeza do erro pode ser visualizada na distância entre o ponto, valor real, e a curva, equação de inferência, como mostras a Figura **??**. Lembrando que, os valores de teste não podem ser utilizados durante o treinamento, garantindo assim, que o modelo não tenha visto antes estes dados.

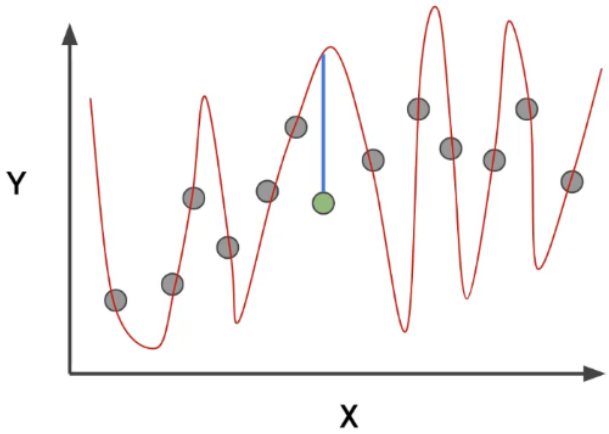
Figura 17 – Apresenta distância entre o novo ponto e a predição do modelo, ou seja, uma visualização do erro.



Fonte – (**Complete79:online**)

Ao considerar a Figura **??**, nela esta representado o mesmo o modelo da Figura **??**, considerado bom, e o valor utilizado para teste na Figura **??**. Por mais que o modelo seja muito simples e que não faça um trabalho perfeito (algumas previsões estão erradas), ele se sai tão bem nos dados de treinamento quanto no ponto de teste, ou seja, a equação não se ajusta demais nos dados de treinamento.

Figura 18 – Mostra que a predição do modelo bom não seria perfeita, mas apresentaria um erro menor do que a com sobre-ajuste.



Fonte – (**Complete79:online**)

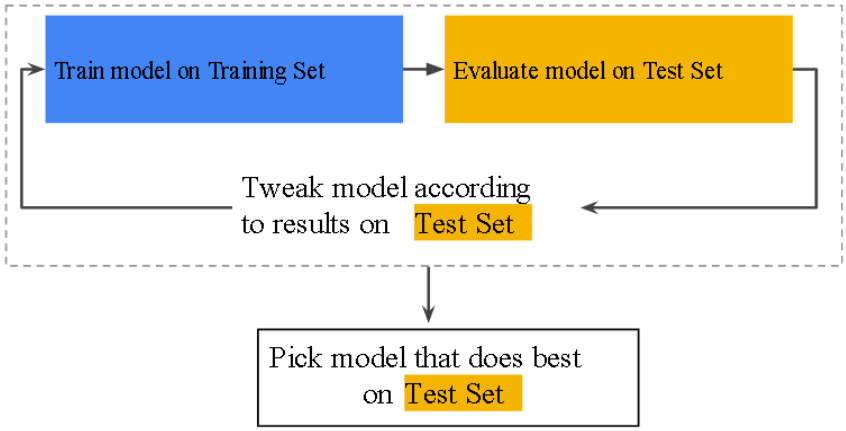
Um bom desempenho no conjunto de teste, indica o quanto que se pode confiar nos resultados obtidos com novos dados. Entretanto, isso só é válido se o subconjunto de teste for grande o suficiente; e se não houver "trapaça", isto é, se o mesmo conjunto de teste não for usado repetidamente, ou no lugar do de treinamento. Esse último, pode-se ser concluído por métricas de avaliação com valores surpreendentemente bons, por exemplo, uma alta precisão pode indicar que os dados de teste vazaram para o conjunto de treinamento.

Outra forma de garantir a qualidade dos resultados é eliminando exemplos que acarretam problemas de aprendizado para o algoritmo. Como no caso de um modelo treinado para classificar *e-mails* como sendo spam ou não, o qual as amostras foram distribuídos em treino e teste. Indiferente do resultado, o esperado é obter valores para as métricas menores no conjunto de teste, isso porque o modelo não o usou como base de informações.

Na hipótese dele atingir a mesma acurácia nos dois conjuntos, treinamento e teste, gera-se um alerta. Logo, os dados devem ser verificados novamente, e se por acaso forem descobertos *e-mail* duplicados entre os dois subconjuntos, a abordagem ideal é remover essas entradas e redividir as amostras. Do contrário, não será possível afirmar o quão bem o modelo consegue se generalizar realizando inferências corretas sobre novos dados.

O fluxo de trabalho da divisão em dois conjunto, treino e teste, é ilustrado na Figura **??**. Nela, a etapa *"Tweak model"* descreve o momento no qual pode-se ajustar a rede, e as possibilidades vão desde alterar a taxa de aprendizado, adicionar ou remover recursos, até projetar um modelo completamente novo do zero. No final deste fluxo de trabalho, escolhe-se o modelo que se sai melhor no conjunto de teste.

Figura 19 – Fluxo de trabalho para separação do conjunto de dados em treino e teste.



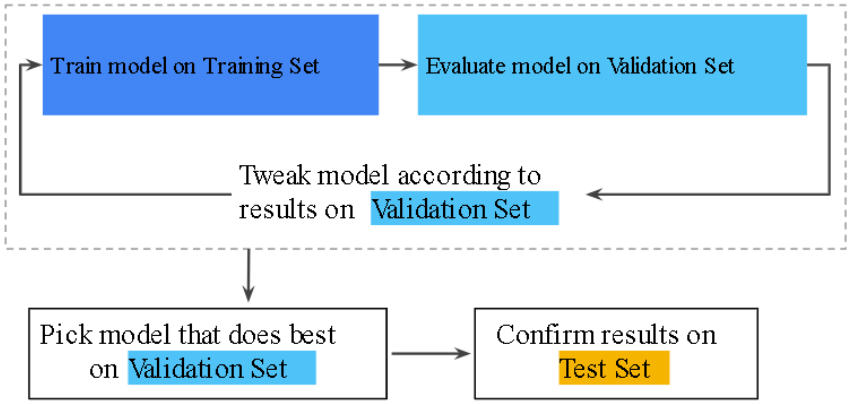
Fonte – (**MachineL6:online**)

Entretanto, deve-se observar que esse procedimento é uma abordagem simplificada, pois a divisão é feita somente em dois blocos, o que não é adequado para obter o desempenho final do modelo. Isso porque é possível atualizar os parâmetros repetidas vezes após avaliar os resultados no conjunto de teste. Por causa disso, os dados costumam ser divididos em três conjuntos: treinamento, validação e teste.

Essa abordagem reduz significativamente as chances de ocorrer sobre-ajuste, pois em resumo, as separações das amostra em 3 blocos são usadas da seguinte forma: com o conjunto para o treinamento o modelo consegue aprender as relações entre atributos e rótulos e assim ajustar-se aos dados. As amostras de validação são aplicadas à rede para verificar o desempenho e ele é a referência nos ajuste do modelo (adição de neurônios

ou camadas, mudança na arquitetura real da rede etc.). Repete-se esse processo até ter resultados que satisfaçam condições específicas. Só então é possível avaliar a verdadeira performance do modelo com a ajuda da terceira divisão de dados, a de teste. Esse bloco é aplicado em métrica específicas para determinar o comportamento final, esperado em situações práticas. É importante observar que após executar o modelo com os dados de teste não será mais possível voltar e refinar os hiper-parâmetros da rede. O fluxo de trabalho da divisão em três conjunto é ilustrado na Figura **??**.

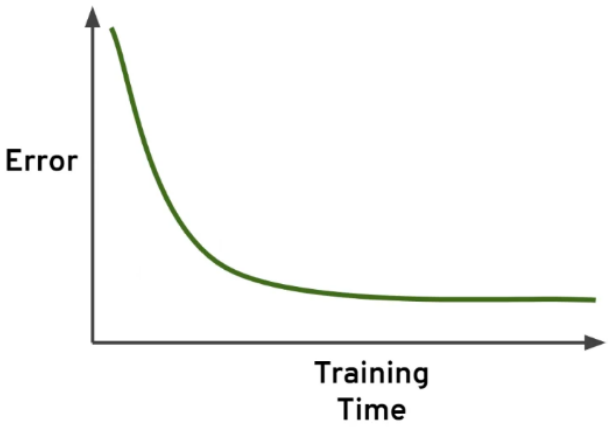
Figura 20 – Fluxo de trabalho para separação do conjunto de dados em treino, teste e validação.



Fonte – (**MachineL6:online**)

Até agora, as explicações foram realizadas com base em problemas de uma única atributos, pois são de fácil visualização. Entretanto, isso não ocorre frequentemente em situações práticas, em que a maior parte do tempo, trabalha-se com conjuntos de dados multidimensionais, o que exige uma abordagem diferente. Para discutir o melhor procedimento, pode-se considerar a Figura **??** que apresenta a relação entre medição do erro e tempo de treinamento de um modelo considerado ideal.

Figura 21 – Modelo ideal em relação ao erro versus tempo de treinamento.

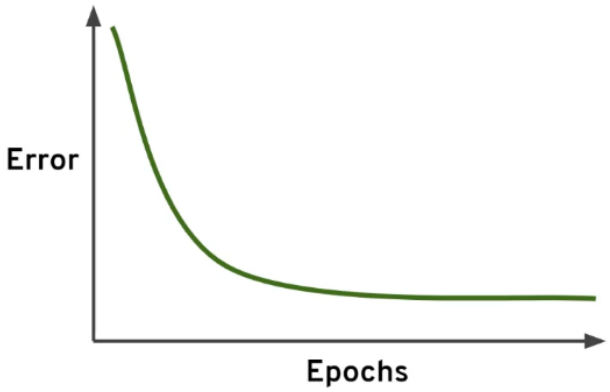


Fonte –

Ao aplicar os dados de treinamento pela primeira vez no modelo, o valor de perda obtido será grande. Isso ocorre porque o algoritmo não teve acesso a essas informações antes e não ajustou quaisquer parâmetros internos como hipótese inicial. No entanto, conforme ocorre o aprendizado, ou seja, quanto mais tempo de treinamento, espera-se que o erro diminua até que se estabilize e convergindo para algum tipo de mínimo.

No caso de redes neurais, este tempo de treinamento tem uma denominação específica, época[16]. Essencialmente, uma época ocorre quando a totalidade dos dados de treinamento passa pelo modelo. Esse procedimento acontece várias vezes até que se obtém um bom modelo, como mostra a Figura **??**.

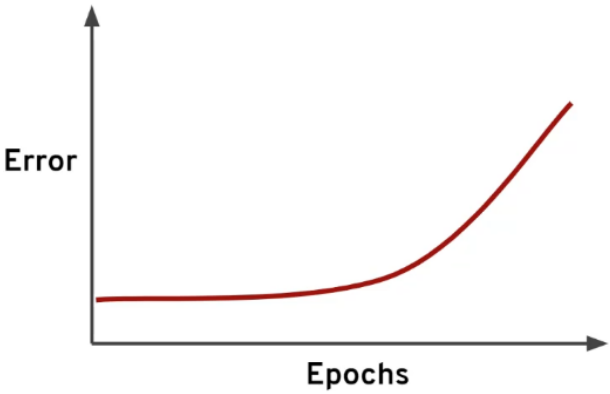
Figura 22 – Modelo ideal em relação ao erro versus épocas.



Fonte – (**Complete79:online**)

Um modelo péssimo apresenta erros crescentes a medida que o tempo passa, ou seja, o modelo não aprende a cada época. Esse comportamento é demonstrado pela Figura **??**.

Figura 23 – Modelo ruim em relação ao erro versus épocas.



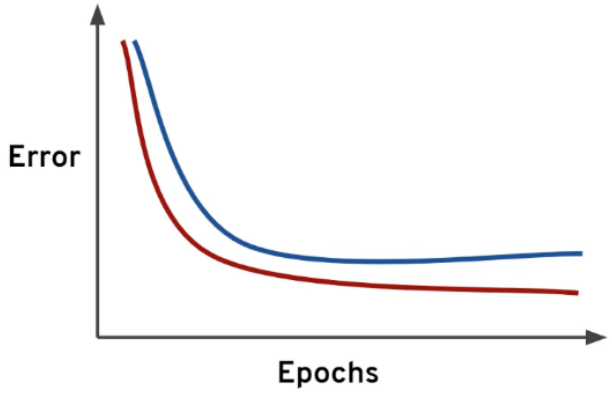
Fonte – (**Complete79:online**)

Portanto, o estudo da simplicidade/complexidade dos modelos é importe para compreender a relação entre o desempenho deles nos conjuntos de treino e de teste/validação,

[16] Em inglês epochs

sendo possível visualizar esse comportamento conforme as épocas passam. A Figura **??** mostra os resultados ideais obtidos durante o treinamento, curva em vermelho, e de o teste, curva em azul.

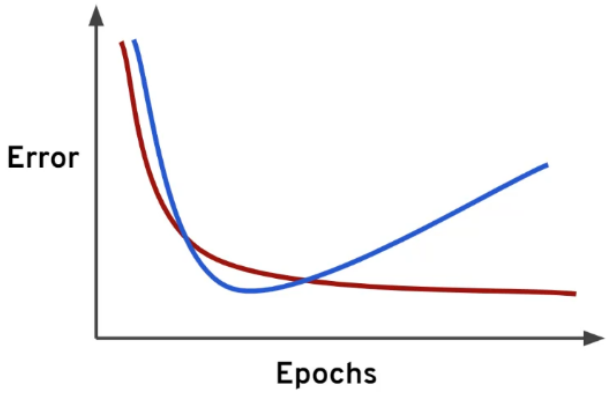
Figura 24 – Modelo ideal de erro de treino e de teste em relação ao as épocas.



Fonte – (**Complete79:online**)

No comportamento ideal, tanto o conjunto de treino como o de teste tem seu erro reduzido a medida em que as épocas passam. O resultado deles é similar, entretanto, no teste espera-se valores de perda maiores. Porém, se há sobre-ajuste, ao testa-los, tanto com os dados de teste como com os de validação, o resultado será ruim o que pode ser analisado na Figura **??**.

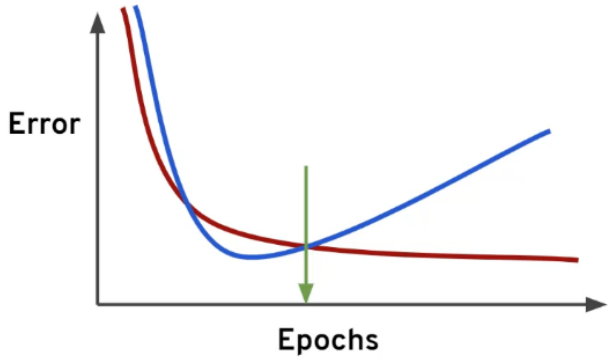
Figura 25 – Modelo ruim de erro de treino e de teste em relação ao as épocas.



Fonte – (**Complete79:online**)

O exemplo da Figura **??** é uma boa indicação de complexidade excessiva nos dados de treino. Nela é possível visualizar o ponto no conjunto de teste, no qual o erro começa a aumentar ao invés de reduzir. Esse ponto define quando parar de treinar o modelo, pois, para as épocas a seguir começará o sobre-ajuste e os resultados para novos dados serão piores (**CollegeL30:online**). Esse ponto de corte é apresentado na Figura **??**.

Figura 26 – Ponto de corte no tempo de treinamento para evitar sobre-ajuste.



Fonte – (**Complete79:online**)

2.2.5 Avaliação de desempenho - Métricas de erro de classificação

Esta seção descreverá as métricas de avaliação de desempenho para problemas de classificação. Elas são utilizadas no final do processo de aprendizado, após o treinamento, com os dados do conjunto de teste e buscam avaliara eficiência do modelo.

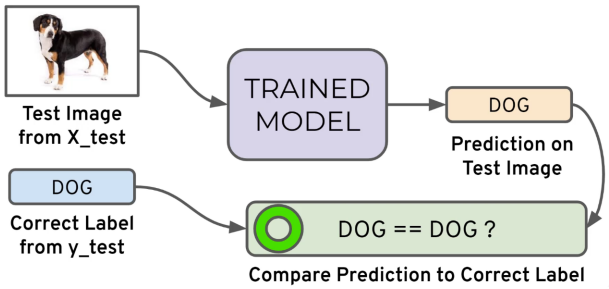
As principais métricas de classificação estudadas por este trabalho são: acurácia, *recall*, precisão e F1-Score. Antes de descrever-las, será explicado o raciocínio por trás delas e como realmente funcionam no mundo real.

Em uma típica tarefa de classificação, o modelo pode alcançar apenas dois resultados: ou ele realizou uma previsão correta; ou ele realizou uma previsão incorreta. Todas as métricas de classificação derivam dessa ideia, e felizmente, é possível expandi-la para situações em que há várias classes. Entretanto, para explicar o propósito delas será considerado uma classificação binária.

A partir de um modelo que tem como objetivo classificar imagens como sendo de cachorro ou de gato, será descrito como determinar o desempenho real/final. O conjunto de teste possuí atributos, x_{test}, imagens de cachorro e gato, e seus respectivos rótulos, y_{test}, o nome cachorro e gato.

As imagens são dadas como entrada no modelo já treinado, que faz uma previsão. Esse resultado será uma predição correta, por exemplo, se a rede inferir cachorro para uma imagem de cachorro, como mostra a Figura **??**.

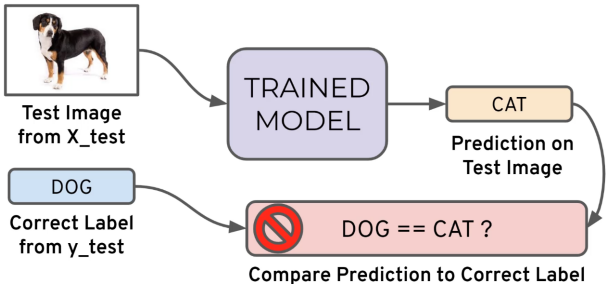
Figura 27 – Predição correta da rede neural.



Fonte – (**Complete79:online**)

Porém, ela estará incorreta quando seu saída for gato e a imagem é de um cachorro, resultado apresentado pela Figura **??**.

Figura 28 – Predição incorreta da rede neural.

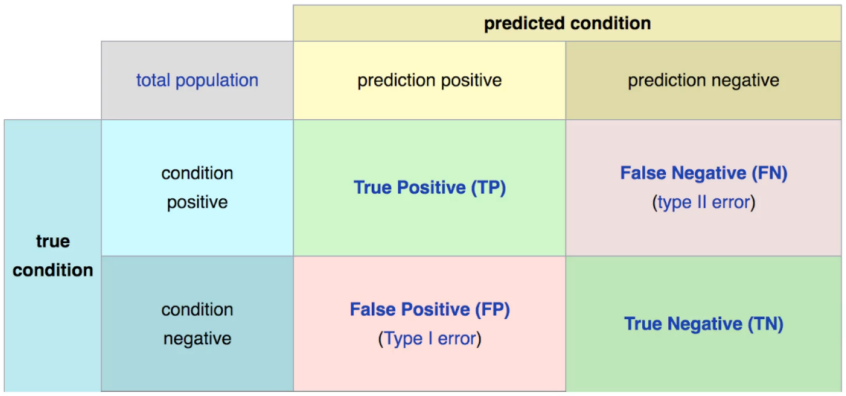


Fonte – (**Complete79:online**)

Ao repetir esse processo para todas as imagens do banco de dados de teste, tem-se uma contagem das correspondências corretas e incorretas. É importante dizer que, no mundo real, nem todas as correspondências incorretas ou corretas têm o mesmo valor. Isso significa que uma única métrica não é o suficiente para caracterizar o desempenho do modelo. Portanto, são utilizados as quatro métricas e organiza-se os valores previstos em comparação com os reais no que é conhecido como matriz de confusão.

A matriz de confusão apresenta os resultados de classificações corretas versus incorretas, como mostra a Figura **??**. As condições verdadeiras (*True Condition*) que indicam qual o rótulo correto/esperado para um dada inferência. Essas condições possuem duas subcategorias, positiva e negativa. Em outras palavras, é realmente cachorro versus não é, conforme Figura **??**. As inferência realizadas pelo modelo são apontadas na condições de predição (*Predicted Condition*), que também podem assumir duas categorias, positiva e negativa.

Figura 29 – Matriz de confusão na qual computa predições em relação do que se esperava como *output*.



Fonte – (**Complete79:online**)

Como é possível analisar, os elementos da diagonais da matriz apresentam as previsões corretas para diferentes classes, enquanto os dados fora da diagonal mostram as amostras que foram mal classificadas. Assim, as possibilidades de resultados de classificações são:

- Verdadeiro Positivo (*True Positive* - TP): o modelo prediz corretamente a classe positiva;
- Verdadeiro Negativo (*True Negative* - TN): o modelo prediz corretamente a classe negativa;
- Falso Positivo (*False Positive* - FP): o modelo prediz incorretamente a classe positiva (erro tipo 1);
- Falso Negativo (*False Negative* - FN): o modelo prediz incorretamente a classe negativa (erro tipo 2).

Considerando como classe positiva a imagem de cachorro e a negativa a de gato, tem-se os seguintes resultados:

- TP: é a imagem de um cachorro e o modelo predisse corretamente cachorro;
- TN: é a imagem de um gato e o modelo predisse corretamente gato;
- FP: é a imagem de um cachorro e o modelo predisse incorretamente gato;
- FN: é a imagem de um gato e o modelo predisse incorretamente cachorro;

A partir das informações discutidas até aqui, é possível compreender as métricas, começando pela acurácia, que pode ser calculada pela média dos valores situados na

"diagonal principal". Neste caso, ela é escrita em termos de positivos e negativos como mostra a equação **??**.

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Frequentemente é definida como o número de previsões corretas feitas pelo modelo dividido pelo número total de predições, conforme a expressão **??**. Independente da forma escolhida para representa-la, a acurácia responde a pergunta, quantas predições corretas o modelo acertou em porcentagem. Por exemplo, um modelo terá uma acurácia de 80% se em um conjunto de testes com 100 imagens ele prever corretamente 80 delas.

$$\text{Acurácia} = \frac{\text{Número de predições corretas}}{\text{Número total de predições}} \tag{5}$$

O problema dessa métrica, é que ela é eficaz somente quando as classes alvo são bem balanceadas, isto é, sempre que há aproximadamente a mesma quantidade de imagens nas diferentes categorias. Mas em situações com desbalanceadas ela é péssima para determinar o desempenho. Para entender o significado disso, pode-se considerar um conjunto de teste com 99 imagens de cães e apenas uma de gato, e um modelo de uma linha que sempre irá prevê cachorro. A acurácia neste caso é de 99 %, porque o único *output* desta rede é cachorro, não importando a imagem dada a ela, e existem 99 imagens deles. Do contrário, se fosse uma reta constante na classe gato, o cálculo resultaria em 1 % de acurácia, pois existe somente uma imagem deste animal. O motivo de utilizar outras métricas é porque na maior parte das situações trabalha-se com desigualdade de rótulos.

Entre as métricas utilizadas com classes desbalanceadas estão *recall* e precisão. A primeira é definida como a capacidade de um modelo encontrar todos os casos relevantes dentro de um conjunto de dados, e sua definição precisa é dado pela equação **??**. Essa métrica tenta responder a pergunta: Qual proporção de positivos reais foi identificada corretamente?

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

Já a precisão é a capacidade de um modelo de classificação identificar apenas os pontos de dados relevantes, sendo descrita matematicamente pela equação **??**. O objetivo dela é responder a pergunta: Qual proporção de identificações positivas estava realmente correta?

$$Recall = \frac{TP}{TP + FP} \tag{7}$$

Constantemente, há a necessidade de encontrar o equilíbrio entre *recall* e precisão. Para isso usa-se uma combinação delas que é conhecida como F1-Score. Ela é obtida através a média harmônica entre essas duas métricas, de acordo com a equação **??**. A F1-Score informa o quão preciso é o classificador, ou seja, quantas instâncias ele classifica

corretamente, bem como, o quão robusto ele é, quer dizer, se ele não perde um número significativo de instâncias.

$$F1 - Score = 2 \cdot \frac{precisão \cdot recall}{precisão + recall} \tag{8}$$

A razão para utilizar a média harmônica em vez da simples, é porque essa pune diferenças extremas. Por exemplo, se um classificador tem uma precisão igual à 1 (perfeita) e o *recall* igual à zero (pior registro possível), com a média simples o resultado é igual a 0,5, já estimado com o F1-Score a solução é 0. Então, pode-se notar que F1-Score penaliza desproporções grandes de precisão e *recall* e, portanto, é uma avaliação mais justa, principalmente para classes desbalanceadas.

A matriz de confusão e as várias métricas calculadas são maneiras de comparar o valor predito com o verdadeiro. Mas uma pergunta relevante é, qual dessas métricas é a "ideal"? A reposta depende da situação na qual o modelo está sendo executado, e do contexto do problema. Não existe um número único que possa afirmar o que é melhor. Por exemplo, não é possível declarar que 99% de acurácia é bom o suficiente para todas as situações, já que para classes desbalanceadas isso não se aplica. A mesma observação vale para precisão e *recall*, pois, ao ajustar o resultado para diminuir um, o outro será aumentado. Ou seja, é necessário decidir se o modelo deve se concentrar na correção de falsos positivos ou nos de falsos negativos. Essa constatação pode ser exemplificada ao substituir o exemplo da classificação de imagens por a de diagnóstico de doença.

No problema de predição de doenças, dificilmente haverá uma enfermidade que cerca de metade da população seja afetada e a outra não, logo, é um problema de classes desbalanceadas. Isso causa um "cobertor curto"entre precisão, que diminuindo melhora os FP, e *recall*, que diminuindo melhora os FN. É importante ressaltar que, modelos de inteligência artificial, em situações de problemas de saúde são usados como diagnóstico rápido antes de realizar um exame mais invasivo, pois, na investigação de patologias mais severas, as "apostas"são altas. Neste caso, é melhor tentar minimizar o número de falsos negativos ao custo de aumentar o de falsos positivos. para se ter certeza de que foram classificados corretamente o maior quantidade possível de casos.

Portanto, o ideal nesta cenário é que todas as pessoas doentes passem para a próxima etapa, isso é muito importante, pois os modelos de inteligência artificial não são perfeitos. Dessa forma, é melhor diagnosticar uma pessoa como enfermar e garantir que ela siga sendo acompanhada para no futuro, com outros exames, possa ser dispensada como um diagnóstico errado, do que não tratar uma pessoa enferma. Isso considerando que haverão testes complementares, que por mais invasivos possam confirmaram ou não esse resultado prévio. O oposto é verificado na fabricação de equipamento, onde prefere-se que máquinas sem defeito sejam descartadas (FP) do que as com mau funcionamento sejam entregues ao consumidor final (FN).

Existem outras métricas calculáveis que tentam prever o desempenho de modelos,

entretanto, este trabalho irá abordar somente as descritas neste capítulo. Para mais informação sobre outras possibilidade de determinar a performance podem ser estudadas em.

citar bibliografia wikipédia

2.2.6 Regressão logística para classificação binária

Muitos problemas requerem uma estimativa de probabilidade como saída. A regressão logística é um mecanismo extremamente eficiente para o cálculo de probabilidades. Em termos práticos, pode-se usar o resultado da probabilidade de uma das seguintes maneiras: como é; ou convertido em uma categoria binária.

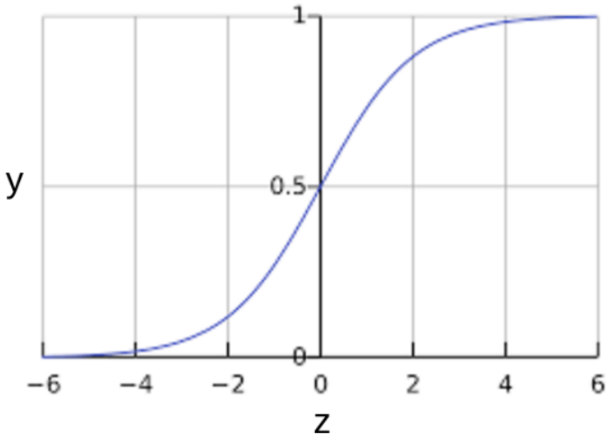
Usando sem alteração, "como está", é calculado pela probabilidade de A ocorrer dado B, definido por **??**.

$$p(A|B) \tag{9}$$

É possível converter o resultado da probabilidade para um valor binário, ou seja, para uma classificação (tema deste trabalho). Isso é realizado a partir de um limiar, também conhecido por limite de decisão, que define quando um valor de probabilidade passa de uma categoria para outra.

Para garantir que a saída do modelo de regressão logística fique entre 0 e 1, é usado uma função sigmoide, que produz o seguinte gráfico, apresentado na Figura **??**.

Figura 30 – Função sigmoide.



Fonte – (**MachineL6:online**)

Se z representa a saída da camada linear de um modelo treinado com regressão logística, então sigmoide (z) produzirá um valor (uma probabilidade) entre 0 e 1, dado pela equação **??**.

$$y = \frac{1}{1 + \exp^{-z}} \tag{10}$$

em que:

- y' é a saída do modelo de regressão logística para um exemplo específico;
- $z = b + w_1x_1 + w_2x_2 + ... + w_Nx_N$
  - w são os pesos aprendidos pelo modelo;
  - b é o bias aprendido pelo modelo;
  - x são as atributos para um exemplo específico.

Z também é conhecido como *log-odds* porque o inverso da sigmoide, dado pela equação **??**. Ele afirma que, z pode ser definido como o *log* da probabilidade de "1" rótulo dividido pela probabilidade do "0" rótulo.

$$z = \log\left(\frac{y}{1-y}\right) \tag{11}$$

Para exemplificar, considere um modelo de regressão logística que recebera como paramenteiros de entrada os seguintes valores e atributos:

- bias: $b = 1$;
- peso 1, 2 e 3: w_1, w_2 e w_3;
- atributos 1, 2 e 3: 0, 10 e 2.

z é dado pelo desenvolvimento a segui:

$$z = b + w_1x_1 + w_2x_2 + w_3x_3$$

$$z = 1 + 2 \cdot 0 + (-1) \cdot 10 + 5 \cdot 2$$

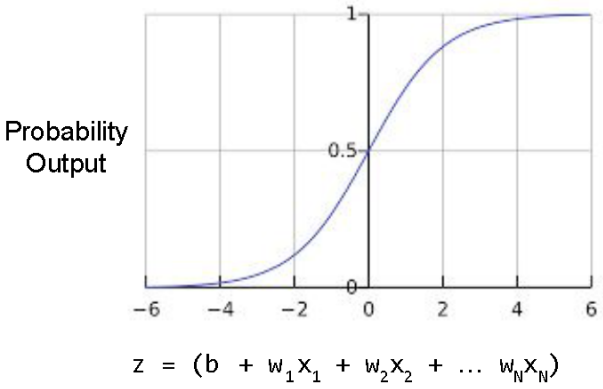
$$z = 1$$

Substituindo $z = 1$ na equação **??**, obtém-se o seguinte resultado demonstrado pela Figura **??**:

$$y = \frac{1}{1 + \exp^{-1}}$$

$$y = 0,731$$

Figura 31 – Saída da regressão logística.



Fonte – (**MachineL6:online**)

3 METODOLOGIA

4 RESULTADOS E DISCUSSÃO

5 CONCLUSÕES

6 RASCUNHO

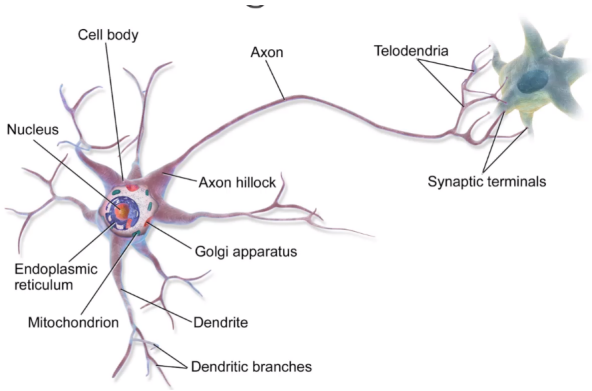
6.1 ESTRUTURA DAS REDES NEURAIS

Nesta seção será abordado o que são os neurônios biológicos e como modelá-los matematicamente em uma única percepção. Será apresentado como são agrupados os neurônios artificiais para construir um modelo de percepção de múltiplas camadas, e como expandi-los para uma estrutura de rede neural de aprendizagem profunda. Também serão introduzidos alguns conceitos matemáticos fundamentais para a funcionalidade desses neurônios.

6.1.1 Modelo de Percepção

O conceito por trás do Aprendizado Profundoé fazer os computadores simularem artificialmente a inteligência biológica e natural, logo, será discutido o funcionamento geral dos neurônios biológicos ilustrado na Figura **??**.

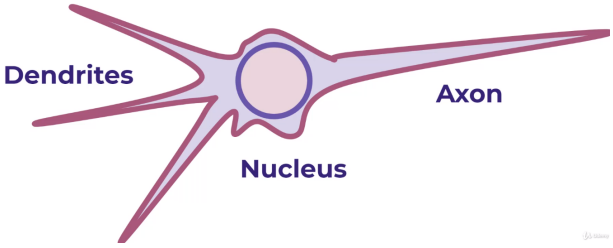
Figura 32 – Ilustração de um neurônio biológico.



Fonte –

Um neurônio saudável é composto por núcleo, corpo da célula, axônios e dendritos. Uma abstração é apresentada na Figura **??**.

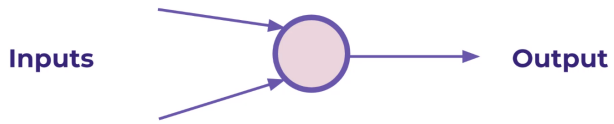
Figura 33 – Simplificação de um neurônio biológico.



Fonte –

Pode-se pensar nos dendritos como algum tipo de entrada para o núcleo principal e no axônio como saída, mas biologicamente isso não é 100 % acurado. Porém, no Aprendizado Profundoessa analogia é difundida e corresponde a alimentação do modelo de percepção. O mesmo vale para o núcleo, que é a representação de alguma função matemática a qual modifica as variáveis de entrada. Essas alteradas passam a ser a entrada do próximo neurônio ou o resultado final do sistema. Assim, é possível converter o modelo de neurônio biológico muito simplificado em um modelo de percepção, como mostra a Figura **??**. Logo, a conversão parte da substituição de cada uma das unidades biológicas por uma representação matemática.

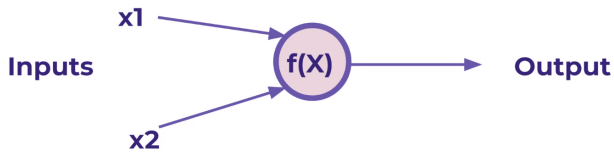
Figura 34 – Representação de um modelo de percepção simples.



Fonte –

Para exemplificar o procedimento um exemplo simples será utilizado. Nele considera-se um modelo constituído por um conjunto de variáveis x_1 e x_2, as quais são dadas de entradas a um ponto único, denominado neurônio, que gera somente uma saída y', apresentado na Figura **??**.

Figura 35 – Exemplo modelo de percepção simples com duas entradas alteradas por uma função soma que gera uma saída única.



Fonte –

Se a função da percepção for uma simples função de soma, então y' é igual ao somatório de x_1 e x_2 (sem levar em conta o conceito das funções de ativação abordado na subseção **??**).

Entretanto, isso apenas mostra como um neurônio biológico é convertido em um modelo perceptivo. Na pratica, espera-se que a própria rede consiga ajustar parâmetros para que a percepção aprenda. Como discutido na seção **??**, modelos supervisionados aprendem a partir do conhecimento prévio do y ideal, informação a qual possibilita a rede ajustar os pesos e bias com base em uma comparação realista.

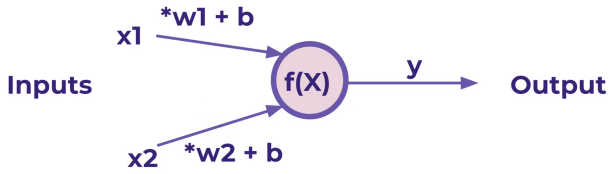
Também foi abordado na seção **??**, que cada entrada x terá um pesos específico a multiplicando. Então, o modelo apresentado na Figura **??** pode ser descrito matematicamente da seguinte forma:

$$y = x_1 w_1 + x_2 w_2$$

Um problema dessa formulação é no caso do valor de qualquer x seja zero. Dessa forma, não importa qual o ajuste feito no peso referente a esta entrada, o resultado será sempre zero, por isso adiciona-se o bias b.

$$y = (x_1 w_1 + b) + (x_2 w_2 + b)$$

Figura 36 – Esquemático de um modelo simples de percepção o qual altera as variáveis de entrada através de uma função soma. Nele cada uma das entradas são multiplicadas por seu respectivo peso mais bias gerando uma única saída y.



Fonte –

Essa é a teoria da percepção, desenvolvida como uma forma de rede neural por Frank Rosenblatt em 1958. Nela a abstração do neurônio biológico é convertido em um modelo matemático. Matematicamente é possível generalizar para n entradas como mostra a equação **??**.

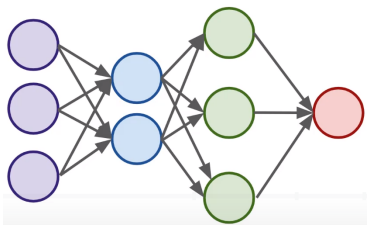
$$y' = \sum_{i=1}^{n} x_i w_i + b_i \tag{12}$$

Um único modelo de percepção não é suficiente para aprender sistemas mais complicados. Felizmente é possível expandir essa ideia para desenvolver um modelo de percepção de múltiplas camadas comumente conhecido como rede neural artificial básica.

6.1.2 Modelo de percepção de múltiplas camadas

Nesta será discutido como o modelo de percepção única é expandido para no qual x é um vetor de informação, ou seja, uma matriz dimensional. Para construir um modelo de percepção neural, basta conectar os camadas[1] de percepção usando o modelos de percepção de multi-camadas. A Figura **??** mostra um exemplo genérico de rede neural.

Figura 37 – Exemplo genérico de um modelo de multi-camadas.



Fonte –

[1] Em inglês *Layers*.

Nela há uma primeira camada vertical de neurônios e cada uma delas gera uma saída, as quais passam a ser a entrada para próxima camada de percepções. Dessa forma, a saída da camada anterior se torna a entrada da próxima camada e assim sucessivamente. Ou seja, todos os neurônios estão conectados a todos os neurônios da próxima camada, isso é conhecido com uma camada totalmente conectada, na qual todas as informações vão da camada de entrada até o final da camada de saída.

Essa alimentação direta entre as percepção é chamado de *feed forward*. Entretanto, existem diferentes tipos de camadas e configurações de rede, como redes neurais recorrentes e redes neurais convolucionais que serão abordadas nas seções **??** e **??**.

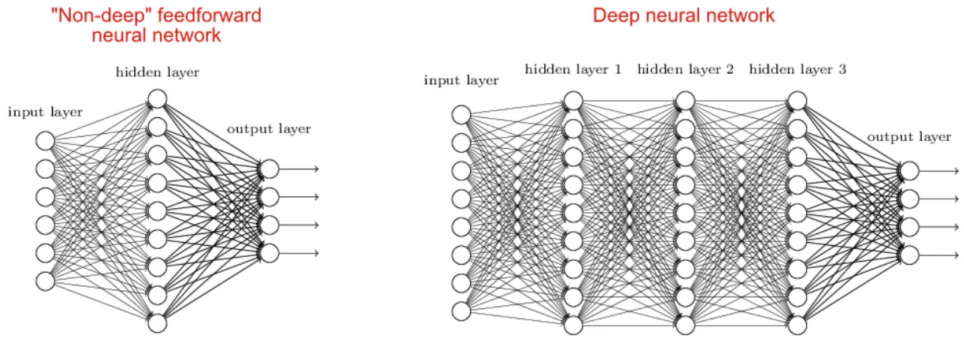
Considerando a Figura **??**, a primeira camada, também conhecida como camada de entrada[2], recebe os dados diretamente, já a última camada é reconhecida como camada de saída[3]. Deve-se ter em mente que, embora essa ilustração mostra apenas um neurônio na camada de saída na prática pode haver mais de um, especialmente quando se trata de classificação de multi-classe. Quaisquer camadas entre a de entrada e a de saída são conhecidas como camadas ocultas[4].

A camada de entrada é a mais fácil de interpretar porque aceita diretamente os dados brutos conhecidos. O mesmo vale para a camada de saída, pois está intimamente associada ao rótulo que o modelo esta tentando prever. Todavia, as camadas ocultas são consideradas caixas preta, pois a cada passo se torna mais profunda com mais camadas ocultas. Como consequência é difícil entender o que um neurônio especifico está captando até a interconectividade da última camada.

Devido quantidade de camadas que uma rede neural pode possuir, foi desenvolvido uma terminologia simples para definir se uma rede em específico é considerada uma Rede Neural. Nela é definido com Rede Neuralquando há ela conter duas ou mais camadas ocultas como mostra a Figura **??**. Outras nomenclaturas importantes são, a largura de uma rede diz quantos neurônios estão em cada camada e a profundidade determina quantas camadas existem no total.

[2] Em inglês *Input Layer*.
[3] Em inglês *Output Layer*.
[4] Em inglês *Hidden Layers*.

Figura 38 – O exemplo a esquerda é de uma rede neural não profunda, já o da direita apresenta um caso de rede neural profunda.



Fonte –

Na seção**??** foi apresentado um modelo com um único neurônio. Nele há uma função de soma simples que somara tudo o que ele receber como entrada, resultado em um grande somatório. No entanto, nos problemas os quais Aprendizado Profundoé aplicado não se quer apenas realizar uma soma direta. Em vez disso, necessita-se que o modelo consiga definir restrições para os valores de saída mais complexos, especialmente em tarefas de classificação em que todas as saídas devem estar entre 0 e 1.

Em problemas de classificação, a distribuição dos valores em uma relação binária representam as atribuições de probabilidade para cada classe. Se for realizado apenas um grande somatório, não haverá um limite superior e inferior para representar corretamente esses dados. Portanto, uma consideração importante é, qual a melhor funções para aplicar a essas entradas. Lembrando que elas também serão multiplicadas pelos pesos e que um viés será adicionado ao valor final. A fim de colocar restrições e controlar o que um único neurônio gera como saída, são utilizados as funções de ativação que serão discutidas no seção **??**.

6.1.3 Funções de Ativação

Nesta seção será explorado como usar as funções de ativação de forma a definir limites para valores de saída dos neurônios.

O valor w aplicado a uma entrada determina o quanto de peso ou força deve-se dar à entrada, pode-se pensar nisso como o quão importante é este x. Assim, se o valor absoluto desse peso for muito grande, a entrada ou recurso é muito importante.

Uma análise similar pode ser aplicada ao b que representa o deslocamento descrito anteriormente. Ele implica que o resultado da multiplicação precisa atingir certo limite antes de apresentar efeito e superar o termo b. Por exemplo, se b é igual a –10, o valor de do produto entre x e w só supera este termo b ou termo de polarização quando for maior que 10. Então, depois disso, o efeito é exclusivamente baseado no valor de w. Portanto, pode-se pensar nesse termo viés como um limite que o neurônio estabelece para que o produto entre x e w comece a exercer algum tipo de efeito majoritário.

Para definir os limites considere a equação **??** apresentada na seção **??**. Então, pode-se aplicar ao termo z alguma função de ativação para limitar o seu valor. Existem várias funções de ativação e pesquisas sobre sua eficácia, neste trabalho serão apresentadas algumas delas, para mais informações acessar o site (**Activati28:online**).