# da INTELIGÊNCIA ARTIFICIAL ao DEEP LEARNING: breve revisão histórica e conceitual

**Sergio Di Fiore1, Gustavo Kimura Montanha2, Adriane Belluci Belorio De Castro3**

¹Graduando em Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia de Botucatu sergiodifiore@gmail.com,

² Docente da Faculdade de Tecnologia de Botucatu, gustavo.montanha@fatec.sp.gov.br,

3 Docente da Faculdade de Tecnologia de Botucatu, adriane.castro@fatec.sp.gov.br.

**RESUMO**

O objetivo do trabalho é, a partir de uma revisão de literatura, expor o desenvolvimento histórico e as principais características da Inteligência Artificial e distinguir desta a tecnologia de *Deep Learning*, apresentando uma análise de diversas fontes de literatura a esse respeito e, em particular, de seus conceitos e técnicas.

**Palavras-chave:** Aprendizado de máquina. *Deep Learning*. Inteligência artificial.

**1 INTRODUÇÃO**

Por muito tempo, a humanidade sonhou com máquinas que fossem capazes de pensar de forma autônoma. Não surpreendentemente, as primeiras referências dos homens fazendo cópias deles mesmos, sem o auxílio dos deuses, vem da mitologia clássica Romana, Grega e Egípcia (NEWQUIST, 2018). Desde a Grécia antiga com Galatea, Talos e Pandora que hoje são entendidas como modelos de inteligências artificiais (BISPHAM, 2006; MARTIN, 2004; POMEROY, 2018; NEWQUIST, 2018). Quando os primeiros computadores começaram a ser concebidos, cem anos antes mesmo do primeiro ser construído já existia a pergunta se máquinas poderiam algum dia ser inteligentes (GRUNSPAN, 2018).

Em torno de 1940 alguns cientistas de computação imaginaram que a forma de se obter a resolução de problemas em nível humano seria tomando o modelo de como o cérebro humano trabalha e conectar isso de alguma forma em um padrão interconectado. Uma primeira tentativa, criada em 1951 por Marvin Minsky, então ainda um recém graduado de Princeton, a Calculadora Estocástica Neural Analógica por Reforço (SNARC - *Stochastic Neural Analog Reinforcement Calculator*), era uma rede de quarenta neurônios artificiais (GRUNSPAN, 2018).

Em 1950, Alan Turing desenvolveu o que hoje é chamado de "teste de Turing": uma máquina exibe habilidades cognitivas humanas se um humano pode interagir com ele (conversando por exemplo) sem ser capaz de dizer que está falando com uma máquina (ALEMI, 2020).

Nessa mesma década de 1950, porém, Minsky desistiu das redes neurais e começou a trabalhar principalmente na inteligência artificial simbólica que visa espelhar os níveis mais elevados humanos por meio de representação de símbolos e regras. Foi para o MIT e em 1967 juntou-se a Seymour Papert nessa escola (GRUNSPAN, 2018).

A Inteligência Artificial é a ciência de fazer com que os computadores resolvam tarefas que requerem uma mente humana para serem solucionadas. O termo foi cunhado em 1955 por John McCarthy, Marvin Minsky, Nathaniel Rochester, e Claude Shannon em sua proposta para o “Projeto de Verão DartMouth para a Pesquisa da Inteligência Artificial”, umtrabalho de dois meses e dez pessoas mantido pela faculdade Dartmouth. Por esse, Marvin Minsky foi agraciado com o prêmio A. M. Turing em 1969 e McCarty em 1971 (GRUNSPAN, 2018).

Após o sucesso do satélite artificial Sputnik em 1957, o Departamento de Degfesa norte-americano criou a agência ARPA que em 1963 cedeu US$ 2 milhões para o projeto MAC (*Machine-Aided Cognition*), o primeiro de muitos projetos na área financiado por essa agência e envolvendo a pessoa de Minsky. Um dos esforções empreendidos foi o retrabalho da linguagem LISP de McCarthy para tirar proveito dos novos avanços em hardware, surgindo uma versão chamada MacLISP, sendo escrita por Richard Greenblatt, quem vinte ano depois foi pioneiro ao levar a IA dos laboratórios ao mercado comercial. Desse trabalho junto com a ARPA também saíram os programas e modelos de Minsky usados por psicólogos para entender o raciocínio (NEWQUIST, 2018).

Em 1965 McCarthy desenvolveu o projeto de programação heurístico (*Heuristic Programming Projet* - HPP) usado por Edward Geigenbaum no Dendral de Joshua Lederberg, Bruce G. Buchanan, Edward Feigenbaum e Carl Djerassi. Foi a primeira incursão pelos sistemas especialistas, buscando uma forma de suporte às pesquisas de exobiologia. Ao final de 15 anos produziu-se a heurística Dendral e o Meta-Dendral. A Heurística Dendral agregava diferentes fontes de conhecimento e o Meta-Dendral, a parte de aprendizado, tomava esses resultados e produzia hipóteses. Por esse trabalho, Feigenbaum recebeu em 1994 o prêmio A. M. Turing (GRUNSPAN, 2018).

Em 1969, Marvin Minsky e Saymour Papert publicaram pelo MIT o livro *Perceptrons: An Introduction to Computational Geometry*, provando matematicamente que os tipo de redes de então, chamadas Perceptron, poderiam somente executar funções básicas (SOMMERS, 2017). Nesse momento estas possuíam somente duas camadas de neurônios artificiais, uma camada para a entrada de dados e uma camada para a saída, e embora, na teoria mais camadas pudessem resolver os problemas apresentados, ninguém possuía conhecimento suficiente de como treiná-las. Eles basicamente provaram que o perceptron não se beneficiava de experiências passadas e para cada experimento precisava repetir tudo de novo.

Figura 1. Capa do livro Perceptron de Minsky e Papert

Padrão do plano de fundo

Descrição gerada automaticamente

Sua técnica, porém, esbarrava na limitação da capacidade computacional dos sistemas na época e o campo toda da IA ficou por muito tempo parado, até em torno do ano 2000, quando o aumento da capacidade de processamento permitiu a retomada das pesquisas a partir desse ponto (SOMMERS, 2017).

Em torno de 2001 praticamente todas as empresas que fizeram a história até aqui resumida tinham quebrado. Os pioneiros como Marvin Minsky e John McCarthy entre outros ou já haviam falecido ou gozavam de suas aposentadorias. Sistemas especialistas e LISP deixaram de ser usados. Pequenas porções de AI ainda restavam aplicações tão improváveis como o corretor ortográfico do Microsoft Word, o portal de busca Alta Vista oferecia o *Babel Fish*, um tradutor online de páginas WEB para os usuários. Foi quando a AI viu seu recomeço. Primeiro sugerindo rotas em sistemas GPS, depois a Siri, a Cortana e a Alexa, assistentes da Apple, Microsoft e Amazon respectivamente resgatando essa ciência do desaparecimento através destes nichos de mercado. E, aos poucos, literalmente do nada, ressurgiu em todo o vigor (NEWQUIST, 2018).

Geoffrey Hinton, hoje conhecido como o pai do *Deep Learning*, trouxe novos avanços. Ele basicamente, conseguiu desenvolver o treinamento de mais camadas, uma técnica que, iria além do tradicional aprendizado de máquina, permitindo encontrar pequenos padrões, que anteriormente não conseguiriam ser detectados (HAO, 2018). O seu projeto reconhecia imagens por padrões e não por lógica. Seu objetivo foi o de conseguir com que as redes neurais reconhecessem padrões pela combinação de um processo de treinamento inicial e então tentativa e erro com o uso de processadores não tradicionais. Curiosamente, as redes neurais eram os vilões da IA, principalmente devido ao trabalhos de Minsk e Papert. A novidade de Hintom era que as redes neurais com as quais ele trabalhava, possuíam diversas camadas, as chamadas camadas ocultas, o que foi chamado de *Deep Learning*. Logo se percebeu que um dos segredos desses sistemas era que funcionavam melhor, quanto mais exemplos de aprendizagem eles tinham (NEWQUIST, 2018).

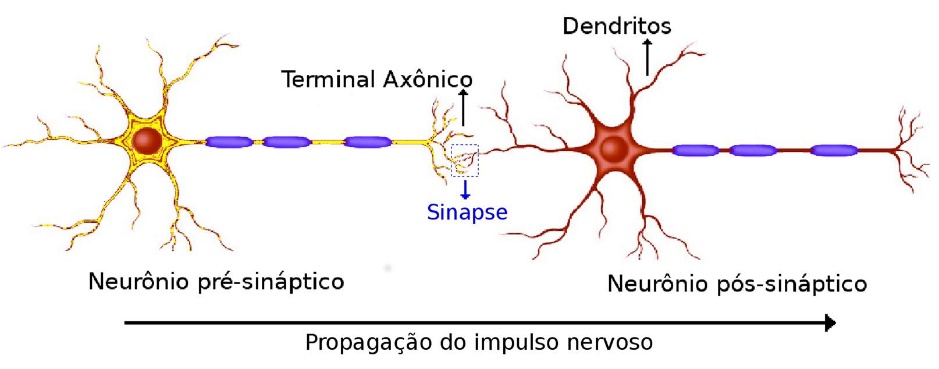
De 2012 a 2015 as redes neurais começaram a reaparecer com toda a força. Em 2011 o Watson da IBM vencia no programa *Jeopardy*, em 2015 a Google introduz o TensorFlow, em 2016 o AlphaGo® venceu o mestre sul-coreano Lee Sedol no jogo de Go. Todos esses fatores aumentaram a exposição midiática da IA, a qual hoje está se provando uma das mais importantes tecnologias da ciência da informação.

**2 DESENVOLVIMENTO DO ASSUNTO**

De uma maneira bem simples, entende-se por um sistema de IA como um sistema computacional capaz de aprender padrões os quais será capaz de reproduzir posteriormente (HAO, 2018). É importante ressaltar que, ao contrário do modelo computacional tradicional, no qual o sistema é instruído passo a passo como processar, a IA aprende por si só.

Entretanto, os sistemas de IA trabalham em modelos que simulam o funcionamento do sistema nervoso animal, particular inspiração vinda do cérebro humano. No cérebro, células nervosas (neurônios) processam um pacote de informações e transmite para uma camada sucessiva de neurônios que continuam o processamento até que, em algum ponto posterior da cadeia, o pensamento é produzido (Figura 2).

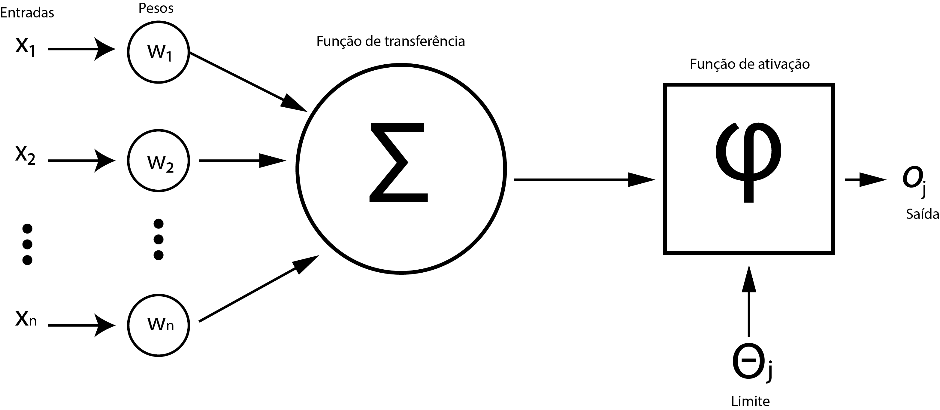
Figura 2. Dois neurônios transmitindo um impulso nervoso



Analogicamente, unidades computacionais básicas, justamente denominadas “neurônios artificiais”, ou simplesmente “nós”, serão responsáveis por um certo nível de processamento e em seguida transmite os resultado para um seguinte neurônio artificial, que por sua vez também será responsável por alguma unidade de processamento.

A Figura 3 apresenta esquematicamente um neurônio artificial, ou simplesmente um nó de dessa rede neural. Para as entradas (x1, x2, ..., xn) são atribuídos respectivos pesos (w1, w2, ..., wn) que serão processados por um determinado algoritmo (função de transferência) e o seu resultado comparado a um certo limite predeterminado. Excedido esse limite, a saída terá um determinado valor, e outro se não, tendencialmente zero.

Figura 3. Neurônio artificial



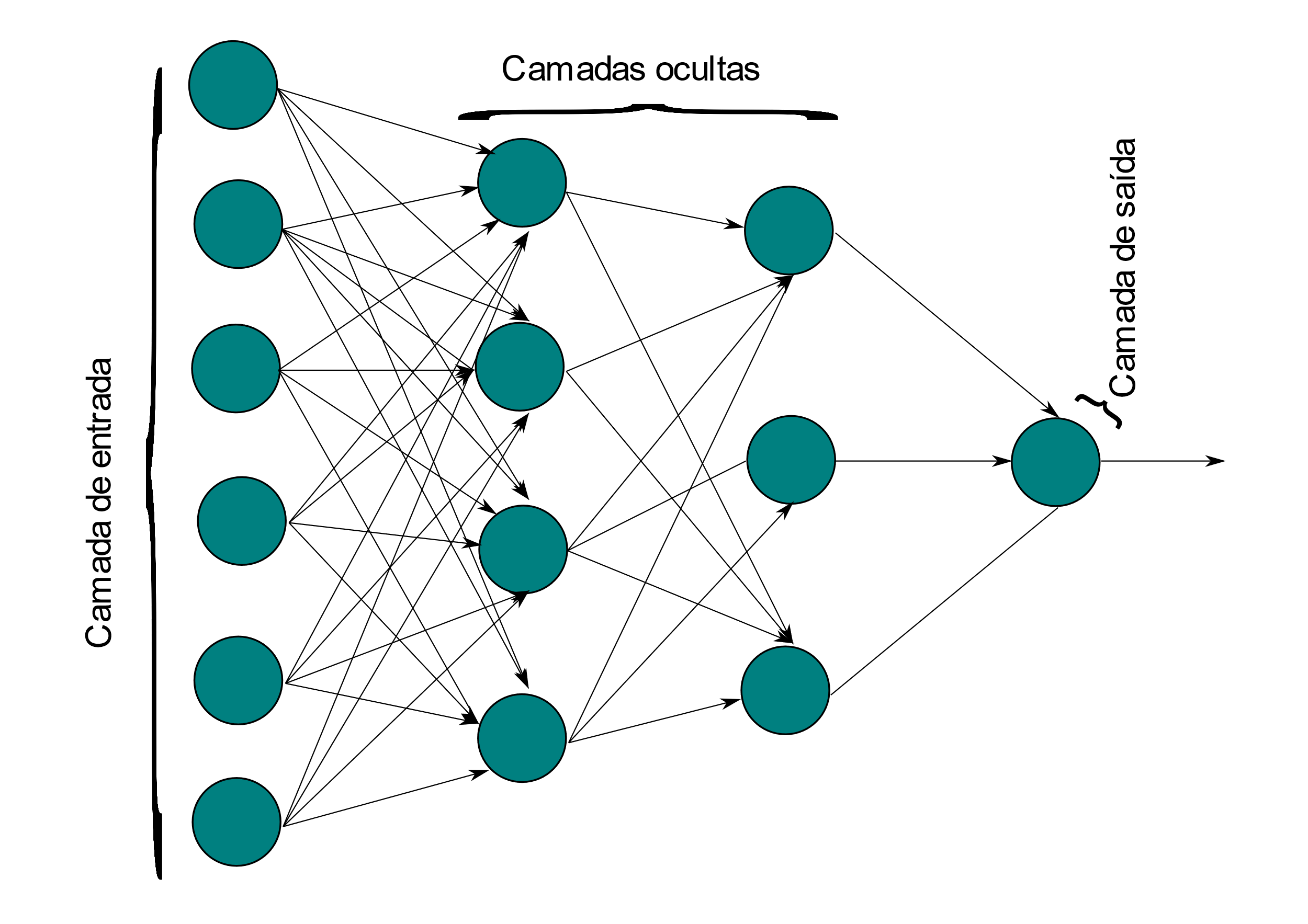
Coincidentemente, o modelo de neurônio artificial apresentado já define um padrão de rede neural anteriormente mencionado, chamada Perceptron. Proposto ainda em 1958 por Frank Rosenblatt, o Perceptron pode ser entendido como o mais simples dos modelos de redes neurais (DESMEDT, 2019), usado para aprendizado supervisiona uma rede neural de uma única camada. O Perceptron é empregado para classificar dados entre duas partes, portanto, binária e por esse motivo também é denominado classificador linear binário.

Como é binário, pode-se entender que a sua saída é representada pela Equação 1:

Equação 1. Cálculo da saída de um neurônio sigmoide

É evidente que um único nó seria um fator decisório muito básico, motivo pelo qual ele toma formas mais complexas, conforme ilustrado na Figura 4, onde camadas adicionais intermediárias são adicionadas, as ditas camadas ocultas, o que passa a constituir as chamadas rede neural profundas ou *Deep Learning*.

Figura 4. O Perceptron com multicamadas constitui uma Rede Neural Profunda ou Deep Learning



Como é possível inferir, outras diferentes composições estruturais são possíveis, dando origem a diferentes tipos de redes neurais e diferentes classificações. Todas as redes apresentadas até o momento, tem como característica que a informação viaja só em um sentido, isso é da entrada para a saída. Essas redes são denominadas redes *FeedForward* e podem ser em camadas simples ou multicamadas.

Existem casos nos quais as saídas são retroalimentadas fazendo o papel de sinais de entrada para outros nós em sistemas com variação no tempo. Alguns tipos dessas redes são: Hopfield, e Perceptron Multicamada com Retroalimentação. Uma rede neural recursiva (*Recurrent Neural Network*) é capaz de retroalimentar os próprios nós (HAO, 2108).

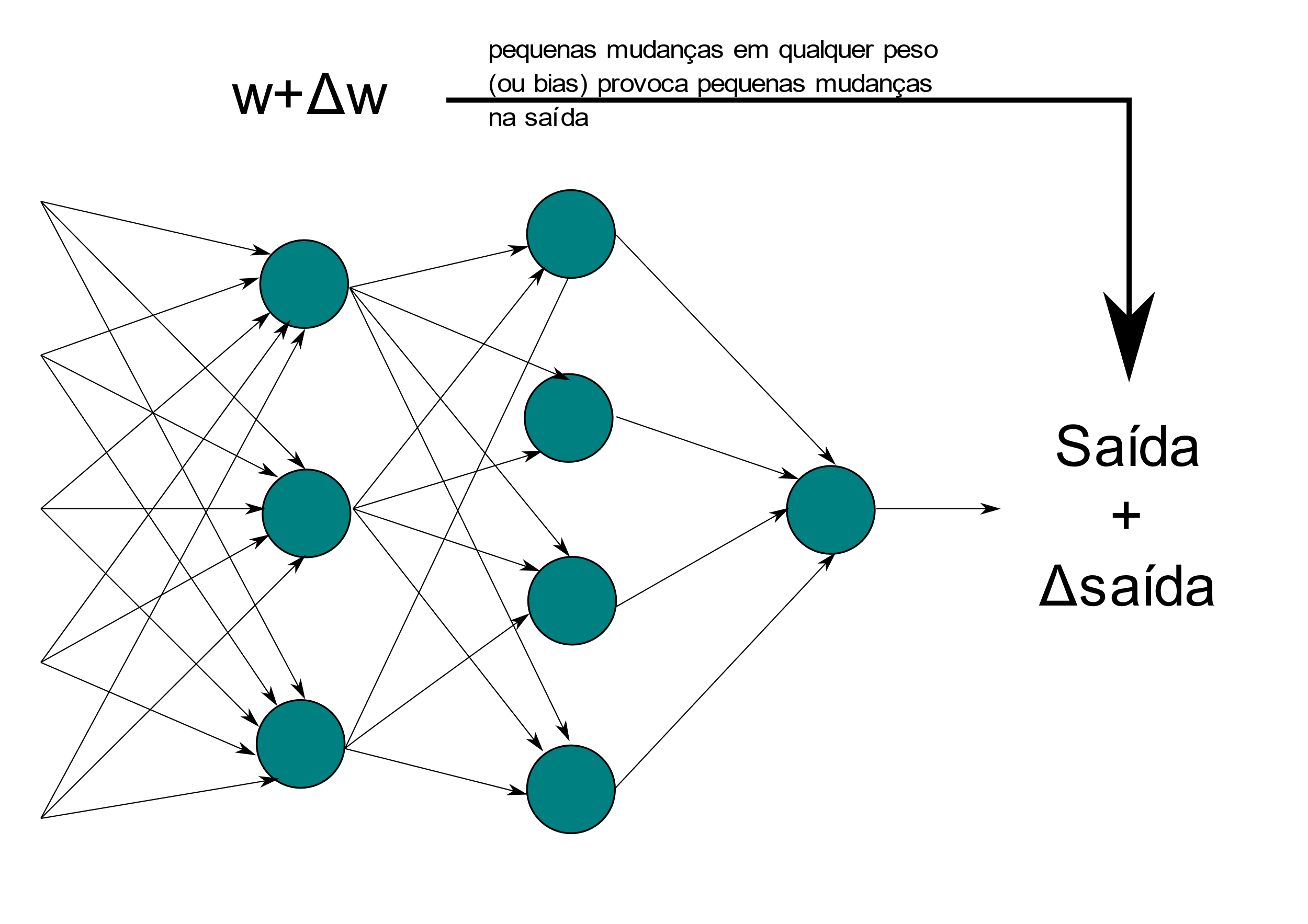
De certa forma, pode-se dizer que o campo de *Deep Learning*,nada mais é do que o aprendizado de máquinas com “anabolizantes”, ou em resumo, que tem a capacidade de encontrar, e amplificar, mesmo os menores padrões, sendo composta pela de referida camada de neurônios “ocultos” desta forma caracterizando uma rede de muitos níveis, de simples nós computacionais, e que trabalham juntos para realizar a previsão. O grande diferencial está na técnica necessária para treiná-las, que foi desenvolvida em 1986 por Geoffrey Hinton.

A figura 4 apresentada anteriormente traz um perceptron com duas camadas adicionais às de entrada e de saída, o que, como já dito, caracteriza uma *Deep learning.* A primeira camada pesa as entradas e toma alguma decisão. Neste exemplo, a saída dessas entradas é alimentada numa nova camada ( a segunda do exemplo) que com os seus próprios pesos toma alguma outra decisão e encaminha para mais uma camada (no exemplo, a terceira) que repete o processo e encaminha para a camada de saída, para só então determinar o resultado de saída. Dessa forma esse perceptron de muitas camadas pode tomar decisões sofisticadas.

Da equação 1 tem-se que a tendência de quão facilmente um perceptron pode atingir o valor 1. Para tendência com valores altos, a chance de se chegar ao valor 1 é alta, já se o tendência for um grande valor negativo, isto será difícil.

Para entender como o processo de aprendizagem pode funcionar, imagina-se (Figura 5) conduzir pequenas mudanças em alguns dos pesos (ou dos tendência) da rede para que às pequenas mudanças referidas somente pequenas mudanças ocorram de forma correspondente à saída. Esta é justamente essa propriedade que torna possível o processo de aprendizagem.

Figura 5. O efeito de pequenas alterações ou tendências



Se fosse verdadeiro que pequenas modificações em pesos (ou tendência) provocam somente pequenas modificações no valor de saída, poder-se-ia usar essa fato para a rede se comportar mais como se deseja. Por exemplo, se é esperado um resultado com o valor 8 e se obtém o 9, pequenas alterações nos pesos e tendência levariam mais próximos ao resultado desejado, e repetindo esse processo a precisão melhoraria.

Mas, não é o que acontece com os perceptrons. É possível que uma pequena alteração em um peso ou tendência cause a mudança completa de estado de um perceptron, por exemplo, de 0 para 1.

Esse problema pode ser evitado com o uso do neurônio sigmoide. Semelhante ao perceptron, mas permite justamente que pequenas mudanças em pesos e tendência reflitam em pequenas mudanças no valor de saída. O cálculo de saída do neurônio sigmoide é dado por:

Equação 2. Cálculo da saída de um neurônio sigmoide

De forma mais explicita, agora com pesos e tendência tem-se:

Equação 3. Cálculo da saída sigmoide considerando pesos e tendência

Figura 6. Gráfico da função sigmoide

Gráfico

Descrição gerada automaticamente

Para entender a similaridade entre perceptron e sigmoide, suponha-se sendo um grande número positivo. Então e por isso, . Ou seja, quando é um número grande positivo, a saída da função sigmoide é aproximadamente 1, como ocorreria com um perceptron. Quando é um valor de grande magnitude negativa, então , causando, . Isto é, para valores grandes negativos, o comportamento da função sigmoide também parece a do perceptron. Somente quando possuir valores moderadamente intermediários é que vai haver um desvio considerável do perceptron.

Figura 7. A função degrau ou de Heaviside

Gráfico

Descrição gerada automaticamente

Se fosse de fato uma função degrau (figura 7), o neurônio sigmoide seria um perceptron. Pelo uso da função obtém-se um perceptron “suavizado”. E justamente essa suavização que é o fator crucial. Ela significa que pequenas alterações dos pesos ∆wj e das tendência ∆b produziram uma pequena alteração na saída ∆saída. Pode-se saber por meio da disciplina do cálculo que ∆saída será aproximadamente:

Equação 4. Cálculo aproximado de ∆saída

onde a soma é entorno de todos os pesos, wj, e denotam as derivadas parciais a saída com relação a wj e b respectivamente. Portanto, ∆saída é uma função linear das mudanças que ocorrem com ∆wj e ∆b.

**3 APRENDIZADO DE MÁQUINA**

O aprendizado de máquina apresenta quatro mais importantes possibilidades: supervisionadas, semi-supervisionado e não supervisionadas e por reforço (HAO, 2018).

Para compreendê-los melhor, assuma-se um conjunto de dados como sendo um conjunto de exemplos rotulados . Cada elemento de é chamado um vetor de características, que nada mais é do que um vetor no qual cada dimensão contém um valor que de alguma forma descreve o evento. Esse valor é denominado característica e é denotado como . Assim, cada valor na colação pode, por exemplo, representar uma pessoa, então, a primeira característica pode conter a altura em centímetros, a segunda característica, o peso em kg, e o sexo, e assim em diante. Para cada exemplo no conjunto, a posição da característica na posição contém algum tipo de informação. Assim, se contém o peso em kg em algum exemplo xi, então também conterá pelo em kg para cada exemplo de . O rótulo pode tanto ser um elemento que pertence a um conjunto de classes finitas , ou um número real, ou alguma estrutura mais complexa como um vetor, uma matriz, uma árvore, um gráfico. Pode-se entender uma classe como a categoria a qual o elemento pertence. Assim, se os exemplos são mensagens de correio eletrônico e o problema é a detecção de spam, tem-se duas classes: (BURKIV, 2019).

No aprendizado supervisionado, que é o mais comum, os dados são rotulados de forma a dizer à máquina exatamente por quais padrões ela está procurando. O objetivo é tomar um vetor de entrada x e obter-se como informação de saída valores que permitam com que se deduza o rótulo desse vetor de características. Assim, por exemplo, um modelo criado usando um conjunto de pessoas pode tomar como entrada um vetor das características que descrevam a pessoa e como saída a probabilidade que essa pessoa tenha câncer (BURKIV, 2019).

No aprendizado semi-supervisionado, o conjunto possui tanto exemplos rotulados quanto não rotulados. É comum que a quantidade de exemplos não rotulados seja muito maior que o de rotulados. O objetivo aqui é o mesmo do supervisionado, mas, espera-se que usando essa quantia de exemplos não rotulados possa ajudar o algoritmo de aprendizagem a encontrar um modelo mais adequado (BURKIV, 2019).

No aprendizado não supervisionado, os dados não são rotulados. A máquina encontra algum padrão por si só. Assim, o conjunto de exemplos é um conjunto não rotulados . Novamente, x é um vetor de características e o objetivo é o de se criar um modelo que tome este vetor x de entrada e o transforme em um outro vetor, ou então, em um valor que possa ser usado para a resolução do problema. Por exemplo, em agrupamentos, o modelo retorna à identificação de um grupo para cada vetor no conjunto. Em redução dimensional, a saída do modelo é um vetor de características que possui uma quantidade menor de características que a entrada x (BURKIV, 2019).

O aprendizado por reforço caracteriza-se pelo fato que o algoritmo aprende por tentativa e erro a atingir um determinado objetivo. Ele faz diversas tentativas e é premiado ou penalizado dependendo de como se aproxime desse resultado objetivo. Pode-se entender como a parte do aprendizado de máquina onde a máquina “vive” no ambiente e é capaz de perceber o estado desse ambiente como um vetor de características. A máquina pode executar ações em cada estado. Ações diferentes trazem “recompensas” diferentes e podem levar a máquina par um outro estado do ambiente. O objetivo aqui é aprender a “política”, que é uma função (similar à do modelo de aprendizado supervisionado) que toma o vetor das características de um estado como sendo a entrada e as saídas uma ação otimizada para ser executada naquele estado. Essa ação é otimizada se maximiza a recompensa média esperada (BURKIV, 2019).

O aprendizado por reforço resolve um tipo particular de problema onde a tomada de decisão é sequencial e o objetivo é de longo prazo, tal como jogos, robótica, gerenciamento de recursos ou logística (BURKIV, 2019).

**4 REDES NEURAIS**

Uma rede neural (NN) é expressa pela função matemática:

Equação 5. Função de uma rede neural

A função tem uma forma particular: é uma função aninhada. Assim, como exemplo, em uma rede neural de três camadas que retorna um escalar têm-se:

Equação 6. Função de uma rede neural de três camadas

Na equação acima, e são funções vetoriais na forma:

Equação 7. Função dos vetores f1 e f2

Onde é chamado de índice da camada, e que endereça a camada que pode ir de 1 até o número total das camadas. A função é chamada função de ativação, sendo normalmente uma função linear fixa escolhida pelo analista de dados antes do processo de aprendizagem. Os parâmetros (uma matriz) e (um vetor) —para cada camada são aprendidos por meio do gradiente de descida pela otimização, em função da tarefa, de uma função custo em particular (tal como a MSE) de entrada e que devolve um número como saída. Essa FFNN pode ser do tipo de regressão ou modelo de classificação, dependendo da função de ativação empregada (BURKIV, 2019).

**5 ANATOMIA DE UM ALGORITMO DE APRENDIZADO**

**5.1 COMPOSIÇÃO**

Pode-se entender um algoritmo de aprendizado como sendo composto por três partes:

1) Função custo;

2) Critério de otimização baseado na função de perda (função de custo, por exemplo); e

3) Rotina de otimização aproveitando dados de treinamento para encontrar uma solução para a otimização critério.

Estes são os bloco de construção de qualquer algoritmo de aprendizagem. Ao ler literatura moderna sobre aprendizado de máquina, muitas vezes encontra-se referências à descida gradiente ou descida de gradiente estocástico. Estes são dois algoritmos de otimização mais usados em casos em que o critério de otimização é diferente (BURKIV, 2019).

O desejável é que o algoritmo que se está buscando permita definir todos os pesos e tendência de tal forma que o resultado de saída da rede aproxime y(x) para todos os x na entrada. Isto é quantificado pela função custo:

Equação 8. Função Custo

onde n é o conjunto de todos os pesos na rede, b todos as tendência, n o número total de entradas de treinamento, a é o vetor de saída da rede quando x é a entrada, e a soma é sobre todos os x de entrada. A notação ||v|| simplesmente indica o comprimento de um vetor. *C* é chamada de função quadrática do custo, muitas vezes conhecida também pela denominação erro médio quadrático ou simplesmente MSE (*mean squared error*). Verifica-se que esta função não é negativa uma vez que cada termo da soma também não o é. Nota-se que torna-se menor, isto é, , quando y(x) é de forma aproximada igual à saída, a, para todas as entradas de treino x. Depreende-se que o algoritmo é eficiente quando encontra pesos e tendência de tal forma que . Já, se esse valor for alto, entende-se que y(x) não se aproxima da saída para muitos dos valores da entrada. O objetivo é então minimizar a função custo, sendo está uma função de pesos e tendência. Para fazê-lo, será utilizado o algoritmo conhecido como método do gradiente (GOODFELLOW, 2016).

**5.2 O ALGORITMO DO MÉTODO DO GRADIENTE**

O objetivo imediato ao treinar uma rede neural é encontrar valores para pesos e tendênciaque minimizem a função quadrática do custo . Assim, deseja-se minimizar uma função , que pode ser qualquer função de valor real com múltiplas variáveis v = v1, v2, ..., vn. Para fazê-lo é útil o exercício mental de se imaginar como uma função de somente duas variáveis, v1 e v2 que pode ser facilmente representada graficamente como na figura 8:

Figura 8. Função de duas variáveis vi e v2

Gráfico, Gráfico de superfície

Descrição gerada automaticamente

Inicia-se encontrando uma forma de escolher 1 e 2 de forma a fazer negativo, ou seja, em direção e sentido como se “descesse” o gráfico da função. Para facilitar define-se como um vetor de mudança em v, 1, 2)*T*, onde *T* novamente é a operação de transposição. Define-se também o gradiente de C como sendo o vetor das derivadas parciais . O vetor gradiente é representado por , desta forma tendo a equação 6:

Equação 9. Vetor gradiente

Com essas definições pode também ser entendido:

Equação 10. também pode ser entendido como:

Suponha-se que é escolhido:

Equação 11. Escolha de taxa de aprendizado

onde é um parâmetro de pequena magnitude, positivo que é conhecido como taxa de aprendizado. Considerando que . Como , é garantido que ou seja, C sempre será decrescente. Caso efetue-se a mudança de v de acordo com a equação anterior, entre os limites da aproximação da equação 10, pode-se usar a equação 11 para calcular o valor de quando se move para a posição v:

Equação 12. Cálculo de com movimentação para a posição v

E essa mesma regra será empregada então sucessivamente para os próximos deslocamentos. Repetindo-se esse deslocamento é esperável que se alcance o mínimo global, o que pressupõe figuradamente imaginar uma suposta esfera caminhando pela superfície procurando o ponto mais baixo, pressuposto que é um paralelo com certa razoabilidade com como o caminho supra descrito pode ser imaginado.

Até agora, analisou-se uma função de duas variáveis, mas o mesmo raciocínio se aplica a um maior número delas. Suponha-se: v1, ..., vm. A mudança de no valor de C produzido por uma pequena variação é onde o gradiente é o vetor de acordo com a equação 13.

Equação 13. Vetor gradiente para múltiplas variáveis

Aplicar o método do gradiente para determinar os pesos wk e as tendência bl de tal forma que se minimize a equação do custo (equação 5)? Reescreva-se a regra de atualização do método do gradiente com os pesos e tendência substituindo as variáveis v­j, ou seja, a “posição” possui componentes wk e bl e o vetor gradiente tem os componentes correspondentes e . Tem-se assim as equações 11 e 12:

Equação 14: Componente do peso

Equação 15: Componente do tendência

Equação 16: Componente do tendência

Aplicando-se repetidamente essa regra, consegue-se “rolar para baixo” e espera-se, atingir o mínimo. Esta deve ser, portanto, a estratégia a ser usada para a aprendizagem da rede neural neste trabalho.

Deve-se perceber um primeiro desafio, que para entender melhor voltaremos à equação quadrática do custo, a equação 8. Perceba-se que esta tem a forma geral , ou seja, é o custo médio sobre os custos para as práticas de treinamento individuais. Isso implica, na prática que para se calcular os gradientes separados para cada entrada x, e então calcular-se a média . Acontece que, se o número de entradas de treinamento for muito grande, e quanto maior, melhor, essa abordagem implicará em um alto tempo de processamento e, portanto, longo tempo de aprendizagem.

O gradiente de descida estocástico pode ser empregado para acelerar o aprendizado. Basicamente se estima o gradiente através do cálculo de x para uma pequena amostra de treinamento escolhida de forma aleatória. Quando se faz a média sobre essas pequenas amostras, acaba sendo uma maneira rápida de se estimar o real a e, portanto, de se acelerar o aprendizado.

Para precisar melhor, o gradiente de descida estocástico funciona através da seleção de uma pequena quantidade de entradas de treinamento m escolhidas de forma aleatória. Rotulando-as x1, x2, ..., xm serão referenciadas como um minilote. Admitindo-se que o tamanho da amostra m seja grande o suficiente, espera-se que o valor de seja aproximadamente igual a média sobre todos os x, isto é:

Equação 17: O gradiente de descida estocástico sobre uma pequena amostra de dados de treino é aproximadamente o de toda a amostra

Assume-se que wk e bl denotem os pesos e tendência na rede neural em questão. O gradiente de decida estocástico, como consequência do que foi visto acima, é operado pela escolha aleatória de um minilote de dados de entrada de treinamento e fazendo-se o treinamento com estes:

Equação 18: Componente peso com o gradiente de descida estocástico

Equação 19: Componente tendência com o gradiente de descida estocástico

(GOODFELLOW, 2016).

**6 CONSIDERAÇÕES FINAIS**

O *Deep Learning*, as redes com mais de duas camadas, ou com camadas ocultas, chega como uma teoria, e prática, que permite enfrentar desafios e modelos muito mais complexos, acompanhando a evolução de equipamentos de processamento mais potentes. A tecnologia, literalmente, ficou estagnada pelas décadas de 1980 e 1990, aguardando a teoria que viabilizasse o próprio *Deep Learning* e o desenvolvimento de tecnologias de hardware com capacidade de processamento suficiente para enfrentá-las. Hoje, o *Deep Learning* é uma tecnologia consistente e capaz de resolver eficientemente uma série de novos desafios.

# 6 REFERÊNCIA

Alemi, M. (2020). *The Amazing Journey of Reason from DNA to Artificial Intelligence.* Springer Open.

Burkiv, A. (2019). *The Hundred-Page Machine Learning Book.* Burkiv, Andiy.

Edward Bispham, T. H. (2006). *The Edinburgh Companion to Ancient Greece and Rome.* Edinburgh University Press.

Grunspan, S. L. (2018). *The Computer Book: From the Abacus to Artificial Intelligence.* Sterling.

Hao, K. (17 de 11 de 2018). *What is machine learning?* Fonte: MIT Technology Review: https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/

Ian Goodfellow, Y. B. (2016). *Deep Learning*. Fonte: Deep Learning: http://www.deeplearningbook.org

Jerome Nilmeier, P. (2019). *Data Science and Engineering at Enterprise Scale.* IBM / O’Reilly Media.

Krohn, J. (2020). *Deep Learning Illustrated.* Pearson Education.

Martin, C. (2004). *Ovid Metamorfoses Kindle Edition.* W. W. Norton & Company.

Newquist, H. (2018). *The Brain Makers: The History of Artificial Intelligence – Genius, Ego, And Greed In The Quest For Machines That Think - Kindle Edition.* The Relayer Group.

Nielsen, M. (2019). *Chapter 1*. Fonte: Neural Networks and Deep Learning: http://neuralnetworksanddeeplearning.com/chap1.html

Sarah B. Pomeroy, S. M. (2018). *Ancient Greece - A Political, Social, And Cultural History.* Oxford University Press.

Sejnowski, T. J. (2018). *The Deep Learning Revolution.* MIT Press.

Somers, J. (29 de 9 de 2017). *Is AI Riding a One-Trick Pony?* Fonte: MIT Tecnology Review: https://www.technologyreview.com/2017/09/29/67852/is-ai-riding-a-one-trick-pony/

Stuart Russell, P. N. (2016). *Artificial Inteligence - A Modern Approach.* Pearson Education.

Taulli, T. (2019). *Artificial Intelligence Basics: A Non-Technical Introduction.* Apress.