



Inteligência Artificial Fundamentos e Aplicabilidades

Álvaro Farias Pinheiro

Secretaria de
Administração



EXPEDIENTE

Governador de Pernambuco
Paulo Henrique Saraiva Câmara

Vice-governadora de Pernambuco
Luciana Barbosa de Oliveira Santos

•

SECRETARIA DE ADMINISTRAÇÃO

Secretária
Marília Raquel Simões Lins

Secretário Executivo
Adailton Feitosa Filho

Diretora do CEFOSPE
Analúcia Mota Vianna Cabral

Coordenação de Educação Corporativa
Priscila Viana Canto Matos

Chefe da Unidade de Coordenação Pedagógica
Marilene Cordeiro Barbosa Borges

Autor
Álvaro Farias Pinheiro

Revisão de Língua Portuguesa
Eveline Mendes Costa Lopes

Diagramação
Sandra Cristina da Silva

•

Material produzido pelo Centro de Formação dos Servidores e Empregados Públicos do Poder Executivo Estadual – CEFOSPE

Julho, 2020 (1ª. ed.)

Ficha catalográfica elaborada pela bibliotecária Ana Luiza de Souza/ CRB 2066

P654i Pinheiro, Álvaro Farias
 Inteligência Artificial: fundamentos e aplicabilidades/ Álvaro Farias Pinheiro; Governo do Estado de Pernambuco, Secretaria de Administração, Centro de Formação dos Servidores e Empregados Públicos do Poder Executivo Estadual. – Recife: Cefospe, 2020. 62p.: il.

1. Inteligência Artificial (AI). 2. Automação. 3. Redes Neurais Artificiais (RNA). I. Governo do Estado de Pernambuco. II. Secretaria de Administração. III. CEFOSPE. IV. Título.

CDD 004.8
CDU 004.89

Sumário

Apresentação	6
Introdução.....	9
1. Abordagem Neural	12
2. Evolução	19
3. Características	21
4. Entendimento.....	24
5. Perceptron	36
6. Adaline	44
7. MultiLayer Perceptron (MLP)	47
8. Convolutional Neural Network (CNN).....	49
Conclusões	58
Referências	59
Material Complementar.....	60
Sobre o autor.....	61

Lista de Figuras

Figura 1.1. Alan Turing	6
Figura 1.2. Neurônios Biológico e Artificial	7
Figura 1.3. Áreas da Inteligência Artificial	8
Figura 2.1. Rede Neural Artificial.....	9
Figura 2.2. Representação de Entrada/Saída	10
Figura 2.3. Representação Gráfica de $z=f(x,y,z)$	11
Figura 3.1. Função Linear	13
Figura 3.2. Função Não Linear	13
Figura 3.3. Função Sigmoide ou Logística	15
Figura 3.4. Função Tangente Hiperbólica (TanH)	15
Figura 3.5. Função Ativação Linear Retificada (ReLU)	16
Figura 3.6. Função Unidade Linear Exponencial (ELU)	16
Figura 3.7. Passos para a Aplicação de uma RNA.....	17
Figura 4.1. Neurônio McCulloch-Pitts	19
Figura 4.2. Perceptron	19
Figura 4.3. ADALINE	20
Figura 5.1. Método Indutivo	21
Figura 6.1. Imagem de Cachorro e Lobo.....	24
Figura 6.2. Imagens de Flores de Íris.....	27
Figura 6.3. Gráfico de Dispersão	32
Figura 6.4. Reta de Separação.....	32
Figura 6.5. Reta de Separação com Melhor Ângulo	33
Figura 6.6. Parâmetro de Erro (E)	34
Figura 6.7. Dois Classificadores	35
Figura 7.1. Perceptron.....	36
Figura 7.2. Perceptron	37
Figura 7.3. Pré-processamento.....	38
Figura 7.4. Perceptron de Frank Rosenblatt.....	39
Figura 7.5. Passos para Treinamento de uma Rede Perceptron.....	40
Figura 7.6. Arquitetura Perceptron para Problema de Ocupação	41
Figura 9.1. MultiLayer Perceptron	47
Figura 10.1. Convolutional Neural Network	49
Figura 11.1. Revolução Industrial	58

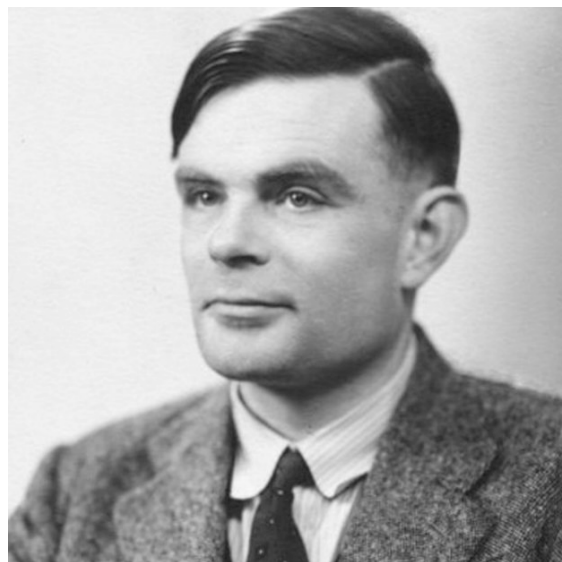
Apresentação

A finalidade deste material não é ser um livro sobre Inteligência Artificial (IA), mas sim, um artefato, que possa ser um guia introdutório e orientador para fornecer conceitos básicos necessários ao entendimento do que é IA e qual sua proposta de uso. IA é um campo vasto, que vem sendo explorado e evoluído desde a década de 1950. Assim para ser um especialista nesse campo, deve-se buscar verticalizar num dos subcampos, mas ter-se uma noção do todo é importante para poder saber o que é e para que serve.

Assim este material está dividido em capítulos que trazem informações fundamentais sobre diversos aspectos desde o seu surgimento e o processo de evolução. É bom lembrar que a inovação nessa área é constante, o que torna este material rapidamente desatualizado, se a proposta fosse falar da inovação. Dessa forma, fará referência a livros, artigos, sites e vídeos como material complementar que poderão subsidiar melhor os interessados em relação a aspectos mais atualizados sobre o tema.

Pode-se dizer que a Inteligência Artificial começou com Alan Turing que é considerado o pai da IA por ter publicado na década de 1950 o artigo “Computing Machinery and Intelligence”.

Figura 1.1. Alan Turing



Há muitos anos, a humanidade estuda como ocorre o funcionamento da inteligência humana e, nas últimas décadas, vem tentando replicá-la nas máquinas. Hoje estamos em um estágio da evolução que utilizar máquinas eletrônicas, que resolvem problemas científicos ou da indústria, não nos é mais suficiente.

A busca atual está na automatização de tarefas, *Robotic Process Automation* (RPA), isto é, automação robótica de processo, e na aplicação de *Artificial Intelligence* (AI), nos problemas

complexos, como identificação de imagens por reconhecimento de padrões, agrupamento de imagens por semelhança, entre tantas outras.

Porém, para a construção de software com Inteligência Artificial (IA), precisa-se de uma capacidade mais profunda das máquinas que as encontradas em aplicações comerciais mais complexas, como um *Enterprise Resource Planning* (ERP), que apoia o controle e a gestão de todas as áreas de uma organização nos níveis operacional, tático e estratégico.

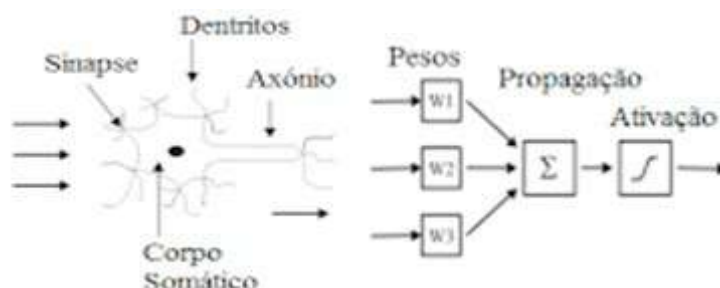
Só que a indústria do entretenimento fantasiou, de certa forma, esse tipo de máquinas, tornando-as sedutoras e, ao mesmo tempo, temerosas, como o computador HAL 9000 de “Uma Odisseia no Espaço” ou a Skynet do “Exterminador do Futuro”, mas foi quando Kasparov, até então campeão mundial de xadrez, foi vencido em 1997 pelo supercomputador IBM Deep Blue que a capacidade e promessa da IA teve seu verdadeiro potencial entendido.

Na década de 1950, muitos estudos e teorias começaram a ser propostos, buscando criar algoritmos estruturados com lógicas análogas à lógica humana, saindo de uma lógica algorítmica, binária, para uma lógica proposicional, evoluindo para uma lógica fuzzy, mais semelhante à da cognição humana.

Na década de 1970, vislumbrou-se o quanto é desafiador esse campo de pesquisa e quão lento era o seu progresso, resultando na diminuição de investimento e perda de interesse pela comunidade, o que só foi retomado na década de 1980, quando foram lançadas novas ideias.

Um fator que reacendeu o interesse pela pesquisa nessa área foi o estudo do cérebro das abelhas, que, com uma fração de grama, era capaz de realizar tarefas complexas, como orientação em voo. Assim as pesquisas em redes neurais se intensificaram e passaram a compor um método na inteligência artificial, denominada de computação biológica inteligente.

Figura 1.2. Neurônios Biológico e Artificial



Nos dias atuais, as redes neurais se sofisticaram e estão na base de muitos métodos computacionais, fazendo coisas impensáveis há pouco tempo, como aprender a executar tarefas a sós, no reconhecimento de padrões e tantas outras atividades.

Figura 1.3. Áreas da Inteligência Artificial

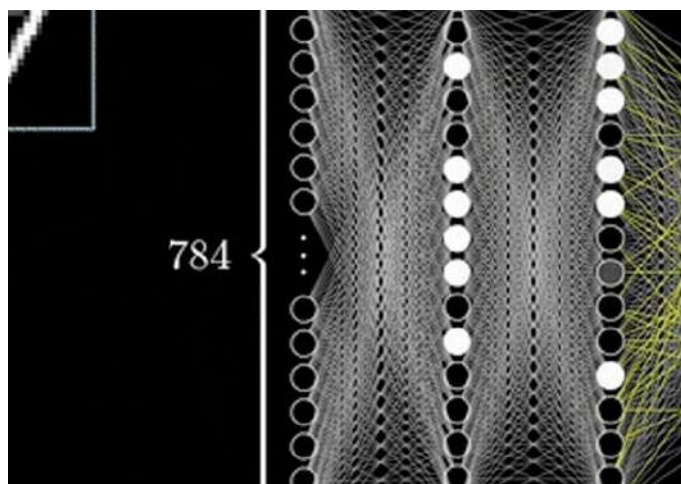
Homem	Máquina	Campos
Falar/Ouvir	Speech Recognition	Statistical Learning
Escrever/Ler	NLP	
Ver	Computer Vision	Symbolic Learning
Imagens	Image Processing	
Movimentar	Robotics	
Padrões	Pattern Recognition	Machine Learning
Padrões Complexos	Object Recognition	Deep Learning

Então, apresentado o objetivo deste material, vamos à introdução ao conceito de redes neurais artificiais.

Introdução

A finalidade deste material é introduzir o conceito de Redes Neurais Artificiais (RNA) e entender o porquê, dentro dos fundamentos de inteligência artificial, de se fazer necessário conhecê-las, mesmo que basicamente. Então vamos iniciar definindo, de forma básica, o que é uma RNA. Podemos então dizer que é uma técnica de Inteligência Artificial (IA), mais corretamente especificando Inteligência Computacional (IC), pois é uma representação subsimbólica, que segue a abordagem conexionista, fundamentada em modelos matemáticos baseados nas estruturas neurais biológicas, que fazem uso de dados para aprender.

Figura 2.1. Rede Neural Artificial



Quando, no parágrafo anterior, nos referimos à “fundamentada em modelos matemáticos”, o que queremos dizer é que RNA fazem uso de uma lista de subsímbolos ou objetos, analogamente à teoria de conjuntos da matemática, em que existem casos de relação entre elementos dos conjuntos.

Essas relações podem ser conforme os diagramas de Venn, em que as ligações dos elementos de dois conjuntos só ocorrem em um sentido, com a denominação de conjunto de partida ou domínio de onde saem os dados e conjunto de chegada ou contradomínio de onde chegam os dados. Mas as relações também podem ser de forma que o conjunto de partida, domínio, deva se relacionar com um elemento do conjunto de chegada, contradomínio. E ainda existem as relações que cada elemento do domínio deva estar relacionado a um e somente um elemento do contradomínio. Assim, caso as três condições existam, essa relação passa a ser considerada uma função.

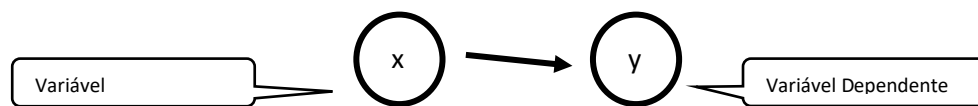
Então, podemos corroborar que, quando algo está em função de alguma outra coisa, estamos dizendo que algo depende de outra coisa, para poder ser definida ou compreendida ou mesmo completada.

Expressando matematicamente, temos $y=f(x)$, onde podemos entender que cada elemento x pertence a um conjunto e esse se associa a um e somente um elemento y que

pertence a outro conjunto. Assim, essa relação $y=f(x)$ indica que cada valor de y é originado a partir de um valor de x que foi previamente escolhido, então y depende de x , visto que é necessário conhecermos o valor de x previamente para que se possa obter o valor de y . Portanto, dizemos que x é a variável independente e y é a variável dependente.

Logo devemos gravar esse conceito básico, pois, em RNA, se usa a terminologia de variáveis independentes, normalmente representadas por x , para expressar as entradas de dados, e o termo variáveis dependentes, representadas comumente por y , para denotar as saídas de dados, conforme a figura abaixo:

Figura 2.2. Representação de Entrada/Saída



Assim, podemos perceber que uma função pode ser explicada como uma quantidade, variáveis dependentes, é determinada por outra quantidade, variáveis independentes, sempre de uma forma única.

Mas é importante ressaltar que há casos em que funções necessitam definir várias variáveis independentes, associadas a uma variável dependente. Isso significa que o cálculo do valor da variável dependente está atrelado ao cálculo prévio dos valores de cada uma das variáveis independentes.

Matematicamente, esse tipo de função com duas ou mais variáveis independentes é expresso na forma $z=f(x, y)$.

Nessas situações em que existem duas ou mais variáveis independentes, o domínio é um conjunto de pares ordenados (x,y) , que se associam às combinações dos pontos do contradomínio. Em outras palavras, para a expressão $z=f(x,y)$, a variável dependente z depende de x e de y , as variáveis independentes.

Se a expressão for $z=f(x,y,z)$, teremos 3 variáveis independentes x,y,z relacionadas a uma variável dependente z . E assim por diante.

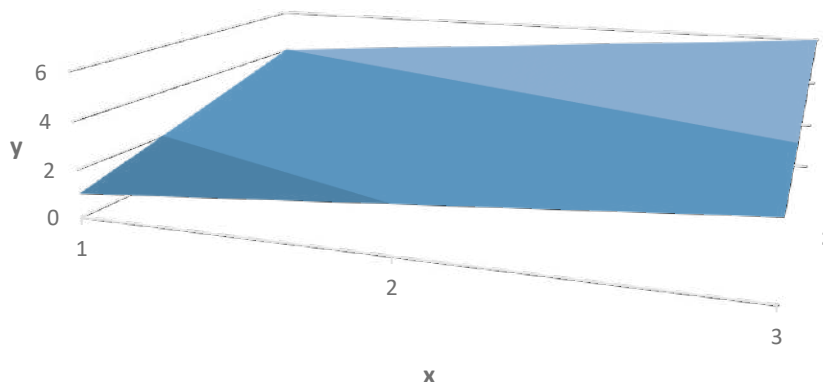
Mas, por limitações de dimensionalidade, não há como representar graficamente uma função com quatro ou mais variáveis independentes, tipo $z=f(x,y,z,k)$, porém a construção matemática de funções com n variáveis independentes não possui limitações.

Exemplificando: para entendermos como as teorias dos conjuntos da matemática estão tão intrinsecamente relacionadas às redes neurais artificiais, vamos levantar a necessidade de se obter a previsão se uma empresa entrará no cadastro da dívida ativa do Estado, com base nas seguintes variáveis independentes: idade da empresa, faturamento mensal, segmento

econômico, localidade da empresa e número de funcionários. Nesse nosso exemplo, são 5 variáveis independentes, isto é, $z=f(x,y,z,k,w)$, que estarão relacionadas a uma única variável dependente z , que dirá se a empresa será ou não devedora nos próximos 2 meses.

A representação gráfica para esse tipo de função segue abaixo:

Figura 2.3. Representação Gráfica de $z=f(x,y,z)$



Seria possível? Sim! Pois essa é a contribuição das Redes Neurais Artificiais (RNA), a análise preditiva, em que matematicamente se pode associar as variáveis independentes, no nosso exemplo x,y,z,k,w , à variável dependente z . Mas serão essas 5 variáveis independentes suficientes para se prever o comportamento em relação ao pagamento dos impostos, ou será mesmo que essas 5 variáveis são as melhores?

Bom, temos que pontuar, neste momento, que Inteligência Artificial (IA), ou mais corretamente para essa abordagem Inteligência Computacional (IC), possui muitos métodos e técnicas. RNA é uma das técnicas, mas para aplicá-la como qualquer outra técnica de IA ou IC, são necessários dados. O dado é a matéria-prima da informação, com esta obtemos o conhecimento necessário para termos a sabedoria na tomada de decisões. Dessa forma, temos que não só ter os dados mas também os melhores dados, as variáveis independentes, para que alcancemos o resultado mais próximo do correto, ou seja, a variável dependente.

Uma forma de se obterem os dados e se extraírem os que são mais correlacionados ao problema que se deseja resolver é aplicar as técnicas de mineração de dados. Vale ressaltar que RNAs não são úteis apenas para análise preditiva, podem também ser utilizadas para classificação.

Dada essa introdução, podemos então complementar a definição inicial de RNA, uma técnica de IC fundamentada em modelos matemáticos baseados nas estruturas neurais biológicas. Essas estruturas fazem uso de dados para aprender com o entendimento de que são funções matemáticas com múltiplas variáveis independentes, as quais se relacionam com uma variável ou mais variáveis dependentes, que representam um resultado.

1. Abordagem Neural

Agora, vamos entender como criar uma Rede Neural Artificial. Para isso, precisamos dar o primeiro passo, que é escolher o modelo adequado ao problema que se deseja atacar, ou como se diz atualmente, qual a dor que se quer resolver.

E por que escolher um modelo de RNA? Porque cada problema possui suas características, que podem ser de classificação ou agrupamento ou otimização ou predição, e cada modelo tem sua eficiência mais adequada para um tipo ou outro. Assim, a escolha do modelo é fundamental para o sucesso na resolução do problema.

O próximo passo é a escolha da função de ativação da RNA. Existem várias, mas as mais utilizadas são: Sigmoide, Tangente Hiperbólica (TanH), Unidade Linear Retificada (ReLU) e Unidade Linear Exponencial (ELU). Para escolher qual se deve utilizar, é necessário entender como elas se relacionam com o problema dos gradientes. Para isso, pode-se utilizar um software de gerenciamento da RNA, pois é a maneira mais prática de gerar a função de ativação e gerenciá-la.

Segue tabela com alguns dos softwares utilizados para gerar e gerenciar funções de ativação em Redes Neurais Artificiais.

Tabela 3.1. Software de Gerenciamento de RNA

Software	Endereço
Matlab	https://www.mathworks.com
NeuroDimension	http://www.neurodimension.com/
Orange Canvas	https://orange.biolab.si/
TensorFlow	https://www.tensorflow.org/
Visual R	https://www.r-project.org/
Wolfram Mathematica	http://www.wolfram.com/

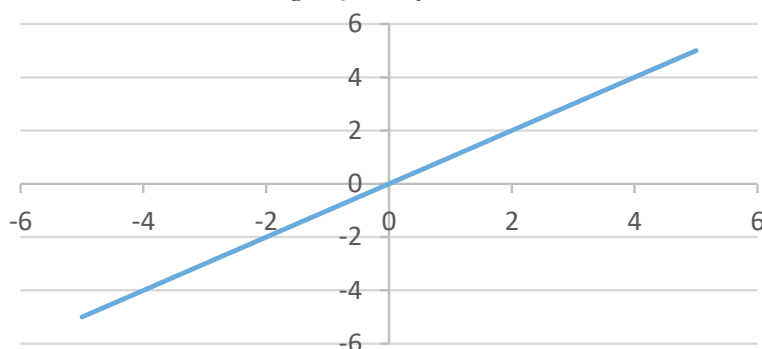
Mas o que é uma função de ativação? Antes de irmos ao terceiro passo da construção de uma rede neural, vamos entender as funções e os modelos.

Porém, antes de explicar as funções de ativação, é necessário entender o porquê delas. Na seção de introdução, já explicamos as funções, mas qual a necessidade de uma função específica denominada de ativação? De forma rápida, podemos dizer que as funções de ativação melhoram o processo de aprendizagem das variáveis dependentes e independentes, em especial para os problemas não lineares.

Então, antes de continuarmos, o que são problemas lineares e não lineares? Vamos entender primeiro os lineares.

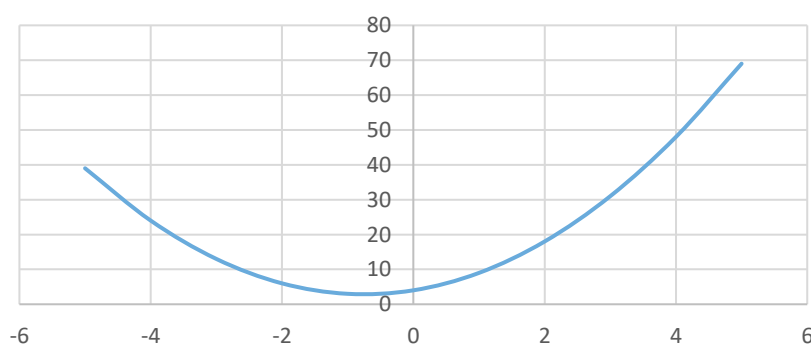
Uma função linear é expressa matematicamente pela forma $f(x)=a.x$, onde a é real e diferente de zero. Essa função é um tipo especial de função do 1º grau ou função afim, que é expressa pela forma $f(x)=a.x+b$, na qual a e b são reais e $a \neq 0$. Simplificando, toda função que não possui valor para o coeficiente b é classificada como função linear.

Figura 3.1. Função Linear



Alguns problemas podem ser resolvidos com funções lineares, mas não todos, pois existem aqueles que são não lineares, como os problemas de fenômenos físicos ou econômicos, que normalmente são sistemas de equações não lineares. Esse tipo de sistema normalmente não gera apenas um resultado para y em função de x , assim um gráfico de uma função não linear pode ser apresentada como uma parábola, com curvas ou retas não lineares, obtendo-se muitos valores de y para um x .

Figura 3.2. Função Não Linear



Entendido o que são funções lineares e não lineares, vamos para funções de ativação. Essas funções são expressões matemáticas, que permitem introduzir características de não linearidade às redes neurais artificiais, pois como a maioria dos problemas reais não são lineares, torna-se necessário usá-las nas RNAs. Mas a sua implementação, apesar de

necessária, no algoritmo de Redes Neurais Artificiais pode gerar alguns problemas de gradiente, como o de dissipação, o de gradiente explodindo ou desvanecendo.

Mas o que é gradiente? Consiste no cálculo de derivadas parciais, isto é, o cálculo dos elementos de uma função, tipo o x, y, z, para se obter qual x que pode maximizar o valor da função. Para isso, deve-se calcular a derivada parcial da função em relação a x.

Entendido o que é gradiente, o que significa ter o problema de dissipação de gradiente ou de gradiente explodindo ou desaparecendo?

Gradiente explodindo, problema que pode ocorrer se os pesos da rede forem grandes e os vieses não forem muito pequenos, provavelmente resultando em valores muito altos, a explosão do gradiente; o inverso desse problema é o gradiente que desaparece, porém o problema mais comum é a dissipação do gradiente, isto porque, para evitar os problemas anteriores, se deve ter o cuidado na aplicação do valor do peso, nem muito grande nem muito pequeno, na razão de $w * a + b$ da função, onde w é o peso. Agora se o intervalo de valores for muito estreito, o que pode ocorrer, às vezes, é a dissipação dos gradientes.

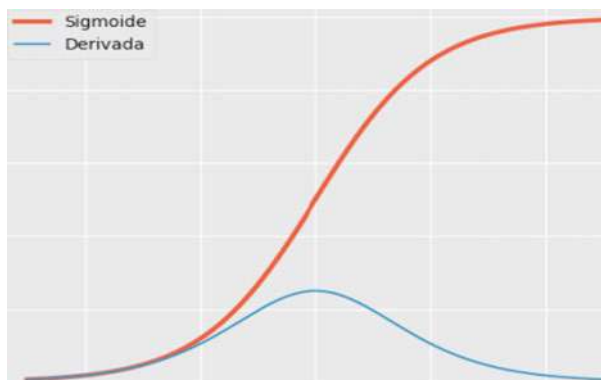
Agora que entendemos o porquê das funções de ativação, vamos falar sobre as principais.

A função sigmoide ou logística e sua derivada são expressas matematicamente por $\sigma(x) = 1 / (1 + e^{-x})$ e por $\sigma' = \sigma(x)(1 - \sigma(x))$, tendo esse nome porque na sua resolução passa a ter um formato de “S”, sendo uma das funções mais utilizada em RNAs, pois funciona de forma binária, isto é, não ativando “0” ou ativando “1”.

Mas o problema dessa função é que fazendo uso de valores acima de 5 ou abaixo de -5, a sua derivada fica saturada, passando a tender a zero, com a propagação do gradiente desvanecendo nessas regiões, o que ocasiona problema durante o treinamento.

Outro detalhe importante é que a derivada da função sigmoide é sempre < 1 , também sendo um problema, pois desvanece o produto pela regra da cadeia na propagação dos gradientes.

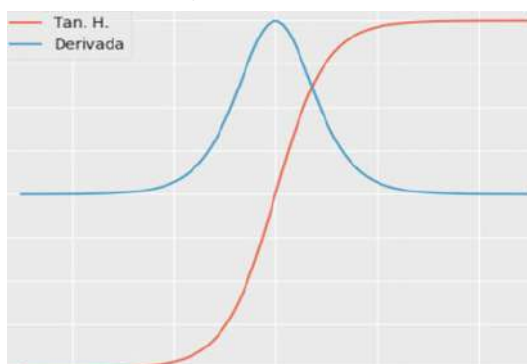
Assim sendo, não é uma boa função de ativação para as RNAs em problemas não lineares, mas pode ser útil em algumas situações, como: na saída das RNAs, na modelagem de variáveis binárias, em alguns modelos probabilísticos, ou nas redes neurais recorrentes e em alguns modelos não supervisionados. Mas, por causa dos problemas acima citados, normalmente ela não é mais usada nos atuais modelos de redes neurais artificiais.

Figura 3.3. Função Sigmoide ou Logística

A função Tangente Hiperbólica (TanH) possui semelhanças com a função sigmoide, pois, também na sua resolução, tem um formato de “S”, porém variando de -1 a 1, em vez de 0 a 1 como na sigmoide.

A função e sua derivada são expressas pela forma $\text{TanH}(x) = 2\sigma(2x) - 1$ e $\text{TanH}'(x) = 1 - \text{TanH}^2(x)$. Essa função permite uma melhor proximidade da identidade, tornando-se uma melhor alternativa em relação à sigmoide como função de ativação às camadas ocultas das RNAs.

Nessa função, também existe o problema de saturação, porém o valor da derivada é maior, podendo chegar a 1 quando $x = 0$. Assim, quando nos casos acima citados, em que existe a indicação do uso da função sigmoide, recomenda-se o uso da TanH no seu lugar.

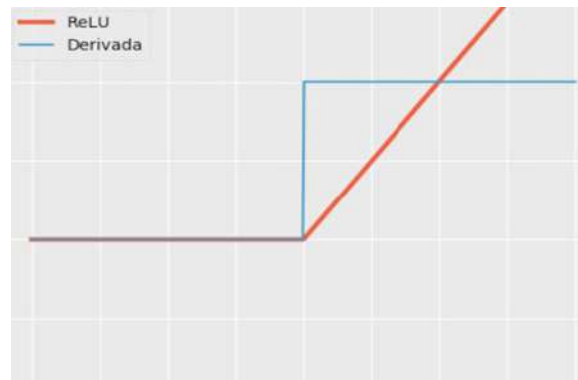
Figura 3.4. Função Tangente Hiperbólica (TanH)

A função Ativação Linear Retificada (ReLU) é expressa matematicamente por $\text{ReLU}(x) = \max\{0, x\}$ e $\text{ReLU}'(x) = \{1, \text{se } x \geq 0; 0\}$. Uma característica dessa função é a facilidade de otimizá-la, uma vez que é semelhante à função identidade, com o detalhe que ela gera zero na metade do seu domínio, assim as derivadas se mantêm grandes e estáveis enquanto a unidade estiver ativa.

A função de ativação ReLU é mais eficiente que as funções acima vistas, tendo forte influência para a popularização do Deep Learning. Porém, uma desvantagem da ReLU é que

os neurônios passam a produzir apenas zeros durante o treinamento, o que ocorre quando a soma ponderada antes da aplicação da ReLU se torna negativa, fazendo com que o neurônio produza zero e os parâmetros do peso fiquem atualizados. Além de os seus cálculos serem muito pesados, fazendo com que as RNAs que as utilizam fiquem muito lentas.

Figura 3.5. Função Ativação Linear Retificada (ReLU)

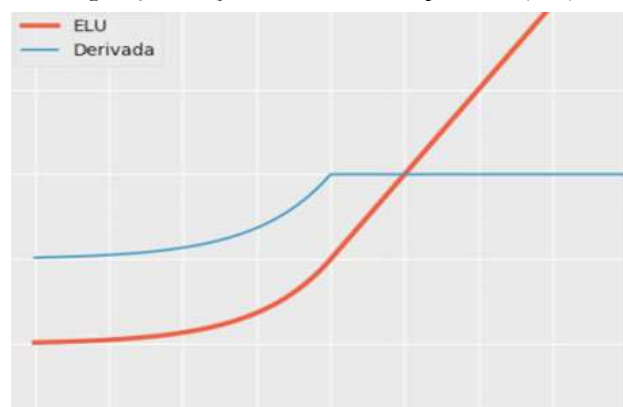


A função de ativação denominada de Unidade Linear Exponencial (ELU) é expressa por:
 $ELU(x, \alpha) = \begin{cases} x, & \text{se } x \geq 0; \\ \alpha(e^x - 1), & \text{se } x < 0; \end{cases}$ e $ELU'(x, \alpha) = \begin{cases} 1, & \text{se } x \geq 0; \\ ELU(x, \alpha) + \alpha, & \text{se } x < 0; \end{cases}$

A ELU não tem o problema dos neurônios “mortos” existente na ReLU, mas a não linearidade da ELU também satura na parte negativa do seu domínio, porém, na prática, não tem se apresentado como um problema.

Essa função é mais eficaz, mas como tudo não é perfeito, o cálculo da função exponencial é ineficiente, fazendo com que as RNAs que as utilizam sejam menos eficientes que as ReLU.

Figura 3.6. Função Unidade Linear Exponencial (ELU)



Em termos de evolução das funções de ativação e das características que a tornam mais eficientes, temos, em primeiro lugar, a ELU, seguida da ReLU, depois a TanH e, por fim, a Sigmoid. Bom, agora que foram apresentadas as funções de ativação, vamos continuar com os passos para a construção de uma rede neural artificial.

O próximo passo consiste em disponibilizar os dados para a RNA, e quanto mais dados, melhor, pois a probabilidade de sucesso no aprendizado está relacionada à quantidade de dados disponibilizados, isto é, quanto mais dados legados forem fornecidos, maior será a acurácia da função para prever comportamentos futuros.

Com os dados disponibilizados, podemos ir ao passo seguinte, que é o treinamento da RNA. Para realizar esse treinamento, devemos dividir os dados em dois grupos: o primeiro será utilizado para ser processado pelo modelo escolhido, que irá ler e reler os dados várias vezes, de várias maneiras diferentes. Ao final dessa etapa, que pode demorar de minutos a até várias horas, dependendo da capacidade do processador, da quantidade de dados e do número de variáveis independentes, será obtida uma função, que poderá retornar à variável dependente as respostas históricas referentes aos dados disponibilizados.

Com o treinamento realizado, deve-se agora ir ao passo de validação de RNA. Para validar, usa-se o segundo grupo, separado previamente. Essa separação dos dados se faz necessária para evitar o enviesamento do treinamento, isto é, impedir influência, o que é chamado de viés, sobre os dados, pois validar sobre os dados treinados poder gerar resultados artificialmente corretos.

Essa fase, a de validação, tem como objetivo, comparar as respostas verdadeiras dos fatos ocorridos com as respostas geradas pela função, podendo-se, assim, obter o percentual de acertos, indicando que a RNA está gerando resultados satisfatórios, mas caso não esteja satisfazendo as expectativas, deve-se retornar à fase de treinamento, seguida de uma nova fase de validação para confirmar a melhoria da eficiência da função, repetindo-se esses dois passos até que os resultados sejam satisfatórios.

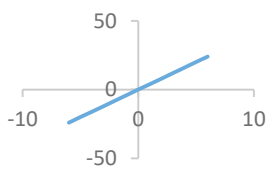
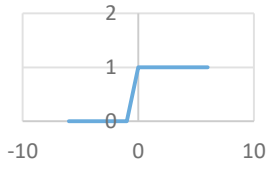
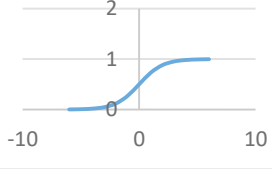
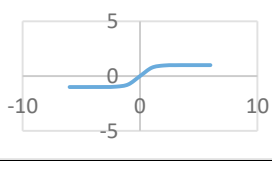
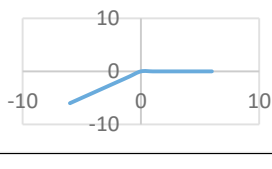
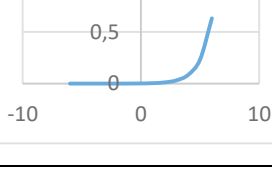
E por fim, chega-se ao último passo, que é o de receber novos valores para as variáveis independentes, aplicar o modelo testado e validado e analisar as respostas obtidas para a variável dependente.

Figura 3.7. Passos para a Aplicação de uma RNA



Para finalizar, segue uma tabela com os gráficos que representam as principais funções de ativação.

Tabela 3.2. Gráficos das Funções de Ativação

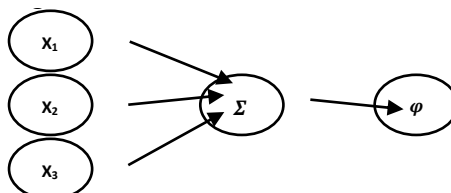
Nome	Plote	Equação	Derivada	Range
Linear (Identidade)		$f(x)=x$	$f'(x)=x$	$(-\infty, \infty)$
Degrau (Binária)		$f(x)=1; \text{ se } x \geq 0$ $0; \text{ se } < 0$	$f'(x)=?; \text{ se } x=0$ $0; \text{ se } < > 0$	$(0,1)$
Sigmoide (Logística)		$f(x)=1/(1+e^{-x})$	$f'(x)=f(x)(1-f(x))$	$(0,1)$
Tangente Hiperbólica		$f(x)=(e^x - e^{-x}) / (e^x + e^{-x})$	$f'(x)=1 - f(x)^2$	$(-1,1)$
Rectifier Linear (ReLu)		$f(x)=x; \text{ se } x \geq 0$ $0; \text{ se } < 0$	$f'(x)=1; \text{ se } x \geq 0$ $0; \text{ se } < 0$	$(0, \infty)$
Leaky ReLu		$f(x)=x; \text{ se } x \geq 0$ $0,01x; \text{ se } < 0$	$f'(x)=1; \text{ se } x \geq 0$ $0,01; \text{ se } < 0$	$(-\infty, \infty)$
Soft Max		$f(x)=(e^{x_i}) / (\sum_{j=1}^K e^{x_j})$	$f'(x)=f(x)(1-f(x))$	$(0,1)$

2. Evolução

Agora que foi explicado basicamente o que é uma Rede Neural Artificial, vamos conhecer um pouco da sua história, como ela surgiu e evoluiu.

As Redes Neurais Artificiais são consideradas modelos computacionais não algorítmicos e tiveram como inspiração inicial a rede de neurônios do cérebro humano, isto é, os axônios, dendritos e sinapses, tendo o primeiro modelo proposto por Warren McCulloch e Walter Pitts, um fisiologista e um matemático respectivamente, em 1943. Esse primeiro modelo foi proposto em um artigo intitulado “A Logical Calculus of the Ideas Immanent in Nervous Activity”, em que foi proposta a analogia entre um neurônio biológico e um processador eletrônico, chamado de neurônio McCulloch-Pitts.

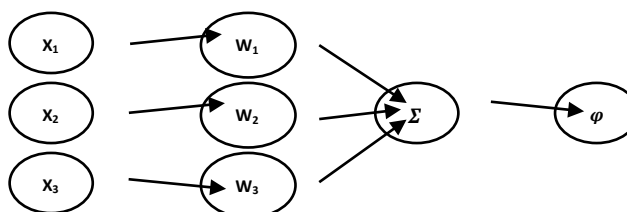
Figura 4.1. Neurônio McCulloch-Pitts



Sete anos depois, em 1950, Frank Rosenblatt publicou o artigo “The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain”, que aperfeiçoa as ideias de McCulloch e Pitts, criando uma rede com vários neurônios, denominada de Perceptron.

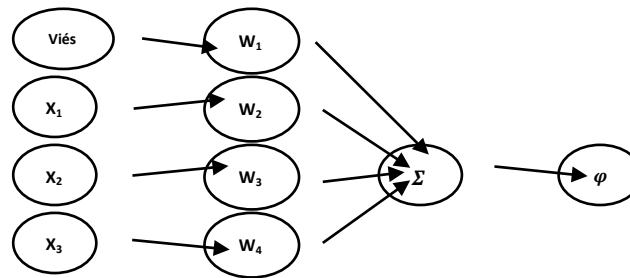
Esse foi o modelo que desencadeou todos os demais que viriam depois, composto basicamente de três camadas, a de entrada, a intermediária e a de saída.

Figura 4.2. Perceptron



Dez anos após a criação do modelo Perceptron por Rosenblatt, em 1960, Bernard Widrow e Marcian Hoff melhoram o Perceptron, criando o ADaptive LInear NEuron (ADALINE), que implementava ao modelo anterior a multiplicação das entradas pelos pesos, incluindo a soma de um viés. Logo, em seguida, foi aprimorado com a criação do Many ADALINE (MADALINE), que utilizava vários ADALINEs em paralelo com apenas uma saída.

Figura 4.3. ADALINE



No início dos anos de 1980, Kohonen desenvolve o modelo Self-Organizing Maps (SOM), e Paul Werbos desenvolve o Backpropagation, mas foi com as pesquisas de Gail Carpenter e Stephen Grossberg que as pesquisas foram intensificadas. Importante salientar que se passaram uns dez anos de baixas contribuições nos estudos de redes neurais artificiais. Mas, com o desenvolvimento do modelo não supervisionado Adaptive Resonance Theory (ART), retomou-se o crescente interesse por essas pesquisas.

3. Características

Com a apresentação do histórico das RNAs, como elas surgiram e se desenvolveram, vamos falar de alguns pontos relevantes da evolução. As redes neurais como visto é um algoritmo, porém não do tipo tradicional, o que chamamos de algoritmos dedutivos, e sim, algoritmos indutivos ou abduativos. Esse tipo de algoritmo utiliza dados para aprender, seja de forma supervisionada, não supervisionada ou por reforço, mas o que é importante é que aprende como nós, seres humanos, aprendemos através de generalizações. Então, os dados são de suma importância nesse processo de aprendizado.

Figura 5.1. Método Indutivo



Assim, em alguns modelos de RNAs, os dados que foram inseridos durante o processo de aplicação do modelo continuarão sendo atualizados e ajustados, objetivando melhorar, cada vez mais, sua precisão da função, simbolizada por caractere do alfabeto grego, que tem a pronúncia de fi.

Então, as redes neurais se diferem da programação tradicional, por que a estrutura do software não é codificada por um programador, estruturando as possíveis soluções baseadas em regras pré-definidas, mas sim, no aprendizado constante baseado na generalização. Dessa forma, quanto mais dados, mais características de algo a RNA irá aprender através da generalização. Explicando: se o desejo é fazer com que a rede distinga um cachorro de um lobo, deve-se fornecer dados para que ela aprenda, por características de cada objeto, o que é um e o que é outro. Quanto mais dados mais ela será melhor, aliás melhor será sua acurácia, grau de acerto.

Vale ressaltar que, sendo um algoritmo indutivo ou mesmo podemos dizer uma lógica não algorítmica, não se deve esperar um acerto de 100% em nenhuma das fases, seja de treinamento ou de produção, já que o que a rede faz é generalizar com base no seu aprendizado. Quanto mais sabe, mais acerta, mas acerta tudo e sempre não é uma característica humana, e

assim sendo, também não é de uma rede neural artificial, já que essa foi construída análoga a uma rede neural biológica.

Explicado isso, entenda que as RNAs não são soluções para todos os tipos de problemas, e sim para os que não possuem uma possível solução exata. Exemplo, se o problema é definir a classificação da dívida ativa do Estado, através de uma regra, tipo, se dívida acima de 1 milhão serão A, entre 500 mil e 1 milhão serão B, de 100 mil a 500 mil serão C e abaixo de 100 mil serão D, é uma aplicação de algoritmo dedutivo ou como normalmente chamamos de tradicional. Agora, se o desejo for classificar a dívida em A, B, C ou D, com base em características das dívidas, características essas que serão agrupadas por similaridades, que aplicada à resolução humana daria muito trabalho, ou mesmo seria irascível, dado a quantidade de campos e o volume de dados, temos uma aplicabilidade para as RNAs.

Outro detalhe importante é que não existe a garantia de alto grau de acerto numa predição, no uso de uma função complexa na implementação e que seus parâmetros foram ajustados da melhor forma possível a uma realidade já passada. Porém para esses problemas, será melhor ter um razoável grau de acerto, mesmo que entre os 80% e 90%, que não ter nenhum.

Uma capacidade importante das RNAs é a generalização, que extrai um dado útil a partir de um conjunto de dados incompletos ou sem integridade, o que só é possível devido à estrutura algorítmica de neurônios, que torna a arquitetura das RNAs bem tolerantes a falhas, já que o erro de um neurônio poderá ser corrigido através das saídas dos neurônios vizinhos.

Também é necessário saber que existem três tipos de aprendizado das RNAs: o supervisionado, o não supervisionado e o por reforço.

O aprendizado supervisionado se baseia em dois conjuntos de dados, um de entradas e outro de saídas correspondentes, em que na fase de treinamento, as entradas são apresentadas à função da rede, que irá verificar o quanto as saídas calculadas correspondem às saídas previamente conhecidas. Se não forem, a rede irá proceder com os ajustes dos pesos, armazenando o conhecimento exigido, repetindo o processo com os mesmos dados de entrada e saída até que a taxa de acertos da rede esteja dentro de uma faixa satisfatória, que não quer dizer 100%, pois isso não é possível no mundo real.

No aprendizado não supervisionado, não é utilizado um conjunto de dados de saída, apenas o conjunto de entradas, pois a rede irá classificar os dados de entrada com base nas características comuns que eles possuírem, ou seja, os neurônios são usados como classificadores, e os dados, como elementos de classificação, mas lembre-se de que é tudo matemática. Exemplificando uma técnica muito usada, a KNN, que escolhe elementos aleatórios do conjunto de dados e calcula seus centroides e depois agrupa os demais elementos por aproximação à centroide que foi obtida, resultando num agrupamento de dados por semelhanças, nesse caso, por proximidade.

O terceiro tipo é o das redes de aprendizado por reforço, que consiste em avaliar como um elemento irá agir de forma que maximize a recompensa em um período de tempo, e o que o torna diferente do aprendizado supervisionado é que não são dados de entrada e saída corretos, já que o objetivo é a rede mapear as ações desejadas.

4. Entendimento

Vamos entender, então! Olhando para as imagens abaixo, nós podemos rapidamente reconhecê-las, afirmando ser um cachorro e um lobo, com um grau elevado de precisão, que, com o passar dos anos, nos faz errar raramente na classificação.

Figura 6.1. Imagem de Cachorro e Lobo



Isso ocorre porque nossos neurônios biológicos conseguem processar uma grande quantidade de informações para reconhecimento de padrões contidos nas imagens com muita acurácia, isto é, acerto. Mas esse tipo de atividade é incrivelmente difícil para as máquinas, os neurônios artificiais.

E é exatamente esse tipo de problema que se busca resolver com os algoritmos não tradicionais, ou melhor dizendo, com a inteligência artificial. Mas se nós, máquinas de carbono, já resolvemos, por que queremos que as máquinas de silício resolvam? Porque elas resolvem mais rápido, melhor e não se cansam.

Vale salientar, porém, que determinadas tarefas são difíceis para os seres humanos e fáceis para as máquinas, como multiplicar milhões de número. Por outro lado, algumas tarefas são difíceis para os computadores tradicionais, mas fáceis para os seres humanos, como o reconhecimento facial.

Tabela 6.1. Matriz das Máquinas

Problema	Máquina de Carbono	Máquina de Silício
Aritmética na casa dos milhões	Difícil	Fácil
Reconhecimento Facial	Fácil	Difícil

Então distinguir um cachorro de um lobo para nós é algo fácil, mas não é uma tarefa trivial para os computadores, e é aí que pode entrar a Inteligência Artificial, fazendo o que não é trivial para a lógica tradicional.

O que se busca com a IA é fazer com que as máquinas pensem de forma semelhante ao homem, mas as máquinas não pensam, elas calculam, então o que se faz é computar de forma análoga ao pensamento humano. O que de forma básica ocorre é que um computador recebe dados como entrada, por exemplo, dois números, e esses são processados, e o resultado da operação é saído. A máquina de Von Neumann, entrada ► processamento ► saída.

Entendido que isso é trivial, imaginemos agora que você quer classificar a sociedade em indivíduos da classe A, B, C e D, o que significa, respectivamente, classe alta, classe média, classe baixa e abaixo da linha de pobreza.

Para fazer essa classificação, com parâmetros definidos e conhecendo as regras, é relativamente simples. Visando entender, vamos considerar apenas poucas variáveis e uma regra básica: a variável independente será o salário, a entrada de dados, a variável dependente será a classe, a saída de dados, e a regra simples será, se salário acima de 1000 será A, entre 100 e 999 será B, entre 10 e 99 será C e abaixo de 9 será D. Esse é um típico problema dedutivo, pois na relação $y=f(x)$, você tem o x que é a entrada, o salário, você, o f que é a regra, basta executá-la que terá o y a saída, que é a classe.

Agora imagine que você deseja classificar uma sociedade em A, B, C e D, então você sabe que saída quer, você sabe qual é a sua variável dependente, o y da questão, a classe social, mas você não sabe ou não tem todas as variáveis independentes, ou seja, as entradas, os $x(s)$, porém você sabe qual regra usar, o f da função, que será o condicionamento dessas variáveis independentes. Nesse caso, você tem um problema indutivo e não dedutivo como o anterior.

Mas, imagine agora que além de não ter todos os $x(s)$, você também não sabe que regra usar, qual será o f , nesse caso, você tem um problema abdutivo. Seja um problema indutivo ou abdutivo, você terá um problema complexo para ser solucionado. Você terá um problema de Inteligência Artificial, pois um algoritmo tradicional, isto é, dedutivo, não tem como resolvê-lo ou o tempo para resolvê-lo será inviável, como o problema do caixeiro viajante.

Quando estamos resolvendo um problema dedutivo, fazemo-lo seguindo uma receita de bolo: se isso, faça isso; se aquilo, faça assado, e assim por diante. Mas se estamos resolvendo um problema indutivo ou abdutivo, fazemos por tentativa e erro, isto é, por aprendizado, tentando uma resposta melhor repetidamente até chegarmos à melhor resposta para aquele momento, a heurística.

Veja, nós humanos, quando damos determinadas respostas, não somos precisos como máquina, mas damos um retorno aproximado, o quão aproximado a resposta for, melhor será o seu retorno, melhor será sua acurácia. Exemplo: alguém lhe pergunta: Quanto tempo demora para chegar ao ponto B, saindo de A? Você não é uma máquina para dar uma resposta precisa em horas, minutos e segundos em relação à distância exata em quilômetros! Você é um humano e dará uma resposta aproximada. Quanto melhor a aproximação, melhor será sua resposta. Concorde? Bom, é assim com Inteligência Artificial, não buscamos os 100%, mas algo entre 80% e 90%, para ser uma boa resposta.

Como para nós, máquinas de carbono, também para as máquinas de silício, consegue-se melhorar as respostas para as perguntas com o estudo, isto é, com o treinamento. Quanto mais se estuda, mais se treina, melhor será seu grau de acerto, acurácia, é assim que funciona com ambas as máquinas, as que usam os neurônios biológicos e as que usam os neurônios artificiais.

Mas, entenda! Todos os computadores precisam de entradas para que sejam processadas e gerem uma saída. Isso independe se é uma lógica dedutiva, indutiva ou abdutiva. As redes neurais não são diferentes.

Então, para que serve uma solução de inteligência artificial? Serve quando não sabemos exatamente como algo funciona, fazendo uso de estimativa com base em algum modelo no qual possamos ajustá-lo, objetivando refiná-lo com base no quão errado ele está em comparação ao que desejamos.

Até este ponto, entendemos o que é IA. Vamos agora entender como usá-la. Ela pode ser usada para resolver problemas de classificação, agrupamento, predição, otimização e busca. Mas, como esse material tem a missão de dar o entendimento básico, vamos nos concentrar em classificação e predição.

A classificação nos servirá para agrupar, por características comuns, o que chamamos de generalização. Por outro lado, predição servirá para prever o comportamento de algo com base na comparação das características comuns analisadas anteriormente.

Para entender melhor o que será explicado, vou usar uma ferramenta denominada de Orange Canvas, que é um software de mineração de dados open source, desenvolvido pela Universidade Ljubljana da Eslovênia, podendo ser baixado pelo endereço eletrônico <https://orange.biolab.si/>. Também para explicar os conceitos de Inteligência Artificial desse ponto adiante, usarei a base de dados pública de íris, que são dados de uma flor.

Na imagem acima, pode-se observar que, apesar de serem a mesma flor, chamadas de

íris, as flores possuem pétalas de tamanhos e larguras variadas, isto é, características distintas para a mesma flor, o que pode ser utilizado para gerar agrupamentos, isto é, classificações.

Figura 6.2. Imagens de Flores de Íris



A base de dados das flores de íris é composta pelos seguintes metadados: nome da íris, comprimento e tamanho da pétala e da sépala, com 150 ocorrências, que estão expressos na tabela a seguir:

Tabela 6.2. Base de Dados das Flores de Íris

íris	sepal length	sepal width	petal length	petal width
Íris-setosa	5.1	3.5	1.4	0.2
Íris-setosa	4.9	3	1.4	0.2
Íris-setosa	4.7	3.2	1.3	0.2
Íris-setosa	4.6	3.1	1.5	0.2
Íris-setosa	5	3.6	1.4	0.2
Íris-setosa	5.4	3.9	1.7	0.4
Íris-setosa	4.6	3.4	1.4	0.3
Íris-setosa	5	3.4	1.5	0.2
Íris-setosa	4.4	2.9	1.4	0.2
Íris-setosa	4.9	3.1	1.5	0.1
Íris-setosa	5.4	3.7	1.5	0.2
Íris-setosa	4.8	3.4	1.6	0.2
Íris-setosa	4.8	3	1.4	0.1
Íris-setosa	4.3	3	1.1	0.1
Íris-setosa	5.8	4	1.2	0.2
Íris-setosa	5.7	4.4	1.5	0.4
Íris-setosa	5.4	3.9	1.3	0.4
Íris-setosa	5.1	3.5	1.4	0.3

Íris-setosa	5.7	3.8	1.7	0.3
Íris-setosa	5.1	3.8	1.5	0.3
Íris-setosa	5.4	3.4	1.7	0.2
Íris-setosa	5.1	3.7	1.5	0.4
Íris-setosa	4.6	3.6	1	0.2
Íris-setosa	5.1	3.3	1.7	0.5
Íris-setosa	4.8	3.4	1.9	0.2
Íris-setosa	5	3	1.6	0.2
Íris-setosa	5	3.4	1.6	0.4
Íris-setosa	5.2	3.5	1.5	0.2
Íris-setosa	5.2	3.4	1.4	0.2
Íris-setosa	4.7	3.2	1.6	0.2
Íris-setosa	4.8	3.1	1.6	0.2
Íris-setosa	5.4	3.4	1.5	0.4
Íris-setosa	5.2	4.1	1.5	0.1
Íris-setosa	5.5	4.2	1.4	0.2
Íris-setosa	4.9	3.1	1.5	0.1
Íris-setosa	5	3.2	1.2	0.2
Íris-setosa	5.5	3.5	1.3	0.2
Íris-setosa	4.9	3.1	1.5	0.1
Íris-setosa	4.4	3	1.3	0.2
Íris-setosa	5.1	3.4	1.5	0.2
Íris-setosa	5	3.5	1.3	0.3
Íris-setosa	4.5	2.3	1.3	0.3
Íris-setosa	4.4	3.2	1.3	0.2
Íris-setosa	5	3.5	1.6	0.6
Íris-setosa	5.1	3.8	1.9	0.4
Íris-setosa	4.8	3	1.4	0.3
Íris-setosa	5.1	3.8	1.6	0.2
Íris-setosa	4.6	3.2	1.4	0.2
Íris-setosa	5.3	3.7	1.5	0.2
Íris-setosa	5	3.3	1.4	0.2
Íris-versicolor	7	3.2	4.7	1.4

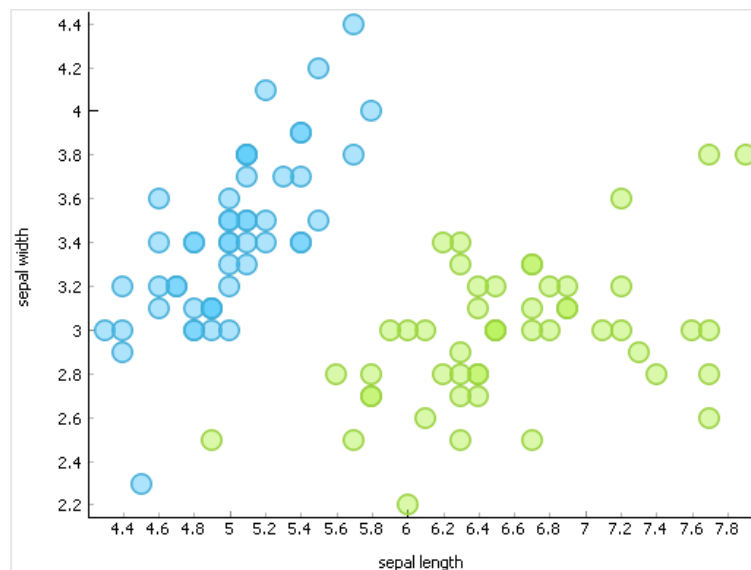
Íris-versicolor	6.4	3.2	4.5	1.5
Íris-versicolor	6.9	3.1	4.9	1.5
Íris-versicolor	5.5	2.3	4	1.3
Íris-versicolor	6.5	2.8	4.6	1.5
Íris-versicolor	5.7	2.8	4.5	1.3
Íris-versicolor	6.3	3.3	4.7	1.6
Íris-versicolor	4.9	2.4	3.3	1
Íris-versicolor	6.6	2.9	4.6	1.3
Íris-versicolor	5.2	2.7	3.9	1.4
Íris-versicolor	5	2	3.5	1
Íris-versicolor	5.9	3	4.2	1.5
Íris-versicolor	6	2.2	4	1
Íris-versicolor	6.1	2.9	4.7	1.4
Íris-versicolor	5.6	2.9	3.6	1.3
Íris-versicolor	6.7	3.1	4.4	1.4
Íris-versicolor	5.6	3	4.5	1.5
Íris-versicolor	5.8	2.7	4.1	1
Íris-versicolor	6.2	2.2	4.5	1.5
Íris-versicolor	5.6	2.5	3.9	1.1
Íris-versicolor	5.9	3.2	4.8	1.8
Íris-versicolor	6.1	2.8	4	1.3
Íris-versicolor	6.3	2.5	4.9	1.5
Íris-versicolor	6.1	2.8	4.7	1.2
Íris-versicolor	6.4	2.9	4.3	1.3
Íris-versicolor	6.6	3	4.4	1.4
Íris-versicolor	6.8	2.8	4.8	1.4
Íris-versicolor	6.7	3	5	1.7
Íris-versicolor	6	2.9	4.5	1.5
Íris-versicolor	5.7	2.6	3.5	1
Íris-versicolor	5.5	2.4	3.8	1.1
Íris-versicolor	5.5	2.4	3.7	1
Íris-versicolor	5.8	2.7	3.9	1.2
Íris-versicolor	6	2.7	5.1	1.6

Íris-versicolor	5.4	3	4.5	1.5
Íris-versicolor	6	3.4	4.5	1.6
Íris-versicolor	6.7	3.1	4.7	1.5
Íris-versicolor	6.3	2.3	4.4	1.3
Íris-versicolor	5.6	3	4.1	1.3
Íris-versicolor	5.5	2.5	4	1.3
Íris-versicolor	5.5	2.6	4.4	1.2
Íris-versicolor	6.1	3	4.6	1.4
Íris-versicolor	5.8	2.6	4	1.2
Íris-versicolor	5	2.3	3.3	1
Íris-versicolor	5.6	2.7	4.2	1.3
Íris-versicolor	5.7	3	4.2	1.2
Íris-versicolor	5.7	2.9	4.2	1.3
Íris-versicolor	6.2	2.9	4.3	1.3
Íris-versicolor	5.1	2.5	3	1.1
Íris-versicolor	5.7	2.8	4.1	1.3
Íris-virginica	6.3	3.3	6	2.5
Íris-virginica	5.8	2.7	5.1	1.9
Íris-virginica	7.1	3	5.9	2.1
Íris-virginica	6.3	2.9	5.6	1.8
Íris-virginica	6.5	3	5.8	2.2
Íris-virginica	7.6	3	6.6	2.1
Íris-virginica	4.9	2.5	4.5	1.7
Íris-virginica	7.3	2.9	6.3	1.8
Íris-virginica	6.7	2.5	5.8	1.8
Íris-virginica	7.2	3.6	6.1	2.5
Íris-virginica	6.5	3.2	5.1	2
Íris-virginica	6.4	2.7	5.3	1.9
Íris-virginica	6.8	3	5.5	2.1
Íris-virginica	5.7	2.5	5	2
Íris-virginica	5.8	2.8	5.1	2.4
Íris-virginica	6.4	3.2	5.3	2.3
Íris-virginica	6.5	3	5.5	1.8

Íris-virginica	7.7	3.8	6.7	2.2
Íris-virginica	7.7	2.6	6.9	2.3
Íris-virginica	6	2.2	5	1.5
Íris-virginica	6.9	3.2	5.7	2.3
Íris-virginica	5.6	2.8	4.9	2
Íris-virginica	7.7	2.8	6.7	2
Íris-virginica	6.3	2.7	4.9	1.8
Íris-virginica	6.7	3.3	5.7	2.1
Íris-virginica	7.2	3.2	6	1.8
Íris-virginica	6.2	2.8	4.8	1.8
Íris-virginica	6.1	3	4.9	1.8
Íris-virginica	6.4	2.8	5.6	2.1
Íris-virginica	7.2	3	5.8	1.6
Íris-virginica	7.4	2.8	6.1	1.9
Íris-virginica	7.9	3.8	6.4	2
Íris-virginica	6.4	2.8	5.6	2.2
Íris-virginica	6.3	2.8	5.1	1.5
Íris-virginica	6.1	2.6	5.6	1.4
Íris-virginica	7.7	3	6.1	2.3
Íris-virginica	6.3	3.4	5.6	2.4
Íris-virginica	6.4	3.1	5.5	1.8
Íris-virginica	6	3	4.8	1.8
Íris-virginica	6.9	3.1	5.4	2.1
Íris-virginica	6.7	3.1	5.6	2.4
Íris-virginica	6.9	3.1	5.1	2.3
Íris-virginica	5.8	2.7	5.1	1.9
Íris-virginica	6.8	3.2	5.9	2.3
Íris-virginica	6.7	3.3	5.7	2.5
Íris-virginica	6.7	3	5.2	2.3
Íris-virginica	6.3	2.5	5	1.9
Íris-virginica	6.5	3	5.2	2
Íris-virginica	6.2	3.4	5.4	2.3
Íris-virginica	5.9	3	5.1	1.8

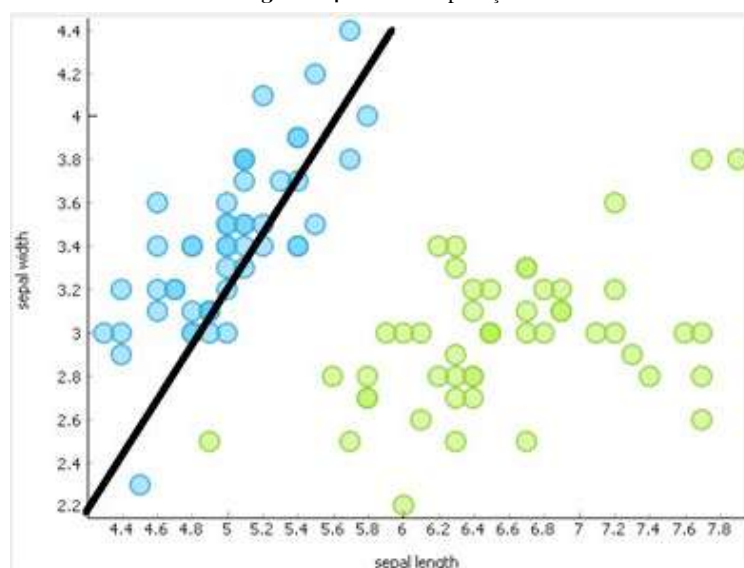
Vamos agora observar o gráfico de dispersão abaixo que mostra o agrupamento dos dados da tabela acima. Vemos que os dados que compõem a flor Íris-setosa, em azul, e da Íris-virginica, em verde, têm características comuns em relação a tamanho e comprimento da sépala, o que os deixa bem agrupados separadamente, com exceção dos dados com tamanho entre 2.2 e 2.4 e comprimento entre 4.4 e 5, que ficam bem próximos.

Figura 6.3. Gráfico de Dispersão



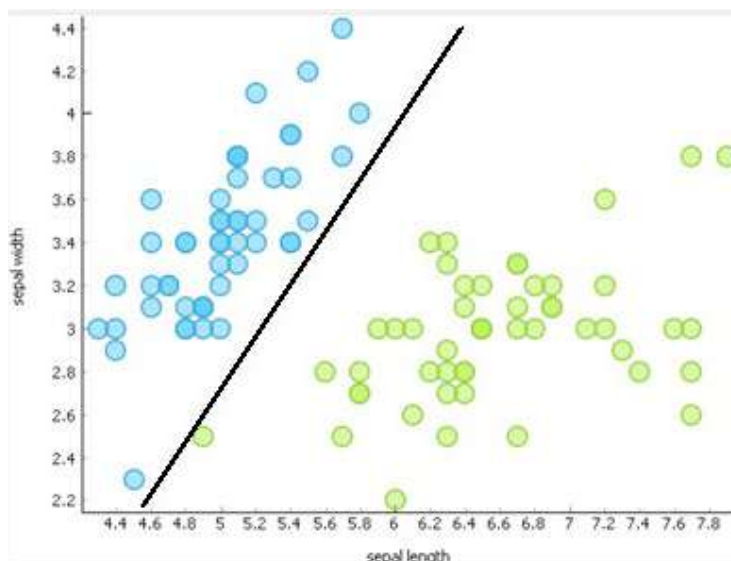
É isso que importa nas redes neurais artificiais, ocorrências que possuam dados significativos e que possam ser usados para agrupar por semelhanças. Mas se passarmos uma reta nesse gráfico, como vamos observar a seguir, vemos que podemos usar a linha para separar os diferentes tipos de Íris.

Figura 6.4. Reta de Separação



No gráfico acima, se a reta está dividindo as setosas, azuis, das virginicas, as verdes, então a reta poderia ser usada para classificar um bug em relação às medições. Porém, a reta acima não faz isso, ainda por que, parte das setosas está do mesmo lado da linha divisória das virginicas. Então, vamos ajustar a inclinação da reta até que tenhamos uma reta que separe da melhor forma os dois grupos.

Figura 6.5. Reta de Separação com Melhor Ângulo



A reta traçada acima separa corretamente as setosas das virginicas. Dessa forma, podemos usar essa linha como um separador, utilizando a linha divisória para classificar facilmente o que é setosa ou virginica.

Nesse momento, chegamos ao ponto crucial das redes neurais artificiais, que é como inclinar a reta para que separe corretamente os elementos em foco, isto é: Como podemos melhorar uma linha que sabemos não estar sendo o melhor divisor entre dois conjuntos de dados? A resposta é treinando a rede. E para entender o que é treinamento de uma rede neural, vamos observar os dados da tabela a seguir:

Tabela 6.3. Base de Dados das Flores de Íris

exemplo	sépala tamanho	sépala comprimento	íris
1	3.5	5.1	íris-setosa
2	3.3	6.3	íris-virginica

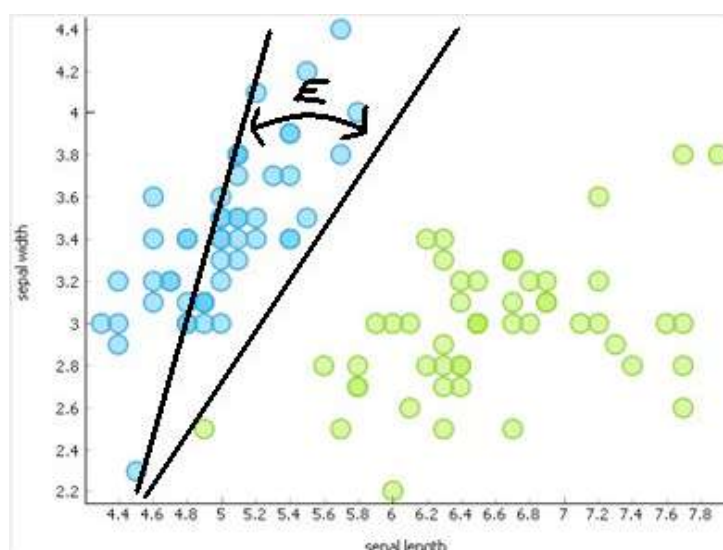
No exemplo acima, temos uma ocorrência que tem tamanho 3.5 e comprimento 5.1, o que sabemos é uma setosa. Temos também uma ocorrência de tamanho menor 3.3 e de comprimento maior 6.3, que é uma virginica. Esse é um conjunto para treinamento com

base em dados reais. E serão esses exemplos que ajudarão a refinar a inclinação da função de classificador. Dados reais, usados para ensinar um preditor ou um classificador, são denominados de dados de treinamento.

O processo se inicia traçando uma linha divisória aleatória, pois precisamos começar de algum ponto. Passaremos a chamar o tamanho de x e o comprimento de y , para termos nossa reta divisória. Se iniciarmos com $y=0.25x$, veremos que essa reta não é uma boa classificadora, pois a linha não divide os dois tipos de flores de Íris, a setosa e a virginica, assim precisamos mudar o ângulo da reta para movê-la.

Temos que testar os valores para chegarmos a uma reta que separe da melhor forma possível as características das flores de íris e, para chegarmos a ela, teremos que ir testando, observando as diferenças, isto é, os erros. O erro nos servirá para ajustarmos os parâmetros, a fim de testarmos novamente e repetirmos o processo até chegarmos à reta desejável. Usamos a letra E para significar o Erro, que é, na verdade, o destino desejado para a saída real.

Figura 6.6. Parâmetro de Erro (E)

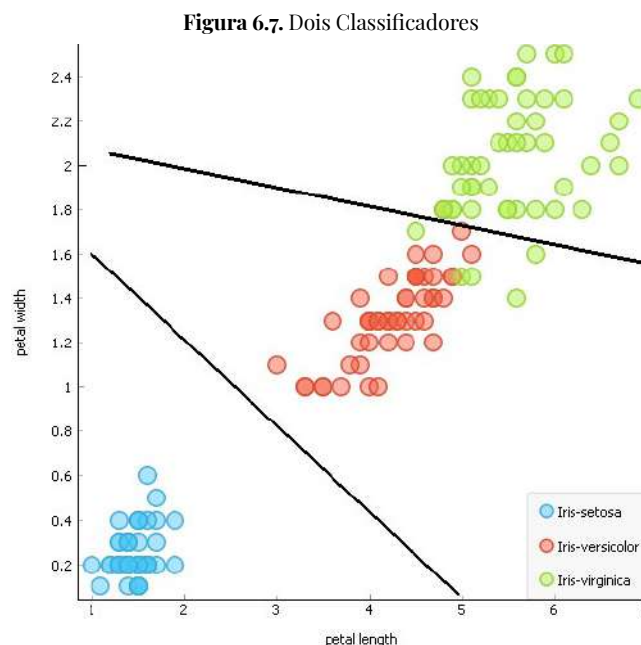


Com tudo o que foi mostrado até aqui, já podemos entender que redes neurais artificiais fazem uso puramente da matemática. Resumindo o que mostramos até aqui é que o erro de saída de um classificador linear é o nosso parâmetro de inclinação da reta, que irá separar os elementos analisados, mas podemos também expressar essa ideia da seguinte forma: realizamos o ajuste da inclinação da reta para remover o erro de saída. Tanto faz, dá no mesmo, entender de um jeito ou de outro.

Porém, é importante ressaltar que esse ajuste ocorre em relação ao último exemplo de treinamento, ignorando todos os exemplos de treinamento anteriores. Então uma técnica para

se corrigir isso é fazer uso de um parâmetro chamado de taxa de aprendizado, que servirá para atenuar esse problema, pois nenhum exemplo de treinamento único irá enviesar totalmente o aprendizado. Então não esqueça, sempre usar a taxa de aprendizagem, porque ela poderá atenuar os problemas de sujeira dos dados que podem existir na base de dados real.

Só que, às vezes, um classificar, isto é, uma reta divisória, não é suficiente. Observe o gráfico abaixo para entender isso. Se estivermos analisando não dois conjuntos, mas três ou mais conjuntos de dados, que é o caso se estivermos analisando as flores de íris setosa, virginica e versicolor, uma única reta não seria capaz de separá-los. E esse é o caso real; nos problemas da vida real, não analisamos apenas dois conjuntos de dados, mas sim um grande número deles.



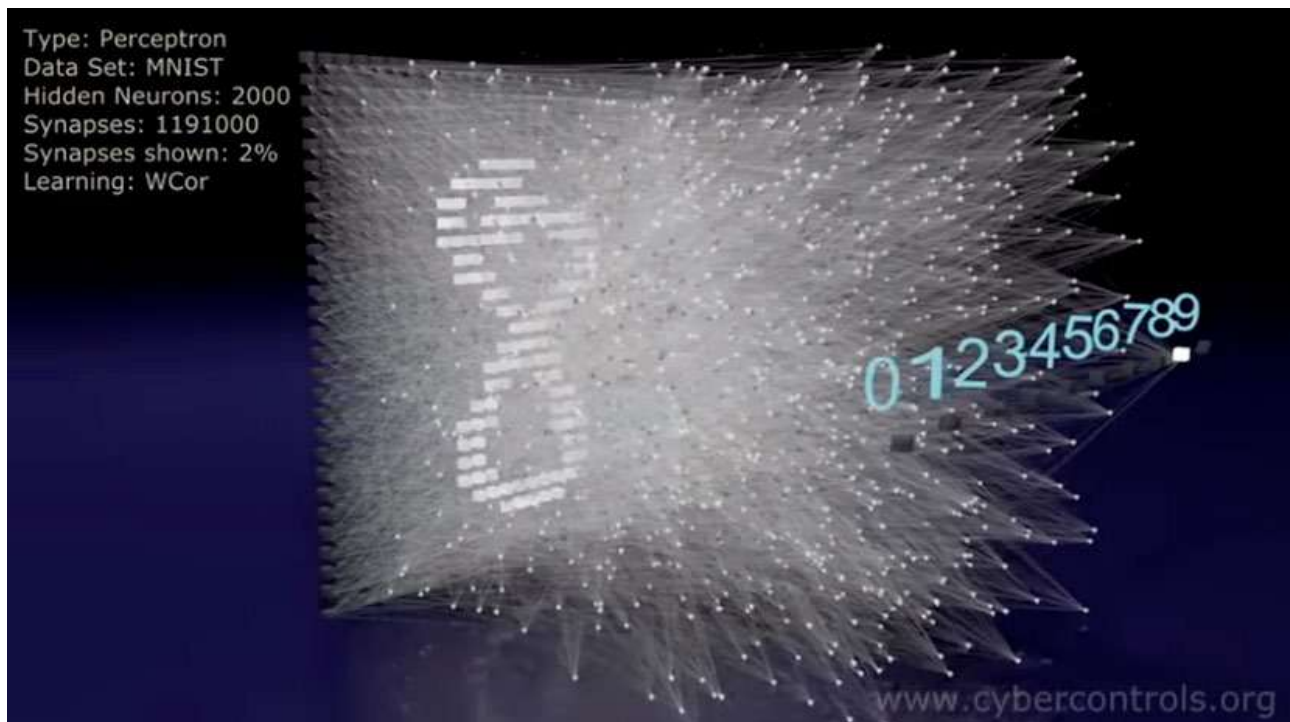
As redes neurais artificiais precisam resolver não apenas problemas lineares mas também problemas subjacentes, isto é, problemas não lineares, o que, como já dito, é a maioria dos problemas reais. Então já percebemos que as RNAs podem ter muito classificadores, isto é, muitas retas divisórias no mesmo gráfico.

A seguir, iremos ver a evolução das Redes Neurais Artificiais da mais básica, começando com a Perceptron, até as mais complexas, as atuais convolucionais.

5. Perceptron

Entre os anos 1950 e 1960, Rosenblatt e vários outros pesquisadores estavam propondo inovações nos modelos de redes neurais com base no modelo original de McCulloch e Pitts, objetivando usá-los em problemas de aprendizado. De todas as pesquisas dessa época, a que mais gerou impacto foi a de Frank Rosenblatt, que foi batizada de Perceptron.

Figura 7.1. Perceptron



Fonte: <https://youtu.be/3JQ3hYko51Y>

A proposta de Rosenblatt para o Perceptron foi resolver problemas de reconhecimento de padrões, atividade realizada por nós, máquinas de carbono, facilmente e de forma instantânea, mas, pelas máquinas de silício, é um dos maiores problemas para serem resolvidos.

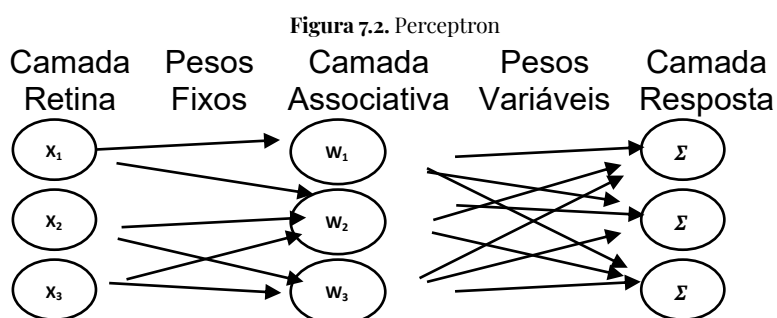
Na computação, o reconhecimento de padrões significa o processo pelo qual um sinal de entrada, o padrão, é classificado, rotulado, como pertencente a uma única classe de um conjunto de classes. Esse conjunto de classes para as quais os padrões são classificados pode já ter uma definição antes do início do processamento. Nesse modelo, o sistema aprende a classificar durante o treinamento denominado supervisionado, no qual os padrões previamente escolhidos e classificados com base na regra de classificação desejada lhe são apresentados.

O outro modelo é denominado de aprendizado não supervisionado, em que o sistema

aprende a classificar um conjunto de padrões por meio do próprio sistema, uma vez que o próprio sistema tem que determinar quais as classes de padrões devem ser classificadas, com base em propriedades intrínsecas dos padrões.

Podemos exemplificar como problemas de reconhecimento de padrões: informar o sexo de um indivíduo com base nas características faciais; determinar que animal é com base nos sons por ele produzido, até mesmo o reconhecimento de um indivíduo com base na sua voz; reconhecimento de objetos entre outros objetos; determinar que bebida é com base nas suas características ou a safra do vinho com base no seu aroma, o nariz artificial. São atividades realizadas relativamente com facilidade por um ser humano, mas que requerem grande esforço de processamento das máquinas.

O Perceptron, que foi concebido por Frank Rosenblatt na década de 1960, era composto pelas seguintes partes: camada de retina, camada associativa, camada de resposta, que respectivamente eram a camada de entrada, a camada oculta e a camada de saída, que possuíam um sentido único entre as conexões, que se aplicavam pesos constantes entre a entrada e a camada oculta e pesos variáveis entre a camada oculta e a camada de saída.



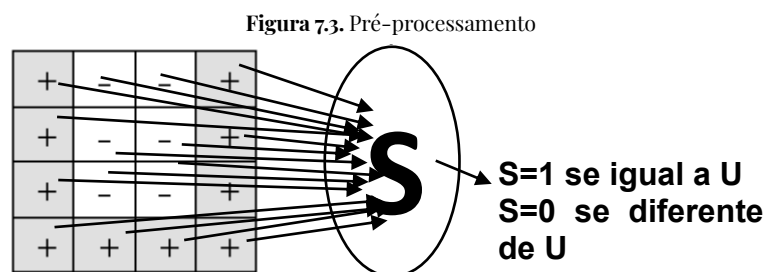
As conexões entre os neurônios artificiais seguem sempre o sentido para frente, denominado de feed-forward, da camada retina para a camada associativa e da camada associativa para a camada de resposta. Para as conexões, os pesos sinápticos entre a retina e a camada associativa não são totais, mas parciais, sendo que um neurônio x da camada associativa recebe sinapses apenas de uma parte da camada de retina, denominada de campo receptivo do neurônio x , com pesos fixos. Por outro lado, os pesos sinápticos entre a camada associativa e a camada de resposta são totais, pois cada neurônio da camada de resposta recebe conexões sinápticas de todos os neurônios da camada associativa, com pesos variáveis, sendo iniciado o processamento de treinamento com pesos aleatórios dentro de um intervalo -1 e $+1$.

Essa arquitetura proposta por Rosenblatt do Perceptron é baseada no sistema visual dos animais. Assim a camada retina, de entrada, baseada na retina da visão dos animais, é

responsável pela entrada de dados. Por outro lado, os neurônios da camada associativa são responsáveis pelo processamento das características e, por fim, a ligação com os neurônios de resposta.

O pensamento de Rosenblatt, com o estudo baseado na visão dos animais, possibilitou o entendimento de que uma forma de simplificar o reconhecimento dos padrões é a análise dos seus contornos, pois se verificou que as mudanças abruptas ocorrem no contorno dos objetos, assim as imagens podem ser decompostas em características com orientação da forma.

Para melhor entender a ideia de Rosenblatt, que gerou o modelo de Perceptron, segue a figura que tem como entrada de dados a letra U numa matriz 4 por 4.



Explicando essa figura: o que ocorre é um pré-processamento das características entradas na camada retina pela camada associativa, que consiste em uma série de operações de transformações aplicadas sobre os padrões de entrada para deixá-los em condições de serem reconhecidos pelos neurônios da camada de saída. Uma das funções do pré-processamento é a redução das dimensionalidades, visto que, no mundo real, essas dimensões são muito grandes.

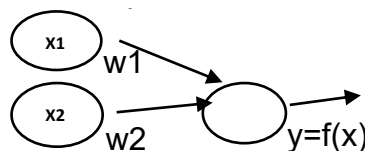
Assim o pré-processamento é uma etapa importante para a melhora do desempenho de um sistema de reconhecimento de padrões, seja numa rede neural ou não, visto que, se os pesos fixos dos neurônios da camada de entrada não forem adequados, ou o número de características for pouco ou muito, o problema pode não ter solução.

A camada de saída do Perceptron servirá para realizar a combinação das características detectadas na camada de entrada em um padrão único, o que acarretará a resposta do sistema. Assim sendo, o Perceptron é um sistema, que aprende de maneira supervisionada, uma vez que a tarefa de reconhecimento do padrão apresentado na entrada será realizada pela camada de saída.

O problema do Perceptron é encontrar os pesos das sinapses que conectam os neurônios da camada de entrada aos neurônios da camada de saída, de forma que, no seu treinamento, encontrem-se valores de pesos que convirjam para valores que gerem a classificação correta

dos padrões desejados. Para isso, Rosenblatt criou a regra de aprendizado do Perceptron. Essa regra é mais fácil de ser observada de maneira gráfica. Supondo que existam apenas dois neurônios na camada de entrada, x_1 e x_2 , e usando uma rede Perceptron com apenas um neurônio na camada de saída, temos a seguinte figura:

Figura 7.4. Perceptron de Frank Rosenblatt



Se para o viés do neurônio de saída tivermos $v=0$, então a função de ativação da unidade de saída será o somatório da multiplicação das entradas pelos pesos, tendo como saída a função binária para o resultado da expressão anterior como maior ou igual a 0, sairá 1, e caso contrário, se menor que 0, sairá 0.

Esse modelo de rede neural artificial é o mais simples que existe, sendo, por isso, importante entendê-lo bem, pois com essa base fundamentada, todos os demais modelos evolutivos ficaram mais simples de entender. Fixando: Redes Neurais Artificiais aprendem por exemplos.

Existem algumas considerações importantes que devemos citar nesse modelo Perceptron. A saída faz uso da Lei do Tudo ou Nada, cuja saída é expressa pela função degrau, que indica quando o neurônio está ativo (1) ou inativo (0), isto é, $y=f(\text{net})$, onde, se $\text{net} \geq 0$ então 1, caso contrário é 0.

A regra matematicamente é definida como:

$$\Delta w_{ij} = \alpha (d_i - y_i) x_j$$

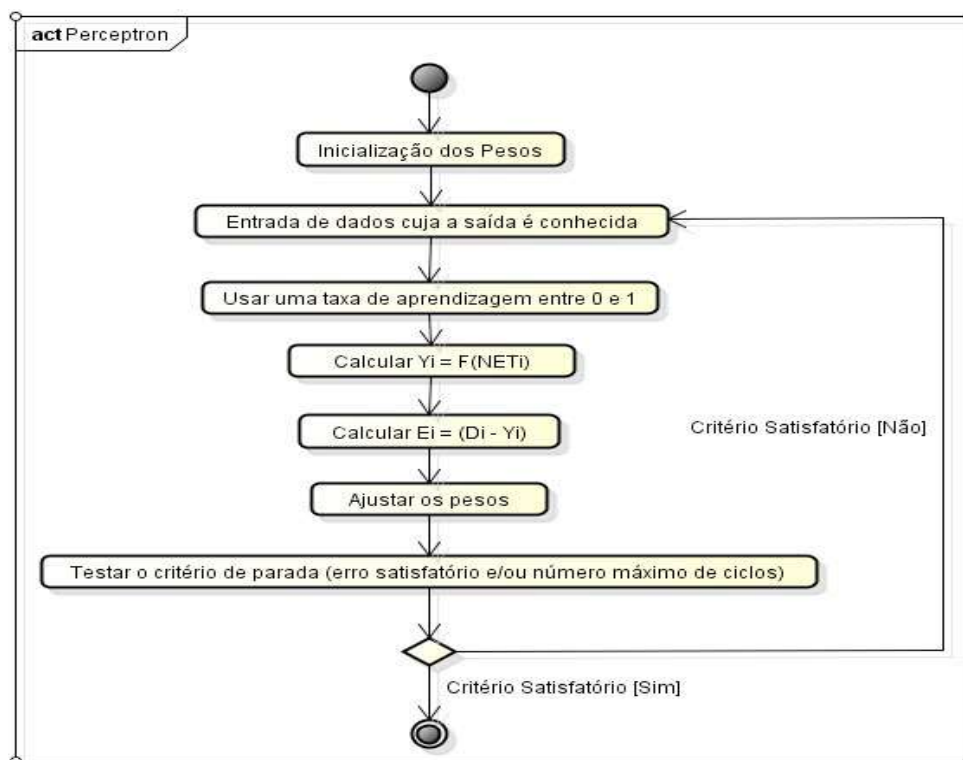
$$\Delta w_{ij} = w_{ij}(\text{novo}) - w_{ij}(\text{antigo})$$

$$w_{ij}(\text{novo}) = w_{ij}(\text{antigo}) + \alpha (d_i - y_i) x_j$$

Onde x_j é o sinal de entrada; d_i é o alvo; w_{ij} é o *peso sináptico*; $y_i = f(\text{net}_i)$ é a saída; α é a taxa de aprendizagem, um valor entre 0 e 1; e $(d_i - y_i)$ é o erro da saída, que é representado por e_i .

E o que é treinar a rede neural? É realizar o ajuste dos pesos por meio de uma regra de aprendizagem até que as saídas sejam adequadas para o problema que se quer solucionar.

Figura 7.5. Passos para Treinamento de uma Rede Perceptron



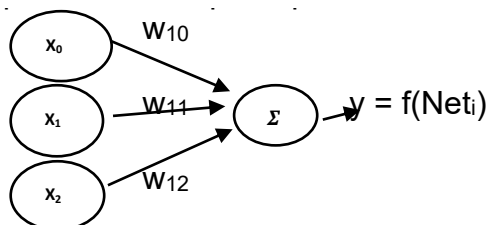
Entendido o conceito, vamos a alguns exemplos para sua fixação. Imagine dois sensores que serviram para indicar se um quarto de hotel está ocupado ou não. Quando há ocupantes no quarto, dois LEDs azuis ficam acessos (1), mas quando um dos dois LEDs ou os dois ficam apagados (0), é sinal de que *não há ocupantes no quarto*.

Tabela 7.1. Tabela de Ocupação dos Quartos

Sensor1 X1	Sensor2 X2	Saída D	Situação Desejada
0	0	0	Desocupado
0	1	0	Desocupado
1	0	0	Desocupado
1	1	1	Ocupado

Para solução desse problema, segue a arquitetura adequada, pois existem 2 sensores como entrada, devendo-se ter uma entrada de viés, que é uma constante e igual a 1 ($x_0 = +1$).

Figura 7.6. Arquitetura Perceptron para Problema de Ocupação



Com base na tabela acima e aplicando a regra de aprendizagem e gerando como pesos iniciais os valores 0 para w_{10} , 3 para w_{11} , 3 para w_{12} , temos a seguinte resolução: $net_1 = w_{10}x_0 + w_{11}x_1 + w_{12}x_2 = (0).(1) + (3).(0) + (3).(0) = 0$ $f(net_1) = 1$, mas para essas entradas, o valor esperado era 0, então os pesos precisam ser ajustados, assim:

$$w_{10}(\text{nov}) = w_{10}(\text{antigo}) + \alpha.(d-y).x_0 = (0) + 1.(0-1).1 = -1$$

$$w_{11}(\text{nov}) = w_{11}(\text{antigo}) + \alpha.(d-y).x_1 = (3) + 1.(0-1).0 = 3$$

$$w_{12}(\text{nov}) = w_{12}(\text{antigo}) + \alpha.(d-y).x_2 = (3) + 1.(0-1).0 = 3$$

Aplicando os novos pesos, temos: $net_1 = w_{10}x_0 + w_{11}x_1 + w_{12}x_2 = (-1).(1) + (3).(0) + (3).(0) = 0$ $f(net_1) = 0$, sendo esse o valor desejado. E quando todos os dados de exemplo são apresentados, um ciclo de aprendizagem foi completado, passando-se para o segundo ciclo de treinamento. Isso deve se repetir até que a rede convirja para os dados desejados; quando isso acontecer, a rede estará treinada.

Para finalizar o entendimento de um Perceptron, seguem os sete passos para resolvê-lo, com base no exemplo abaixo:

Tabela 7.2. Tabela para Exemplo de Perceptron

NOME	COMPOSITOR	CIENTISTA
MOZART	X	
BACH	X	
EINSTEIN		X
TESLA		X

1º. CRIAR A PERCEPÇÃO

= COMPOSITOR

= CIENTISTA

o & o = MOZART

& 1 = BACH

& o = EINSTEIN

1 & 1 = TESLA

2º. APLICAR A PERCEPÇÃO ÀS ENTRADAS E SAÍDAS

NOME	VIÉS (X _o)	ENTRADA ₁ (X ₁)	ENTRADA ₂ (X ₂)	DESEJADO (D)
MOZART	1	0	0	0
BACH	1	0	1	0
EINSTEIN	1	1	0	1
TESLA	1	1	1	1

3º. GERAR A TAXA DE APRENDIZAGEM(A); VIÉS(B); PESOS(W):

A=1; V=1; W_o=0; W₁=0; W₂=0; W₃=0; W₄=0

4º. DEFINIR A FUNÇÃO DEGRAU (FUNÇÃO DE ATIVAÇÃO LIMAR):

IF (Y <= 0) » (0) ELSE (1) (COM NÚMEROS ENTRE 0,1)

5º. OBTER A ENTRADA LÍQUIDA:

$$Y = (W_o * X_o) + (W_N * X_N)$$

$$Y_1 = (0 * 1) + (0 * 0) + (0 * 0) = 0 \text{ (Y <= 0) » (Y = D) } \tilde{N} \text{ AJUSTA PESOS}$$

$$Y_2 = (0 * 1) + (0 * 0) + (0 * 1) = 0 \text{ (Y <= 0) » (Y = D) } \tilde{N} \text{ AJUSTA PESOS}$$

$$Y_3 = (0 * 1) + (0 * 1) + (0 * 0) = 0 \text{ (Y <= 0) » (Y < > D) AJUSTA PESOS}$$

$$Y_4 = (0 * 1) + (0 * 1) + (0 * 1) = 0 \text{ (Y <= 0) » (Y < > D) AJUSTA PESOS}$$

6º. OBTER ERRO

$$E = D - Y$$

$$E_1 = 0 - 0 = 0$$

$$E_2 = 0 - 0 = 0$$

$$E_3 = 1 - 0 = 1$$

$$E_4 = 1 - 0 = 1$$

7º. REGRA DE AJUSTE DE PESOS DO PERCEPTRON

$$W(\text{NOVO}) = W(\text{ANTIGO}) + A * E * X$$

$$W_{30} = 0 + 1 + 1 + 1 = 3$$

$$W_{31} = 0 + 1 + 1 + 1 = 3$$

$$W_{32} = 0 + 1 + 1 + 0 = 2$$

6. Adaline

Em sequência à criação do Perceptron, surge a rede ADALINE, desenvolvida por Widrow e Hoff, cuja diferença entre a rede anterior está na função de ativação, visto que para os Perceptrons era usada a função degrau e para a ADALINE, a sigmoide ou tangente hiperbólica, sendo mais aconselhável o uso desta última. Outra grande diferença é que o Perceptron só serve para saídas discretas. Se a saída for contínua, deve-se utilizar ADALINE, isto é, para variáveis quantitativas, quando são numericamente mensuráveis, ou seja, seus possíveis valores são numéricos ou resultantes de contagem, sendo chamadas de discretas, como: número de filhos, alunos numa escola, etc.; mas quando os valores são expressos como intervalo ou união de números reais, são denominados de contínuos, como: peso, massa, altura, pressão sistólica, nível de açúcar no sangue, etc.

E para entender, vamos calcular a saída para essa rede, considerando $x_1 = -1$ e $x_2 = 1$. Considere que o viés sempre existirá e será $x_0 = 1$, utilizando como pesos os valores de $w_{10} = -1,0$ (o viés), $w_{11} = 3,0$ e $w_{12} = 4,5$. Para calcular o $Net_1 = w_{01} \cdot x_0 + w_{11} \cdot x_1 + w_{12} \cdot x_2$, temos $Net_1 = (-1,0).(1) + (3,0).(-1) + (4,5).(1)$, então $Net_1 = 1-3+4,5 = (1,0) + (-3,0) + (4,5) = 0,5$. Se utilizarmos a função linear para obtermos o y , teremos $y = f(Net_1)$, assim $y = f(0,5)$ e como $0,5$ é maior ou igual a zero, o retorno é 1. Mas para a função sigmoide, temos a regra $y = f(Net_1) = 1 / (1 + e^{-Net_1})$, então $y = 1 / (1 + e^{-0,5}) = 0,62$. Por outro lado, para a função de ativação Tangente Hiperbólica, que é melhor aplicável na resolução de problemas não lineares, temos $y = f(Net_1) = (e^{Net_1} - e^{-Net_1}) / (e^{Net_1} + e^{-Net_1})$, ficando para esse exemplo $y = (e^{0,5} - e^{-0,5}) / (e^{0,5} + e^{-0,5})$, o que resulta em $y = 0,46$.

A regra de aprendizagem usada para o treinamento da rede ADALINE desenvolvida por Widrow e Hoff foi baseada no método denominado de mínimos quadrados, mais conhecida como regra delta, e ela se baseia na determinação de um conjunto de pesos ótimos para aplicação junto do método de gradiente descendente, fazendo uso do erro médio quadrático. Segue sua expressão: $\Delta w_{ij} = (d_i - y_i) \cdot x_j \cdot f'(Net_i)$, assim o ajuste a ser aplicado nos pesos é de $w_{ij}(\text{novo}) = w_{ij}(\text{antigo}) + (d_i - y_i) \cdot x_j \cdot f'(Net_i)$, lembrando que d é o desejado, o alvo, e y foi a saída obtida e que a subtração de ambos resulta na taxa de erro, o α a taxa de aprendizagem e $f'(Net_i)$, que é a derivada da função de saída ou função de ativação $f(Net_i)$.

Observação: Quando a rede ADALINE não possui a camada escondida ou a camada intermediária, o ajuste dos pesos é realizado pela expressão $w_{ij}(\text{novo}) = w_{ij}(\text{antigo}) + \alpha \cdot x_j (d_i - y_i) x_j y_i (1 - y_i)$.

Para fixarmos, segue um exemplo de previsão com ADALINE. A procuradoria tem um volume mensal de dívidas sistemáticas. A PGE precisa de um sistema que exiba a previsão de

dívidas para prognosticar com antecedência como irá proceder com a cobrança, escolhendo o método de cobrança, com o objetivo de aumentar a arrecadação. Para exemplificar, imagine que os dados das dívidas foram normalizados e ficaram com os valores disponíveis na tabela abaixo:

Tabela 8.1. Tabela com Dados Fictícios de Devedores

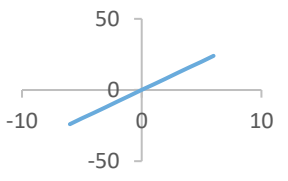
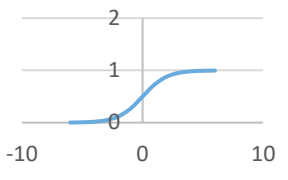
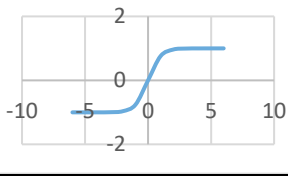
Exemplo	Dívida ₁	Dívida ₂	Dívida ₃	Pago
1	0,30	0,10	0,10	0,19
2	0,03	0,02	0,00	0,11
3	1,00	1,00	1,00	0,60
4	0,40	0,15	1,00	0,31
5	0,90	0,80	0,80	0,52
6	0,50	0,50	0,90	0,39

Com base no exemplo acima, vamos desenhar a arquitetura para esse problema: Teremos um viés, três variáveis referentes às dívidas dos três meses anteriores e o valor pago ou parcelado como saída.

Agora para resolvê-lo, devemos proceder da seguinte maneira: primeiro, iremos realizar 6 ciclos, pois temos, nesse caso, seis exemplos, e para cada ciclo, iremos fazer com os exemplos se apresentem em ordem aleatória. Para cada exemplo de cada ciclo, aplicamos a regra $w_0 * x_0 + w_1 * x_1 + w_2 * x_2 + w_3 * x_3$, visando obter o y . Comparamos o y obtido através da $f(\text{Net}_i) = \text{Net}_i$, como desejado. Em seguida, verifica-se se o y é igual ao d , isto é, se a saída encontrada, a saída líquida, é igual ao desejado. Se não for, recalculam-se os pesos através da expressão $\text{peso}(\text{novo}) = \text{peso}(\text{antigo}) + \text{taxa de aprendizagem} * (\text{desejável} - \text{saída líquida}) * \text{entrada}$. Após se encontrar o novo peso, deve-se calcular o erro quadrático, que é o $(\text{desejável} - \text{saída líquida})^2$ e guardar esse erro quadrático; repetir o processo para cada um dos seis exemplos e, no final, encontrar o Erro Médio Quadrático (EMQ), que é a média aritmética dos erros quadráticos encontrados. Após a execução dos seis ciclos, cada um com seis exemplos rodados aleatoriamente e para cada ciclo obtido o EMC, no final, pode-se fazer um gráfico do Erro Médio Quadrático e verificar se ele é pequeno para os resultados obtidos para a previsão e se não for, deve-se iniciar nova era, repetindo-se todo o processo até chegar a um gráfico de EMQ pequeno.

Lembrando que se a saída da rede ADALINE for linear, diz-se discreta; usa-se $f(\text{Net}_i) = \text{Net}_i$, mas se a saída for contínua, isto é, não linear, usa-se a função $f(\text{Net}) = 1 / (1 + e^{-\text{Net}})$. Segue tabela para relembrar:

Tabela 8.2. Gráficos das Funções de Ativação Linear e Não Linear

Nome	Plote	Equação	Derivada	Range
Linear (Identidade)		$f(x)=x$	$f'(x)=x$	$(-\infty, \infty)$
Sigmoide (Logística)		$f(x)=1/(1+e^{-x})$	$f'(x)=f(x)(1-f(x))$	$(0,1)$
Tangente Hiperbólica		$f(x)=(e^x-e^{-x})/(e^x+e^{-x})$	$f'(x)=1-f(x)^2$	$(-1,1)$

Os exemplos de redes neurais acima citados são básicos, mas necessários para poder compreender as atuais redes neurais, que são muito mais complexas, porém seguem o princípio acima explicado. Sem esse entendimento básico, fica muito difícil tentar compreender as atuais redes neurais.

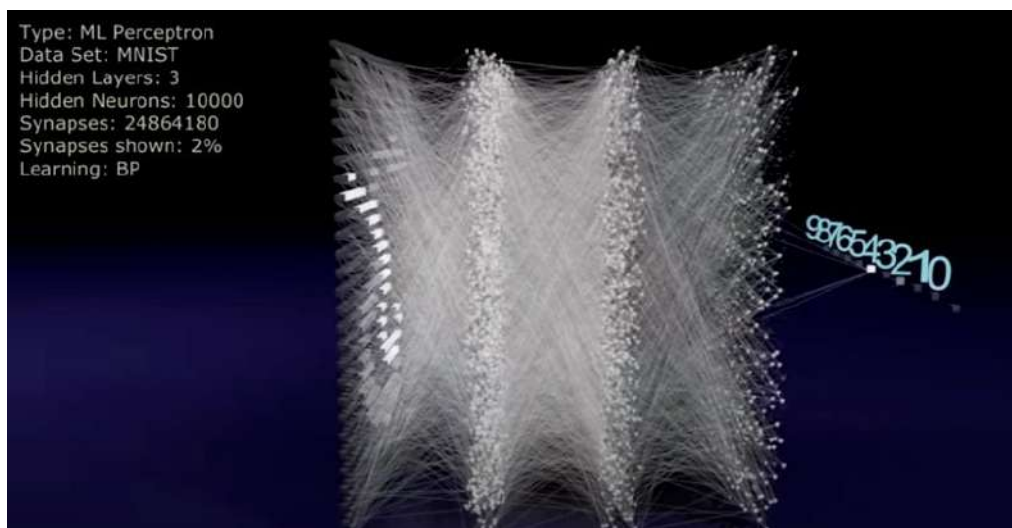
É importante salientar que a Inteligência Artificial não possui apenas esse tipo de abordagem, existindo também as de computação natural, que trazem heurísticas, como os algoritmos genéticos.

O próximo capítulo abordará, de forma básica, as redes neurais artificiais mais usadas atualmente e os fundamentos das metas-heurísticas para que se possa conhecer um pouco mais de outras abordagens sobre o IA.

7. MultiLayer Perceptron (MLP)

Uma MLP é uma rede neural semelhante às, anteriormente, apresentadas, a perceptron e a adaline, porém com uma diferença que possibilitou toda a evolução da abordagem conexionista, o incremento de mais de uma camada de neurônios.

Figura 9.1. MultiLayer Perceptron



Fonte: <https://youtu.be/3JQ3hYko51Y>

As redes MLP são compostas de camadas de neurônios ligadas entre si por sinapses com pesos. O resultado desse incremento foi que o aprendizado é, normalmente, realizado por uma técnica denominada de retropropagação do erro. Assim, as conexões de entrada são ponderadas por um peso sináptico, e cada sinal de entrada, o x , é multiplicado pelo seu respectivo peso sináptico, o w , e posteriormente é conectado ao neurônio.

Nesse tipo de rede, é utilizado um combinador linear, que corresponde à soma dos valores de entrada, multiplicado pelos pesos sinápticos, pela fórmula $(x_1 * w_1 + x_2 * w_2 + x_n * w_n)$, resultando no potencial de ativação, e uma função de ativação avalia esse potencial de ativação.

Dessa forma, o Perceptron multicamadas consiste em três ou mais camadas, sendo uma camada de entrada e uma camada de saída, com uma ou mais camadas intermediárias, também chamadas de camadas ocultas.

Esse tipo de rede é camada de rede totalmente conectada, pois cada neurônio de uma camada está conectado, com um determinado peso, a um neurônio da camada seguinte.

A característica principal dessa rede é o algoritmo de retropropagação, backpropagation, pois ele retropropaga o erro, sendo utilizado no treinamento da rede em dois passos, um de processamento direto e outro de processamento reverso.

No processamento direto, uma entrada é aplicada à rede neural, e seu efeito é propagado pela rede, camada a camada. Durante o processamento direto, os pesos da rede permanecem fixos. No processamento reverso, um sinal de erro calculado na saída da rede é propagado no sentido reverso, camada a camada, e ao final desse processo, os pesos são ajustados de acordo com uma regra de correção de erro.

É dessa forma que a rede aprende, pois se faz uso de um processo iterativo de ajuste de seus pesos sinápticos, o que é tecnicamente chamado de aprendizado supervisionado.

Para melhor entender esse processo, seguem os onze passos realizados por uma Multilayer Perceptron.

1º. OBTER:

TAXA (ALFA); VIÉS (X_o); ENTRADAS (X_N); DESEJADO (D_N); PESOS (W_N)

2º. CALCULAR ENTRADAS LÍQUIDAS INTERMEDIÁRIAS:

$$Y^I = (W_o * X_o) + (W_N * X_o)$$

3º. APLICAR FUNÇÃO DE ATIVAÇÃO:

$$NET^I = 1 / (1 + E(-Y))$$

4º. CALCULAR ENTRADAS LÍQUIDAS SAÍDAS:

$$Y^O = (W_o * NET) + (W_N * NET^I)$$

5º. FUNÇÃO DE ATIVAÇÃO DA SAÍDA (generalização da regra delta):

$$NET^O = 1 / (1 + E(-NET^I)) \text{ (através do gradiente descendente)}$$

6º. CALCULAR ERRO:

$$E = D - Y$$

7º. CALCULAR SENSIBILIDADE DE SAÍDA:

$$S^O = E - \text{DERIVADA}$$

8º. CALCULAR SENSIBILIDADE INTERMEDIÁRIA:

$$S^I = \text{DERIVADA} * W_N * S^O$$

9º. AJUSTAR PESOS DE SAÍDA:

$$W(\text{NOVO}) = W(\text{ANTIGO}) + \text{ALFA} * S^O * W_N$$

10º. AJUSTAR PESOS INTERMEDIÁRIOS:

$$W(\text{NOVO}) = W(\text{ANTIGO}) + \text{ALFA} * S^I * W_N$$

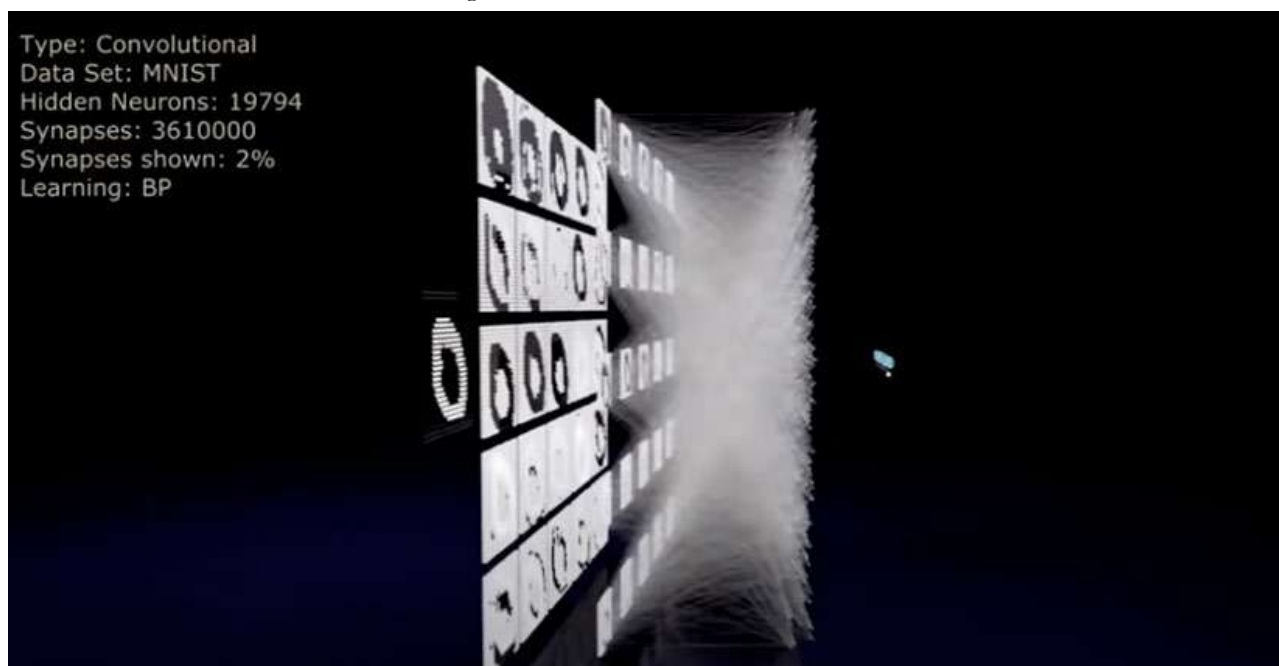
11º. FUNÇÃO CUSTO:

$$EMQ = (D - Y)^2$$

8. Convolutional Neural Network (CNN)

As redes convolucionais ou CNNs são amplamente utilizadas no que é denominado de aprendizado profundo; uma rede neural convolucional, ou CNN ou ConvNet, é uma classe de redes neurais profundas, sendo mais amplamente utilizadas no processamento de imagens. Mas, além do seu uso em reconhecimento de imagem e vídeo, são também utilizadas em sistemas de recomendação, classificação de imagem, processamento de linguagem natural e análise de séries temporais.

Figura 10.1. Convolutional Neural Network



Fonte: <https://youtu.be/3JQ3hYko51Y>

CNNs é a evolução das MLPs, mas que adotam uma abordagem diferente em relação à regularização, pois tiram vantagem do padrão hierárquico dos dados e montam padrões mais complexos, usando padrões menores e mais simples, portanto nas escalas de conexão e complexidade.

As redes convolucionais foram inspiradas por processos biológicos, com base nos estudos dos perceptrons, buscando imitar a organização do córtex visual dos animais, e responder a estímulos apenas em uma região restrita do campo visual, conhecida como campo receptivo.

As CNNs usam relativamente pouco pré-processamento em comparação com outros algoritmos de classificação de imagem, o que é uma grande evolução da técnica, pois a

rede aprende os filtros. Por outro lado, nas técnicas anteriores, eram necessários muitos hiperparâmetros, filtros, codificados pelos programadores, tornando as redes convolucionais independentes de um conhecimento prévio dos dados, diminuindo assim o esforço de programação.

Para melhor entender esse processo de uma rede convolucional, seguem os onze passos.

1º OBTER OS FILTROS:

MATRIZ(MAT); PAD(P); WIDTH(W₁); HIGHT(H₁); STRIDE(S); KERNEL(F)

2º CALCULAR W₂ E H₂:

$$W_2 = ((W_1 + 2 * P - F) / S) + 1$$

$$H_2 = ((H_1 + 2 * P - F) / S) + 1$$

3º GERAR MATRIZ RESULTANTE:

$$MATR = W_2 \times H_2$$

4º GERAR FEED COM OS DADOS DA PASSADA:

$$F = N \times N$$

5º OBTER A QUANTIDADE DE CÉLULAS DE F:

$$CEL = N \times N$$

6º CONVOLUCIONAR A QUANTIDADE DE VEZES DE W₂/H₂:

$$L_N C_N = ((MAT L_1 C_1 * F L_1 C_1) + (MAT L_N C_N * F L_N C_N)) / CEL$$

7º PREENCHER A MATRIZ RESULTANTE COM CADA CONVOLUÇÃO:

$$MATR = L_N C_N$$

8º APLICAR A FUNÇÃO DE ATIVAÇÃO RELU:

MATR = CEL NEGATIVAS RECEBEM ZERO

9º APLICAR CAMADA DE POOLING SOBRE A MATR:

A camada de pooling consiste na aplicação de um filtro 2x2 ou 3x3 (mais comum), que serve para simplificar a informação da matriz que chega da camada anterior, segundo um padrão (valor médio de uma região, valor máximo de uma região, valor mínimo de uma região, etc). Se a saída da camada anterior for 24x24, a saída do pooling será 12x12.

10º. REALIZAR O FLATTING:

FLAT = VETOR COM OS VALORES POSITIVOS

11º. APLICAR A MLP:

MLP = CADA ELEMENTO DO VETOR SERÁ UMA ENTRADA NA MLP

Para melhor visualizar o processo acima explicado, segue o exemplo da convolução da letra X:

Filtros

Pad **0**
 Width1 **9**
 Height1 **9**
 Stride **1**
 Feed **3 x3**

	1	2	3	4	5	6	7	8	9
1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	1	-1	-1	-1	-1	-1	1	-1
3	-1	-1	1	-1	-1	-1	1	-1	-1
4	-1	-1	-1	1	-1	1	-1	-1	-1
5	-1	-1	-1	-1	1	-1	-1	-1	-1
6	-1	-1	-1	1	-1	1	-1	-1	-1
7	-1	-1	1	-1	-1	-1	1	-1	-1
8	-1	1	-1	-1	-1	-1	-1	1	-1
9	-1	-1	-1	-1	-1	-1	-1	-1	-1

Feed 3X3

	1	2	3
1	1	-1	-1
2	-1	1	-1
3	-1	-1	1

Convolução

L1C1	-1	-1	-1
	-1	1	-1
	-1	-1	1

0.778

$$((-1)*(1)+(-1)*(-1)+(-1)*(-1)+(-1)*(-1)+(1)*(1)+(-1)*(-1)+(-1)*(-1)+(-1)*(-1)+(1)*(1))/9$$

L1C2	-1	-1	-1
	1	-1	-1
	-1	1	-1

-0.11

$$((-1)*(1)+(-1)*(-1)+(-1)*(-1)+(1)*(-1)+(-1)*(1)+(-1)*(-1)+(-1)*(-1)+(1)*(-1)+(-1)*(1))/9$$

L1C3	-1	-1	-1
	-1	-1	-1
	1	-1	-1

0.111

$$((-1)*(1)+(-1)*(-1)+(-1)*(-1)+(-1)*(-1)+(-1)*(1)+(-1)*(-1)+(1)*(-1)+(-1)*(-1)+(-1)*(1))/9$$

L1C4	-1	-1	-1
	-1	-1	-1
	-1	-1	-1
0.333			$((-1)^*(1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(1))/9$
L1C5	-1	-1	-1
	-1	-1	-1
	-1	-1	1
0.556			$((-1)^*(1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(1))/9$
L1C6	-1	-1	-1
	-1	-1	1
	-1	1	-1
-0.11			$((-1)^*(1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(1)+(1)^*(-1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(1))/9$
L1C7	-1	-1	-1
	-1	1	-1
	1	-1	-1
0.333			$((-1)^*(1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(-1)+(1)^*(1)+(-1)^*(-1)+(1)^*(-1)+(-1)^*(-1)+(-1)^*(1))/9$
L2C1	-1	1	-1
	-1	-1	1
	-1	-1	-1
-0.11			$((-1)^*(1)+(1)^*(-1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(1)+(1)^*(-1)+(-1)^*(-1)+(-1)^*(-1)+(-1)^*(1))/9$
L2C2	1	-1	-1
	-1	1	-1
	-1	-1	1
1			
L2C3	-1	-1	-1
	1	-1	-1
	-1	1	-1
-0.11			
L2C4	-1	-1	-1
	-1	-1	-1
	1	-1	1
0.333			
L2C5	-1	-1	-1
	-1	-1	1
	-1	1	-1
-0.11			
L2C6	-1	-1	1
	-1	1	-1
	1	-1	-1
0.111			

L2C7	-1	1	-1
	1	-1	-1
	-1	-1	-1

-0.11

L3C1	-1	-1	1
	-1	-1	-1
	-1	-1	-1

0.111

L3C2	-1	1	-1
	-1	-1	1
	-1	-1	-1

-0.11

L3C3	1	-1	-1
	-1	1	-1
	-1	-1	1

1

L3C4	-1	-1	-1
	1	-1	1
	-1	1	-1

-0.33

L3C5	-1	-1	1
	-1	1	-1
	1	-1	-1

0.111

L3C6	-1	1	-1
	1	-1	-1
	-1	-1	-1

-0.11

L3C7	1	-1	-1
	-1	-1	-1
	-1	-1	-1

0.556

L4C1	-1	-1	-1
	-1	-1	-1
	-1	-1	-1

0.333

L4C2	-1	-1	1
	-1	-1	-1
	-1	-1	1

0.333

L4C3	-1	1	-1
	-1	-1	1
	-1	1	-1

-0.33

L4C4	1	-1	1
	-1	1	-1
	1	-1	1

0.556

L4C5	-1	1	-1
	1	-1	-1
	-1	1	-1

-0.33

L4C6	1	-1	-1
	-1	-1	-1
	1	-1	-1

0.333

L4C7	-1	-1	-1
	-1	-1	-1
	-1	-1	-1

0.333

L5C1	-1	-1	-1
	-1	-1	-1
	-1	-1	1

0.556

L5C2	-1	-1	-1
	-1	-1	1
	-1	1	-1

-0.11

L5C3	-1	-1	1
	-1	1	-1
	1	-1	-1

0.111

L5C4	-1	1	-1
	1	-1	1
	-1	-1	-1

-0.33

L5C5	1	-1	-1
	-1	1	-1
	-1	-1	1

1

L5C6	-1	-1	-1
	1	-1	-1
	-1	1	-1

-0.11

L5C7	-1	-1	-1
	-1	-1	-1
	1	-1	-1

0.111

L6C1	-1	-1	-1
	-1	-1	1
	-1	1	-1

-0.11

L6C2	-1	-1	1
	-1	1	-1
	1	-1	-1

0.111

L6C3	-1	1	-1
	1	-1	-1
	-1	-1	-1

-0.11

L6C4	1	-1	1
	-1	-1	-1
	-1	-1	-1

0.333

L6C5	-1	1	-1
	-1	-1	1
	-1	-1	-1

-0.11

L6C6	1	-1	-1
	-1	1	-1
	-1	-1	1

1

L6C7	-1	-1	-1
	1	-1	-1
	-1	1	-1

-0.11

L7C1	-1	-1	1
	-1	1	-1
	-1	-1	-1

0.333

L7C2	-1	1	-1
	1	-1	-1
	-1	-1	-1

-0.11

L7C3	1	-1	-1
	-1	-1	-1
	-1	-1	-1

0.556

L7C4	-1	-1	-1
	-1	-1	-1
	-1	-1	-1

0.333

L7C5	-1	-1	1
	-1	-1	-1
	-1	-1	-1

0.111

L7C6	-1	1	-1
	-1	-1	1
	-1	-1	-1

-0.11

L7C7	1	-1	-1
	-1	1	-1
	-1	-1	-1

0.778

Width2	7	$\frac{((W1 + 2 * P - F) / S) + 1}{1}$
Height2	7	$\frac{((H2 + 2 * P - F) / S) + 1}{1}$

	1	2	3	4	5	6	7
1	0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
2	-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
3	0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
4	0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
5	0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
6	-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
7	0.33	-0.11	0.56	0.33	0.11	-0.11	0.78

Função de Ativação ReLU (7x7)

	1	2	3	4	5	6	7
1	0.77	0	0.11	0.33	0.55	0	0.33
2	0	1.00	0	0.33	0	0.11	0
3	0.11	0	1.00	0	0.11	0	0.55
4	0.33	0.33	0	0.55	0	0.33	0.33
5	0.55	0	0.11	0	1.00	0	0.11
6	0	0.11	0	0.33	0	1.00	0
7	0.33	0	0.56	0.33	0.11	0	0.78

Max Pooling (3 x 3)

	1	2	3
1	1.00	0.55	0.55
2	0.55	1.00	0.55
3	0.56	0.33	1.00

Flatting

1	1
2	0.55
3	0.55
4	0.55
5	1
6	0.55
7	0.56
8	0.33
9	1

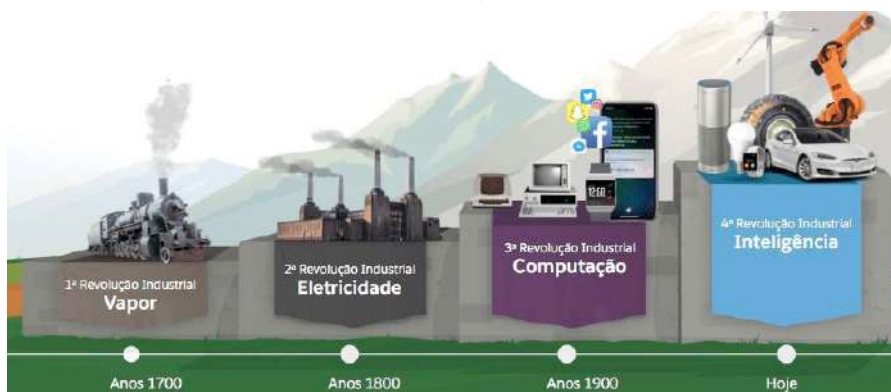
Conclusões

Inteligência Artificial (IA) e Inteligência Computacional (IC) não são denominações novas nem um campo de estudo recentemente criado, mas sua aplicabilidade está sendo, nos últimos anos, a força de inovação em vários aspectos. Entender o que é, os seus fundamentos básicos e suas aplicabilidades no mundo atual é obrigatório para qualquer profissional que queira se manter no mercado de trabalho, pois, em breve, para não dizer, agora mesmo, ninguém poderá dizer que IA não faz parte do seu dia a dia e não precisará dela.

Se voltarmos para os anos da década de 1980, algumas pessoas diziam que microcomputadores não eram ferramentas do seu cotidiano e que suas ocupações não necessitavam desse instrumento. Hoje, nenhuma pessoa integrada socialmente pode dizer que não faz uso de microcomputadores, seja no uso de desktops, laptops, notebooks ou mais amplamente utilizando celulares. O mesmo ocorrerá em breve com IA/IC.

Conhecer o que são essas tecnologias, o que significam os termos, quais as ferramentas disponíveis são fundamentalmente importantes para um gestor público, tomador de decisão, principalmente agora na quarta revolução industrial.

Figura 11.1. Revolução Industrial



Fonte: <https://www.salesforce.com/br/blog/2018/Janeiro/O-que-e-Quarta-Revolucao-Industrial.html>

O que este material aborda é um pequeno pedaço do mundo de heurísticas, algoritmos, técnicas, classes, ferramentas, plataformas etc., uma introdução a esse maravilhoso e espetacular mundo da inteligência artificial, que veio para apoiar e surpreender os humanos. Mas, para quem deseja se especializar na área, é necessário buscar muito mais subsídios, como leitura de artigos, livros, vídeos e aulas, sobre o assunto.

Aconselho buscar uma formação mais especializada, seja um curso técnico, uma especialização lato sensu ou mesmo stricto sensu para que se possa efetivamente entender essa disciplina, que é multidisciplinar, e poder contribuir, permitindo que o governo possa tornar suas atividades mais eficientes e eficazes, com o objetivo de alcançar sua efetividade.

Desejo sucesso aos que pretendem enveredar nessa área!

Abaixo faço a sugestão de leitura dos materiais utilizados como referência para a escrita deste pequeno material como também uma coletânea de materiais complementares.

Referências

- BOSTRON, Nick. **Superinteligência**. 1. ed. Editora Darkside Books. Rio de Janeiro, 2018.
- GOLDSCHMIDT, Ronaldo. **Data Mining**. 2. ed. Editora Elsevier. Rio de Janeiro, 2015.
- GRUS, Joel. **Data Science do Zero**. 1. ed. Editora Alta Books. Rio de Janeiro, 2016.
- LEE, Kai-Fu. **Inteligência Artificial**. 1. ed. Editora Globo Livros. Rio de Janeiro, 2019.
- LINDEN, Ricardo. **Algoritmos Genéticos**. 3. ed. Editora Ciência Moderna. Rio de Janeiro, 2012.
- RUSSEL, Stuard; NORVIG, Peter. **Inteligência Artificial**. 2. ed. Editora Campus. Rio de Janeiro, 2004.
- SCHWAB, Klaus. **A Quarta Revolução Industrial**. 1. ed. Editora Edipro. São Paulo, 2016.
- SILVA, Ivan Nunes da; et al. **Redes Neurais Artificiais**. 2. ed. Editora Artliber. São Paulo, 2016.
- VALENÇA, Mêuser. **Aplicando Redes Neurais**. 2. ed. Editora Livro Rápido. Olinda, 2005.
- VALENÇA, Mêuser. **Fundamentos das Redes Neurais**. 2. ed. Editora Livro Rápido. Olinda, 2010.

Material Complementar

<http://www.revistas.poli.br/index.php/repa/article/view/1299/578>

<https://colab.research.google.com/notebooks/welcome.ipynb>

<https://orange.biolab.si/>

<https://teachablemachine.withgoogle.com/>

<http://www.alvarofpinheiro.somee.com/nn.aspx>

<https://github.com/alvarofpinheiro/NeuralNetwork>

<https://cs231n.github.io/convolutional-networks/>

<https://ieeexplore.ieee.org/document/5206848>

<https://jupyter.org/>

<https://www.anaconda.com/products/individual>

<https://www.python.org/>

https://www.youtube.com/watch?v=eowW5H_Fdoo

<https://www.youtube.com/watch?v=loPPlsIhO3o&t=38os>

<https://cloud.google.com/automl?hl=pt-br>

Sobre o autor

Álvaro Farias Pinheiro é Analista em Gestão de Tecnologia da Informação e Comunicação (AGTIC) da Agência Estadual de Tecnologia da Informação e Comunicação do Estado de Pernambuco (ATI/PE), Coordenador de Sistemas da Procuradoria Geral do Estado de Pernambuco (PGE/PE). Kursou o Bacharelado em Ciência da Computação na Universidade Católica de Pernambuco (UNICAP) e é Bacharel em Sistemas de Informação com Ênfase em Engenharia de Software pela Faculdade Integrada do Recife (FIR), com Especialização em Metodologias de Desenvolvimento na Engenharia de Software pela União Brasileira de Tecnologia (UNIBRATEC) e Mestre em Engenharia de Software pelo Centro de Estudos e Sistemas Avançados do Recife (CESAR). Doutorando em Inteligência Computacional e Engenharia de Software pela Escola de Engenharia da Politécnica de Pernambuco da Universidade de Pernambuco (UPE). Para mais detalhes acesse <http://www.alvarofpinheiro.eti.br/sobre/>.

