Redes Neurais Artificiais: Arquitetura [1]

Prof. Dr. Giancarlo D. Salton

agosto de 2023

[1] Universidade Federal da Fronteira Sul
Campus Chapecó
gian@uffs.edu.br

As redes neurais artificiais (RNAs) são modelos de *machine learning* inspirados pelo funcionamento do cérebro humano. Propostas inicialmente como um modelo matemático do processamento do cérebro, as RNAs passaram por um longo período de desinteresse por parte da comunidade científica em razão de limitações iniciais (que veremos mais adiante). No entanto, desde as décadas de 1980 e 1990, com o surgimento de novas arquiteturas e técnicas de treinamento, assim como o aumento do processamento computacional, o interesse nas RNAs foi renovado.

Inspiração nos Neurônios Biológicos

Desde o início de seu desenvolvimento, a inspiração para as RNAs foi o cérebro humano. Apesar disso, como veremos adiante, as RNAs evoluíram para algo bem diferente das redes neurais biológicas a ponto de muitos pesquisadores afirmarem que devemos abandonar completamente a analogia com as redes biológicas. Ainda assim, muitos dos avanços atuais continuam sendo inspirados pelas descobertas sobre o funcionamento dos neurônios biológicos e as redes formadas por eles.

Como demonstra a Figura 1, um neurônio biológico é composto por algumas partes principais. De forma simplificada, os *dendritos* recebem sinais (pequenos pulsos elétricos) de outros neurônios que estão nas proximidades através da comunicação de neurotransmissores nos pontos conhecidos como sinapses. A *célula* realiza o "processamento" dos sinais recebidos e emite seu próprio pulso elétrico caso a "excitação" produzida atinja um certo potencial. Este pulso elétrico é transmitido pelo *axônio* até seus terminais onde os neurotransmissores irão comunicar o seu sinal para outros neurônios que estejam nas proximidades.

Embora os neurônios biológicos sejam muito mais complexos e funcionem de maneira muito mais complicada do que apresentado acima, a ideia geral nos permite criar um modelo matemático para representá-los computacionalmente[2]. Na Figura 2 temos uma comparação entre um neurônio biológico e um neurônio artificial.

Como podemos perceber, o funcionamento do neurônio artificial é bastante semelhante ao funcionamento (simplificado) do neurônio biológico. Da mesma forma que o neurônio biológico recebe sinais no formato de pulsos elétricos de outros neurônios através dos dendri-

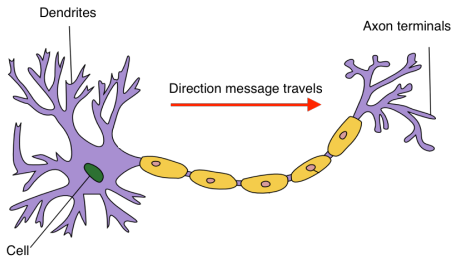
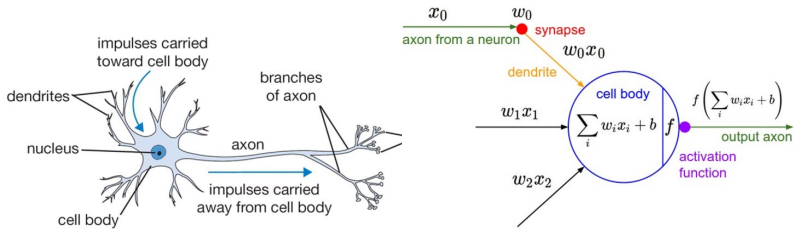


Figura 1: Representação simplificada de um neurônio biológico. Os dendritos servem para receber sinais de neurônios mas proximidades, enquanto a célula faz o "processamento" dos sinais e produz o seu próprio sinal. Este sinal produzido então trafega pelo axônio e é comunicado aos neurônios nas proximidades através dos terminais do axônio.

[2] Walter Pitts Warren S. McCulloch. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4): 115–133, 1943



(a) Neurônio biológico (b) Neurônio artificial

Figura 2: Comparação entre um neurônio biológico e um neurônio artificial. As *conexões* dos neurônios artificiais simulam as sinapses dos neurônios biológicos. A *transformação linear* dos valores de entrada simula o processamento da célula do neurônio biológico enquanto a *função de ativação* calcula a excitação do neurônio. A saída do neurônio artificial e envio para os demais neurônios representa a comunicação feita pelo axônio.

tos, o neurônio artificial recebe variáveis de entrada (representadas na Figura **??** por variáveis x_i) através de *conexões*, (representadas por parâmetros w_i que, na versão original do neurônio artificial, eram definidos manualmente). O processamento do neurônio ocorre através de uma transformação das variáveis de entrada (falaremos sobre essa transformação mais adiante) utilizando os valores das conexões. O neurônio artificial então emite a sua saída após o cálculo de sua ativação, semelhante ao neurônio biológico que também produz seu próprio pulso elétrico quando excitado. Esta ativação é comunicada aos demais neurônio artificiais através da passagem desta ativação como variável de entrada para outros neurônios artificiais conectados à ele, assim como o axônio conduz os sinais produzidos por um neurônio biológico para as sinapses próximas.

A primeira versão do neurônio artificial que possuía capacidade de *aprendizado* ficou conhecida como "*Perceptron*" [3] e seu esquema está representado graficamente na Figura 3. As variáveis de entrada x_i chegam através de conexões w_i, são processadas dentro do *Perceptron* e uma saída (*output*) é produzida. No caso do *Perceptron*, assim como nos neurônios artificiais originais, a saída possui o formato representado pela Equação 1:

$$output = \begin{cases} 0 & \text{if } b + \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } b + \sum_j w_j x_j > \text{threshold} \end{cases} \quad (1)$$

Observe que a saída do *Perceptron* define quando um neurônio artificial está ativo (if $b + \sum_j w_j x_j >$ threshold) ou inativo (if $b + \sum_j w_j x_j \leq$ threshold). Embora estas ideias [4] se aproximassem do comportamento conhecido à época dos neurônios biológicos e demonstrarem capacidade de modelagem de diversos problemas nos quais a saída fosse binária, os *Perceptrons* possuem muitas limitações. Isso ocorre devido ao fato de que a sua função de saída que acaba tornando-os uma função linear, conforme demonstrado na Figura 4.

Outro ponto importante a ser observado é a utilização do parâmetro b, chamado de b. Este parâmetro controla a inclinação da reta ao definir o ponto onde ela toca cruza o eixo y

[3] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958

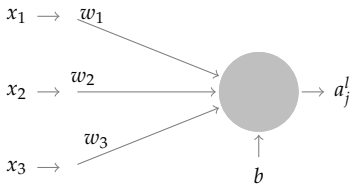


Figura 3: Representação gráfica de um *Perceptron*, onde as variáveis de entrada são representadas por $x_{1...3}$ e as conexões por $w_{1...3}$, além do bb.

[4] Avanços na neurobiologia já demonstraram que estas ideias estavam incorretas. O neurônio biológico nem sempre está 100% ativo ou 100% inativo, após seu "processamento" interno. Ele pode se encontrar em um meio-termo mesmo sem ter recebido sinais de outros neurônios.

Neurônios Artificiais

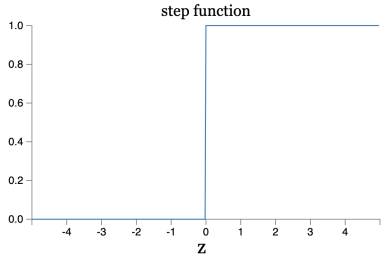


Figura 4: Representação da função de saída de um *Perceptron* conforme definida na Equação 1, formando uma *step function*.

Como observado na seção anterior, a função de saída do *Perceptron* forma uma *step function*, onde todos os valores negativos ou iguais a 0 (zero) da transformação linear são mapeados para a saída igual a *zero* enquanto os valores positivos são mapeados para a saída igual à 1 (um). Como demonstrado por [5] em 1969 (e na Figura 5), os *Perceptrons* com apenas uma camada não conseguem resolver problemas "simples" como a função XOR ("ou exclusivo"), o que provocou um "desânimo" nos pesquisadores da área[6].

Este desânimo na sequência de pesquisas com redes neurais ficou conhecido como "Inverno da IA". Durante este período, muitos outros sistemas inteligentes foram desenvolvidos utilizando métodos mais tradicionais. No entanto, estes sistemas demonstraram-se incapazes de escalar para problemas do mundo real em função da falta de poder computacional ou pelos métodos serem inadequados. Algumas vezes, os problemas eram resolvidos pela simples tentativa de várias soluções e eliminação daquelas consideradas inadequadas. Desta forma, muitos sistemas que funcionavam para pequenos problemas não conseguiam resolvê-los quando algumas regras a mais eram inclusas. No entanto, alguns pesquisadores se mantiveram firmes nas investigações referentes às redes neurais e alguns progressos foram feitos.

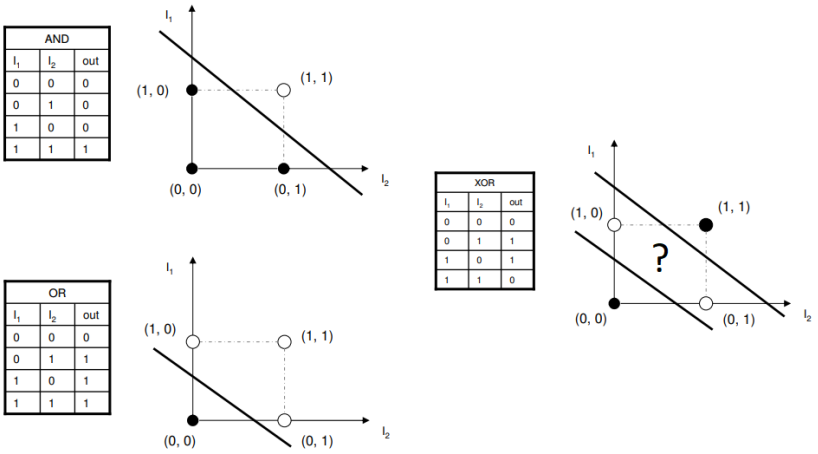


Figura 5: Demonstração da incapacidade dos *Perceptrons* em modelar a função XOR quando comparados com a modelagem das funções AND e OR.

Um dos progressos foi a introdução do neurônio artificial com uma função de ativação conforme definido na Equação 2 e Equação 3:

[5] Seymour A. Papert Marvin Minsky. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1 edition, 1969

[6] Por anos a comunidade científica interpretou erroneamente as demonstrações feitas por Minsky e Papert. Por muito tempo difundiu-se a ideia de que mesmo com mais camadas, uma rede de *Perceptrons* era incapaz de aprender funções como a XOR. No entanto, no próprio livro onde demonstram o problema da função XOR, os autores demonstram a capacidade que redes com várias camadas *Perceptrons* tem de modelar problemas ainda mais complexos que o XOR.

$$z = b + \sum_{j=1}^{N} w_j x_j \tag{2}$$

$$a = \frac{1}{1 + e^{-z}} \tag{3}$$

Perceba que na Equação 2 temos a mesma transformação linear do *Perceptron*. No entanto, uma das, senão a maior mudança, foi a introdução da função sigmoide[7] representada na Equação 3 [8]. A função sigmoide possui uma saída dentro do intervalo $[0, 1]$, ou seja, muito parecida com o *Perceptron*. No entanto, somente com valores de z próxi os de $-\inf$ é que temos a saída próxima de 0 e com valores de z próximos de $+\inf$ é que temos a saída próxima de 1, ou seja, a mudança teria que ser muito grande para inverter completamente a saída da função[9]. A Figura 6 demonstra o gráfico criado pela saída da função sigmoide.

A função sigmoide foi introduzida para corrigir o problema das mudanças drásticas que aconteciam, com frequência no *Perceptron*, quando pequenas mudanças eram feitas nos parâmetros w_j ou b do neurônio. Em outras palavras, por utilizar uma *step function*, uma pequena mudança num parâmetro podia fazer uma saída 0 do *Perceptron* se tornar 1 (*i.e.*, se a transformação linear que antes era igual a -0.1 e produzia a saída 0 for alterada para $+0.1$, a nova saída será 1). Esta propriedade do *Perceptron* não era desejável e portanto, a função sigmoide passou a ser utilizada para transformar a saída do neurônio artificial.

Uma das vantagens de se utilizar uma função como a sigmoide é exatamente a não-linearidade introduzida por ela na saída dos neurônios. Esta característica auxilia no fato de que é possível realizar transformações (ou perturbações) no espaço onde as observações (*i.e.*, os exemplos que são passados pela rede) estão inseridas e, assim, facilitar a busca pelo limiar que separa classes não-linearmente separáveis. Um exemplo desta "distorção" é demonstrado na Figura 7.

Além da sigmoide, outras três funções de ativação são amplamente utilizadas nas camadas nas redes neurais mais modernas, representadas na Equação 4 ($tanh$), Equação 5 ($relu$) e na Equação 6 ($softmax$)[10].

[7] Também chamada de função logística ou simplesmente σ.

[8] Outra forma de representar a função σ é $\frac{1}{\exp(-z)}$ ou $\frac{1}{\exp(-(b+\sum_{j=1}^{N} w_j x_j))}$

[9] Quando $z = 0$, a saída da função sigmoide é 0.5.

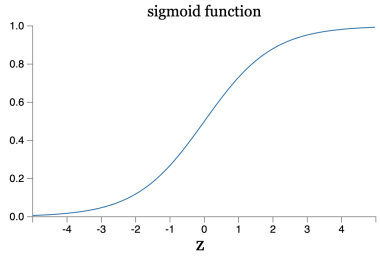


Figura 6: Representação da função sigmoide. Perceba o formato de "$\mathcal{S}$", que dá origem ao seu nome.

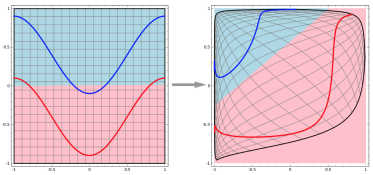


Figura 7: Ao incluirmos um transformação não-linear, alteramos o formato do espaço onde os exemplos estão inseridos.

[10] Retornaremos a elas mais adiante no nosso conteúdo.

$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{4}$$

$$ReLu(z) = \begin{cases} 0 & \text{if } z < 0 \\ z & \text{if } z \geq 0 \end{cases} \tag{5}$$

$$softmax(x)_j = \frac{e^{x_j}}{\sum_k e^{x_k}} \tag{6}$$

Arquitetura das Redes Neurais Artificiais

As rede neurais artificiais são organizadas em camadas de neurônios artificiais interconectados. A forma na qual os neurônios são conectados determina a arquitetura de rede neural e o formato no qual as observações são inseridas no modelo. Inicialmente apenas camadas de conexões simples foram consideradas e hoje elas são conhecidas como camadas *fully-connected*, "totalmente conectadas" ou "*densas*"[11]. Como veremos mais adiante, as redes neurais que possuem apenas camadas *densas* também são chamadas de *feedforward* para diferenciá-las das redes convolucionais e recorrentes.

[11] Futuramente iremos estudar camadas *convolucionais*, *max* ou *average-pooling*, *recorrentes*, entre outras. Muitas vezes a existência de camadas convolucionais ou recorrentes juntamente com as densas acaba nomeando a arquitetura da rede neural (*e.g.*, rede convolucional ou rede recorrente).

As redes neurais artificiais possuem pelo menos duas camadas representando a entrada e a saída do modelo. A camada de entrada é uma camada especial pois cada neurônio nesta camada receber apenas uma entrada, correspondente a uma *feature* específica referente ao problema para o qual estamos treinando o modelo. Portanto, determinamos a quantidade de neurônios nesta camada conforme o número de *features* que estamos utilizando para representar cada problema do domínio. Além disso, os neurônios da camada de entrada não realizam a transformação linear ao receber a sua *feature* e sua ativação é a função identidade $f(x) = x$. A função desta camada é distribuir a sua ativação para a camada seguinte através das conexões dos neurônios artificiais, conforme veremos mais adiante.

A camada de saída é responsável pela predição que o modelo irá realizar. Quando trabalhamos com tarefas de regressão, utilizamos apenas 1 (um) único neurônio nesta camada e que, na maioria das vezes, realiza apenas a transformação linear e utiliza a função identidade como ativação, transformando-os em neurônios lineares. Observe que nada impede de utilizarmos as funções de ativação σ ou $\tanh$, desde que o intervalo da predição a ser realizada coincida com os intervalos de saída destas funções. No entanto, ainda assim é mais comum

vermos neurônios lineares na camada de saída de uma rede neural em tarefa de regressão. Caso estejamos realizando uma tarefa de classificação binária, a camada de saída pode conter 1 (um) ou 2 (dois) neurônios. O que determina a quantidade de neurônios é a função de ativação utilizada nesta camada pois, se utilizamos a função $softmax$ (Equação 6) teremos obrigatoriamente 2 (dois) neurônios para formar a distribuição de probabilidade requerida. Por outro lado, caso utilizemos a função de ativação (σ) determinarmos um *threshold*, geralmente igual a 0.5, que será utilizado para determinar qual as duas classes será predita pelo neurônio. Esta operação de *threshold* é feita de forma semelhante ao *Perceptron* mas utilizando a função de ativação antes da aplicação do *threshold*:

$$\begin{cases} 0 & \text{if } \sigma(z) < threshold \\ 1 & \text{if } \sigma(z) \geq threshold \end{cases} \tag{7}$$

Note que ao definirmos o *threshold*, estamos simulando uma distribuição de probabilidade entre as duas classes, pois de certo modo e a probabilidade da classe predita ser 1 é igual a $\sigma(z)$, então a probabilidade da classe predita ser 0 é igual a $1 - \sigma(z)$. Note que a uma rede neural que possui apenas uma camada de entrada e uma camada de saída com ativações σ em uma tarefa de classificação, pode ser comparada a uma regressão logística *one-vs-all*[12].

Entre as camadas de uma rede neural existem as conexões entre neurônios artificiais. Estas conexões são representadas pelos parâmetros w_j que participam da transformação linear na entrada dos neurônios. A saída de cada neurônio em uma camada serve de entrada para a camada seguinte e a conexão é feita através destes parâmetros w. Desta forma, quando uma rede possuí camadas do tipo *densas*, temos uma representação esquemática conforme demonstrado na Figura 8.

Os neurônios marcados como $x_1 \dots x_4$ representam a camada de entrada (*input layer*) enquanto os marcados como $a_1 \dots a_3$ representam a camada de saída (*output layer*). Perceba que todos os neurônios da *input layer* se conectam a todos os neurônios da *output layer* e essas conexões estão representadas nos parâmetros $w_1 \dots w_4$ na matriz da Figura 9. Além disso, podemos observar que cada neurônio na *output layer* possuí 4 (quatro) conexões de entrada, que estão representadas nas linhas da matrix de parâmetros $\mathbf{W}$. Desta forma, podemos dizer que as linhas da matrix $\mathbf{W}$ representam as conexões de entrada específicas de um neurônio em uma camada l (neste caso a *output layer*) enquanto cada coluna representa as conexões de saída de um neurônio na camada $l - 1$ (representada aqui pela *input layer*). Assim a matriz $\mathbf{W}$ sempre terá o número de linhas igual ao número de neurônios na

[12] Uma regressão logística comum é capas de distinguir entre duas classes apenas. Para incluir mais classes nesta classificação, precisamos incluir uma regressão logística que aprende a distinguir uma classe específica das demais. Em outras palavras, aquela regressão logística é treinada para dizer se uma observação pertence à uma classe ou não. Como incluímos uma regressão para cada classe do problema, dizemos que cada uma delas compete contra as demais para identificar a classe para a qual foi treinada. Por isso o nome *one-vs-all*.

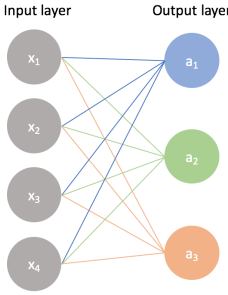


Figura 8: Representação esquemática de uma rede neural com camadas de entrada (*input layer*) representada pelos neurônios marcados como x_i e saída (*output layer*) representada pelos neurônios marcados como w_i.

camada l e terá o número de colunas igual ao número de neurônios na camada $l-1$.

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow[activation]{} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Figura 9: Representação esquemática de uma rede neural com camadas de entrada (*input layer*) representada pelos neurônios marcados como x_i e saída (*output layer*) representada pelos neurônios marcados como w_i.

Na prática, as matrizes de parâmetros w possuem a mesma orientação da camada, isto é, os parâmetros representando o número de conexões entrando em um neurônio da camada l'' são representados na coluna da matriz $\mathbf{W}$ da mesma forma como as entradas x estão representadas na Figura 8, seguindo a orientação da disposição dos neurônios. No entanto, é costume transpor a matriz $\mathbf{W}$, conforme Figura 10, para que fique na orientação da Figura 8 para facilitar o cálculo da transformação linear dos neurônios. Isto se dá pelo fato de que podemos calcular esta transformação utilizando o produto escalar entre a matriz $\mathbf{W}$ e o vetor de *features* ($\mathbf{x}$) conforme demonstrado na Figura 8. Formalmente, esta transformação está demonstrada, já em formato de matrizes, na Equação 8:

$$w_1 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}, w_2 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}, w_3 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \longrightarrow \begin{bmatrix} \leftarrow w_1^{\mathrm{T}} \rightarrow \\ \leftarrow w_2^{\mathrm{T}} \rightarrow \\ \leftarrow w_3^{\mathrm{T}} \rightarrow \end{bmatrix}$$

Figura 10: Representação esquemática de uma rede neural com camadas de entrada (*input layer*) e saída (*output layer*).

$$\mathbf{z} = \mathbf{b} + \mathbf{W}^T\mathbf{x} \tag{8}$$

Esse formato de multiplicação de matrizes nos permite ainda processar múltiplas observações de uma vez só, o que acelera ainda mais o processamento da rede neural caso utilizemos uma biblioteca de software especializada em álgebra linear. Para isso, precisamos fazer uma modificação apenas no vetor com a observação a ser processada e torná-lo uma matriz contendo um observação por coluna conforme a Figura 11:

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} \overset{\text{Observation 1}}{x_1} & \overset{\text{Observation 2}}{x_1} & \overset{\text{Observation 3}}{x_1} & \overset{\text{Observation 4}}{x_1} \\ x_2 & x_2 & x_2 & x_2 \\ x_3 & x_3 & x_3 & x_3 \\ x_4 & x_4 & x_4 & x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} \xrightarrow[activation]{} \begin{bmatrix} \overset{\text{Observation 1}}{a_1} & \overset{\text{Observation 2}}{a_1} & \overset{\text{Observation 3}}{a_1} & \overset{\text{Observation 4}}{a_1} \\ a_2 & a_2 & a_2 & a_2 \\ a_3 & a_3 & a_3 & a_3 \end{bmatrix}$$

Figura 11: Representação esquemática de uma rede neural com camadas de entrada (*input layer*) representada pelos neurônios marcados como x_i e saída (*output layer*) representada pelos neurônios marcados como w_i.

Como agora estamos trabalhando com duas matrizes, atualizamos a equação da transformação linear conforme a Equação 9:

$$\mathbf{z} = \mathbf{b} + \mathbf{W}^T\mathbf{X} \tag{9}$$

Aproveitando o fato de que realizamos os cálculos através de multiplicação de matrizes, é fácil de incluirmos camadas intermediárias entre a camada de entrada e de saída. Estas camadas são chamadas de camadas escondidas (*hidden layers*) e atualizamos a arquitetura da rede conforme a Figura 12.

Além da arquitetura, devemos adicionar matrizes de parâmetros **W** e **b** de acordo. Esta adição de mais camadas na rede neural que faz com que estes modelos consigam aprender representações distribuídas dos dados de entrada e assim realizar o aprendizado profundo (*Deep Learning*)[13]. O cálculo completo de uma rede neural com 1 (um) *hidden layer* está representada na Equação 10.

$$\begin{aligned} h^{(1)} &= g\left(b^{(1)} + W^{T(1)}x\right) \\ o &= g\left(b^{(2)} + W^{T(2)}h^{(1)}\right) \end{aligned} \tag{10}$$

Nesta equação, temos a camada de entrada representada pela entrada x[14], enquanto as conexões entre *input layer* e *hidden layer* são representadas por $W^{T(1)}$ e $b^{(1)}$. Os parâmetros $W^{T(2)}$ e $b^{(2)}$ representam as conexões entre *hidden layer* e *output layer* e $h^{(1)}$ são as ativações produzidas pelos neurônios em *hidden layer* e que servem de entrada para *output layer*.

Algumas regras para as camadas

As *input layer* possuem 1 (um) neurônio para cada *feature* de entrada. Os neurônios desta camada possuem como ativação a *função identidade*: $a_i = x_i$. Nas *output layer*. caso alvo seja de *regressão* (e em algumas classificações binárias) possui apenas 1 (um) neurônio. Caso contrário haverá um neurônio para cada *target* identificado no problema[15]. A ativação da camada de saída depende do problema: (a) se regressão, pode ser a função identidade; (b) se for classificação binária pode ser σ; ou $softmax$; (c) se classificação com mais de duas classes, usa-se a ativação $softmax$.

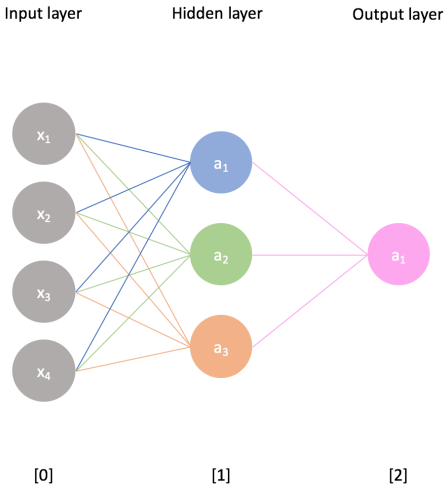


Figura 12: Representação esquemática de uma rede neural com camadas de entrada (*input layer*), saída (*output layer*) e uma camada escondida (*hidden layer*).

[13] Iremos estudar o que são as representações distribuídas mais à frente durante o curso.

[14] Visto que a camada de entrada apenas aplica a função identidade.

[15] Por exemplo, predizer polaridade do sentimento de um *review* de produto, requer 3 (três) neurônios: POSITIVO, NEGATIVO e NEUTRO

Referências

Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016.

John D. Kelleher and Brendan Tierney. *Deep Learning*. MIT Press, 1 edition, 2018.

Seymour A. Papert Marvin Minsky. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1 edition, 1969.

Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL `http://neuralnetworksanddeeplearning.com/`.

Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 3 edition, 2016.

Walter Pitts Warren S. McCulloch. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.