

Redes Neurais (aprendizado profundo)



Nas aulas anteriores, implementamos os seguintes algoritmos de Aprendizado de Máquina:

- Regressão Linear (para problemas de regressão)
- Regressão Logística (para problemas de classificação)

OBS: Com isso, finalizamos a **Parte 1** da disciplina.

Nas aulas anteriores, implementamos os seguintes algoritmos de Aprendizado de Máquina:

- Regressão Linear (para problemas de regressão)
- Regressão Logística (para problemas de classificação)

OBS: Com isso, finalizamos a **Parte 1** da disciplina.

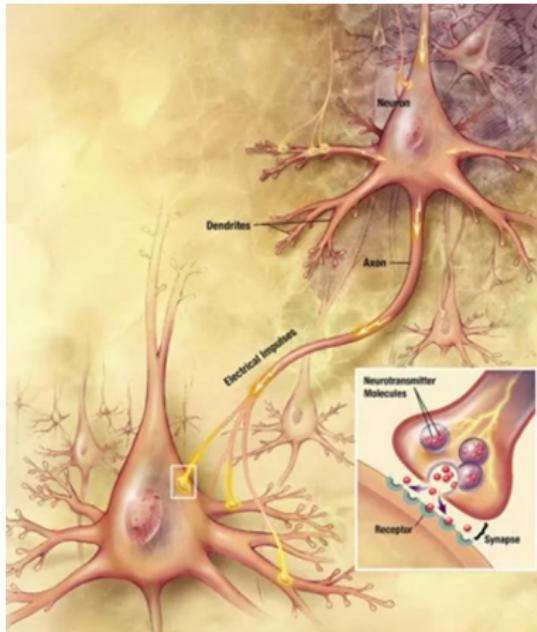
Vamos agora entrar na **Parte 2** da disciplina, onde estudaremos algoritmos **mais avançados**:

- Redes Neurais Artificiais (algoritmos de aprendizado profundo)
- Árvores de Decisão

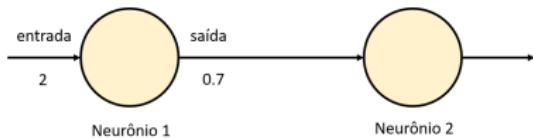
OBS: Tais algoritmos são amplamente utilizados atualmente.

- **Origem:** Algoritmos que imitam o funcionamento do cérebro humano (redes neurais biológicas).
- Eram usadas, depois eram deixadas de lado, depois eram usadas novamente, depois deixadas de lado novamente...
- Depois dos anos 2000 o conceito voltou com tudo, agora sob o termo **aprendizado profundo**
- **Aplicações modernas:** Reconhecimento de voz, Processamento de imagens, Processamento de linguagem natural (texto), ...

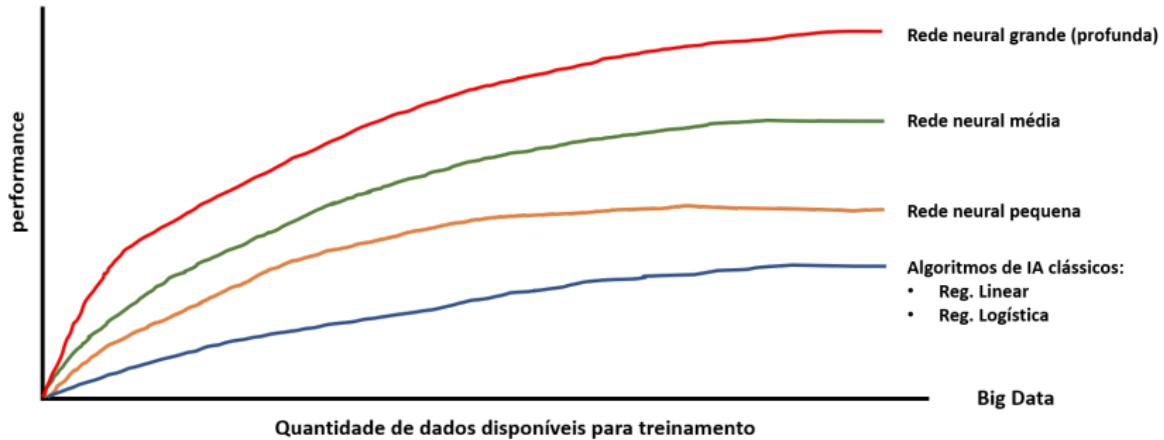
Mas como funcionam os neurônios biológicos?



- **Possuem entradas**, que recebem sinais elétricos de outros neurônios
- Eses sinais são recebidos, processados e enviados por meio das suas **saídas** a outros neurônios.
- Os neurônios seguintes recebem esses sinais de saída como sendo suas entradas, e o ciclo continua.
- Esse raciocínio serviu de inspiração para a criação das redes neurais artificias (ainda que, hoje em dia, a gente nem tenha tanta certeza de que o nossa rede de neurônios de fato funciona dessa forma).



- Para nós, um neurônio é um modelo (função), que recebe uma ou mais entradas, faz alguma conta com esses valores numéricos, e os disponibiliza em sua(s) saída(s)



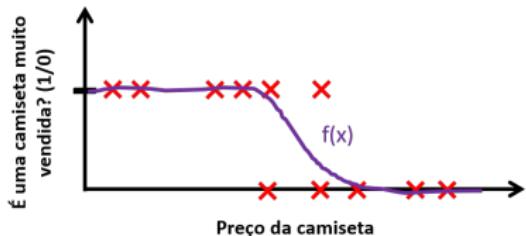
- Devido ao aumento de dados disponíveis digitalmente
- Devido à elevada capacidade de processamento a nível de hardware (GPUs, por exemplo)
- Assim, aplicações bastante complexas se tornaram possíveis (ChatGPT, por exemplo)

Exemplo: Prevendo elevada demanda

Suponha que você possui uma loja de roupas e que você possui os seguintes dados com relação aos tipos de camisetas vendidas.



Exemplo: Prevendo elevada demanda



Tratando como um modelo de **regressão logística**, temos:

Entrada:

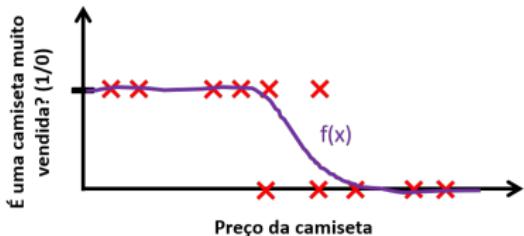
x : preço da camiseta

Saída:

$f(x)$: probabilidade da camiseta com preço x ser *top seller*.

$$f(x) = \frac{1}{1 + e^{-(wx+b)}} \rightarrow \text{Sigmoid}$$

Exemplo: Prevendo elevada demanda



Tratando como um modelo de **regressão logística**, temos:

Entrada:

x : preço da camiseta

Saída:

$f(x)$: probabilidade da camiseta com preço x ser *top seller*.

$$f(x) = \frac{1}{1 + e^{-(wx+b)}} \rightarrow \text{Sigmoid}$$

Tratando como uma **rede neural com um único neurônio**, temos:

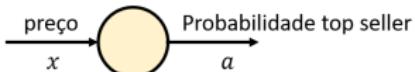
Entrada:

x : preço da camiseta

Saída:

a : probabilidade da camiseta com preço x ser *top seller*:

$$a = f(x) = \frac{1}{1 + e^{-(wx+b)}}$$



- a refere-se à quantidade de ativação feita pelo neurônio
- Nessa representação, note que o neurônio é uma unidade de regressão logística.

Exemplo: Prevendo elevada demanda

Buscando tornar o problema **mais realista**, vamos considerar agora que o fato de uma camiseta ser top seller ou não depende de 4 características principais (e não apenas do seu preço).

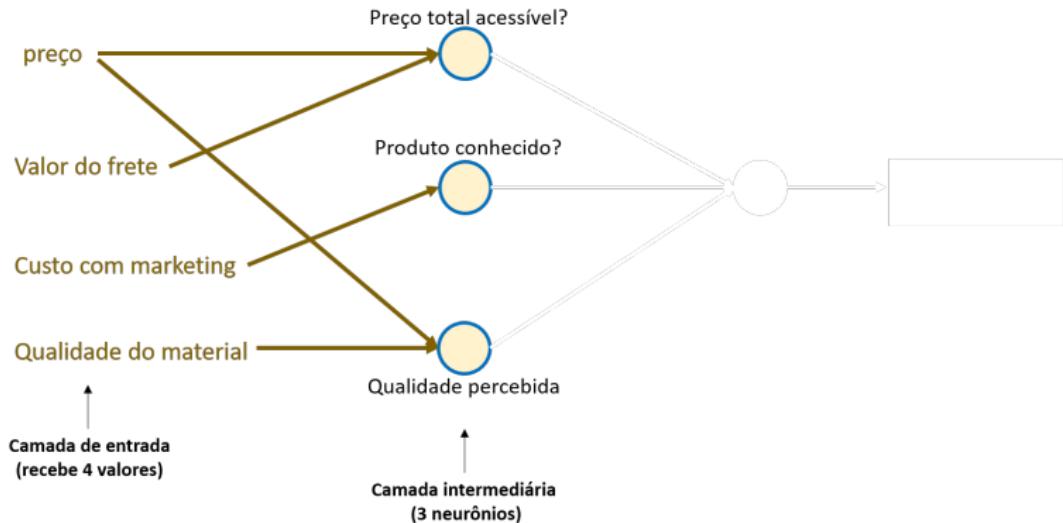
As características agora são:

- preço
- valor do frete
- custo com marketing
- qualidade do material

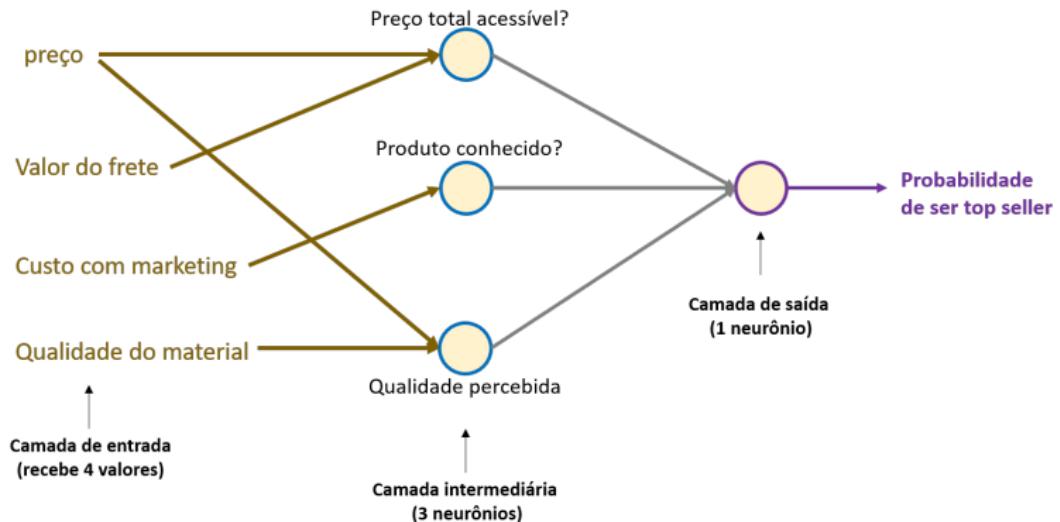
Pergunta:

Como poderíamos criar uma rede neural que descreve esse comportamento?

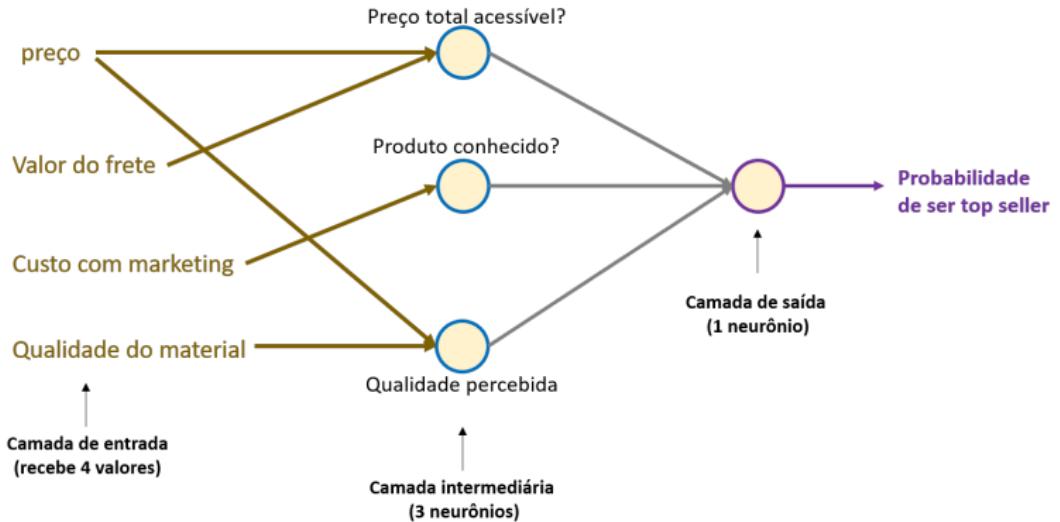
Exemplo: Prevendo elevada demanda



Exemplo: Prevendo elevada demanda

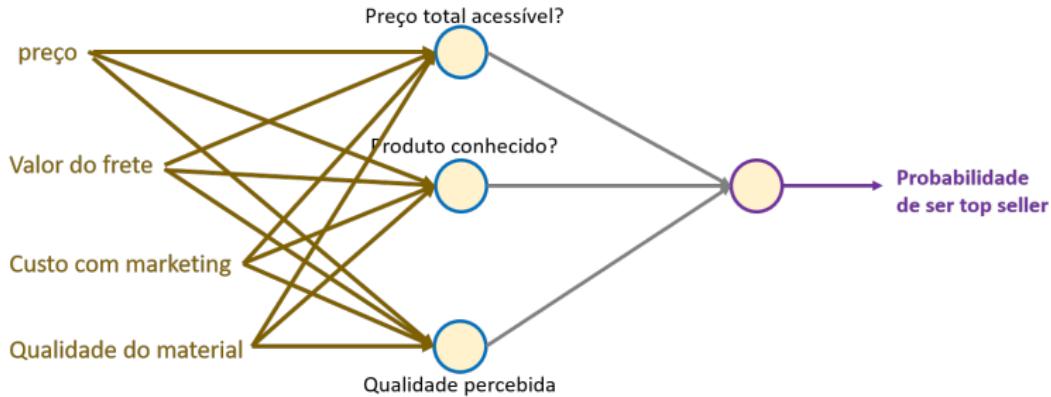


Exemplo: Prevendo elevada demanda



- A rede recebe 4 números por meio das suas 4 entradas
- Os três neurônios da Camada intermediária (também chamada de 'camada oculta') calculam suas 3 ativações correspondentes com base nos valores presentes em sua(s) entrada(s)
- Por fim, o neurônio da Camada de saída recebe esses 3 valores e calcula seu próprio valor único de ativação
- Cada neurônio pode ser entendido como sendo uma unidade elementar da regressão logística

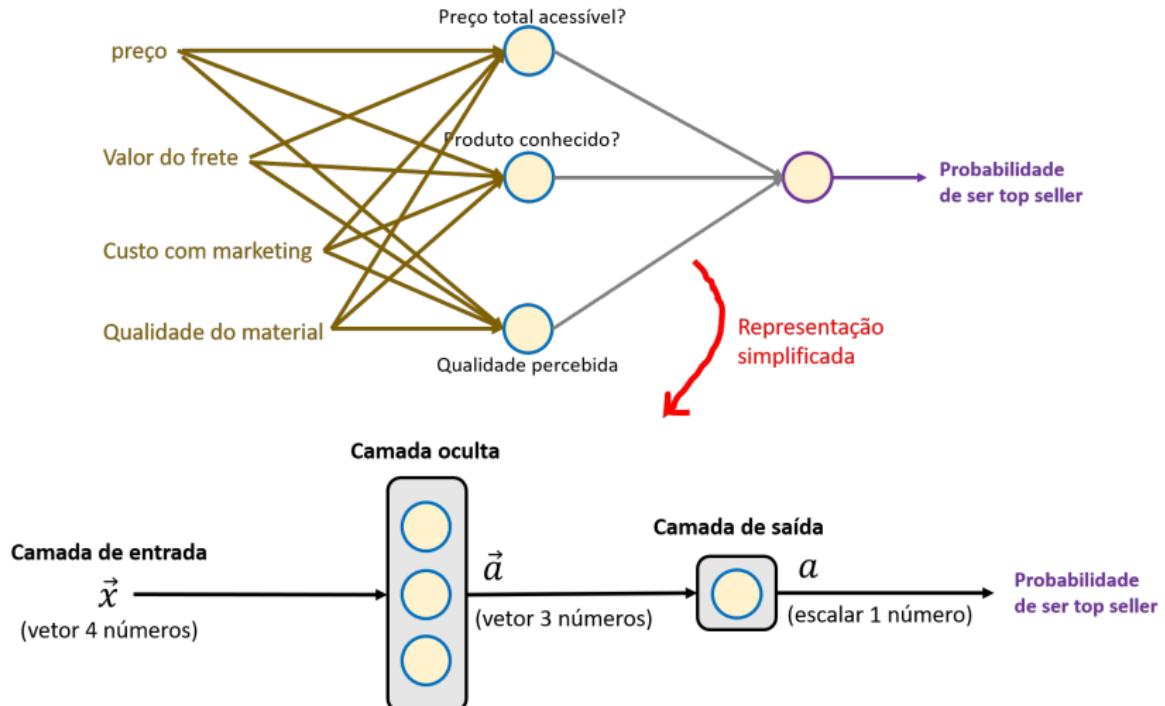
Exemplo: Prevendo elevada demanda

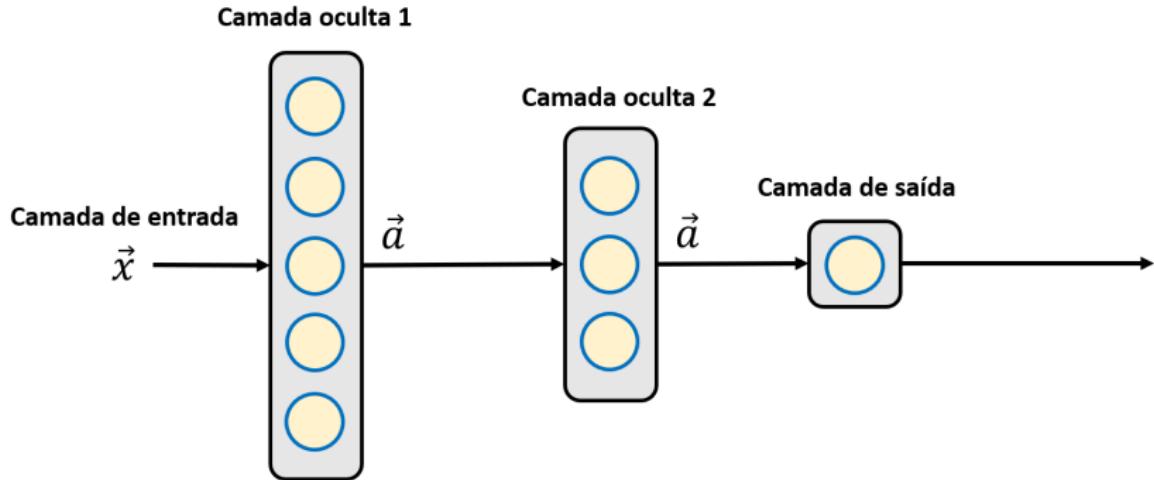


Simplificando a ideia:

- Na prática, é difícil ficar pensando quais são as entradas mais relevantes para cada neurônio, igual nós fizemos para esse exemplo
- Por isso, nas redes neurais de fato usadas nós consideramos que cada neurônio tem acesso a todos os valores que vêm da camada anterior
- A ideia é que rede neural consiga aprender de forma autônoma quais características presentes nas suas entradas são relevantes e quais não são
- Ou seja, não é tão necessário fazer **Engenharia de Características** (como fizemos no caso da Regressão linear), já que a tendência é que a própria rede crie características que ela acha que são mais apropriadas (nesse exemplo, tem-se **acessibilidade, conhecimento acerca do produto, e qualidade percebida**)
- Isso faz com que as Redes Neurais sejam tão poderosas.

Exemplo: Prevendo elevada demanda





Perguntas:

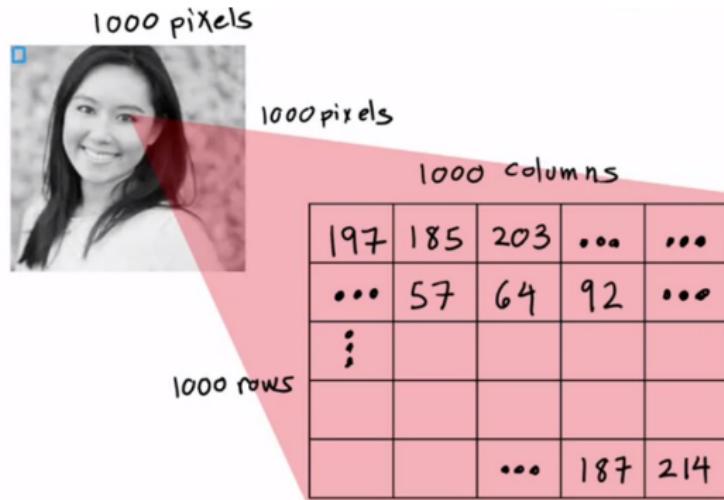
Quantas camadas escolher para a rede? Quantos neurônios colocar em cada camada?

Resposta:

Trata-se de um problema de **definição da arquitetura da rede**.

Um outro Exemplo: Reconhecimento facial

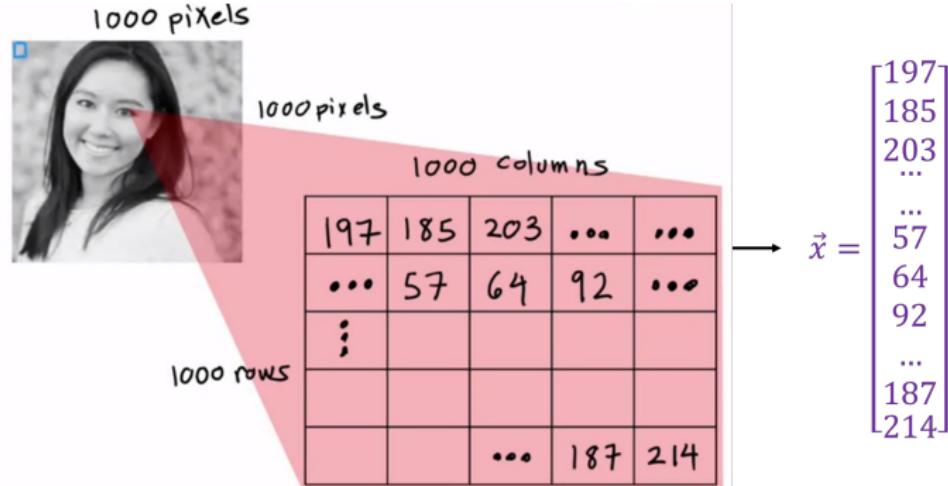
Suponha que você deseja identificar a identidade de uma pessoa a partir de uma imagem.



Fonte: **Machine Learning Specialization**, deeplearning.ai, Stanford Online, Coursera.org.

- Para o computador, a imagem é uma matriz de pixels (1000×1000 , nesse caso), onde cada pixel possui um próprio valor de intensidade de brilho (de 0 a 255, nesse caso)
- Seria possível montar um vetor de entrada para a nossa rede neural a partir da matriz?

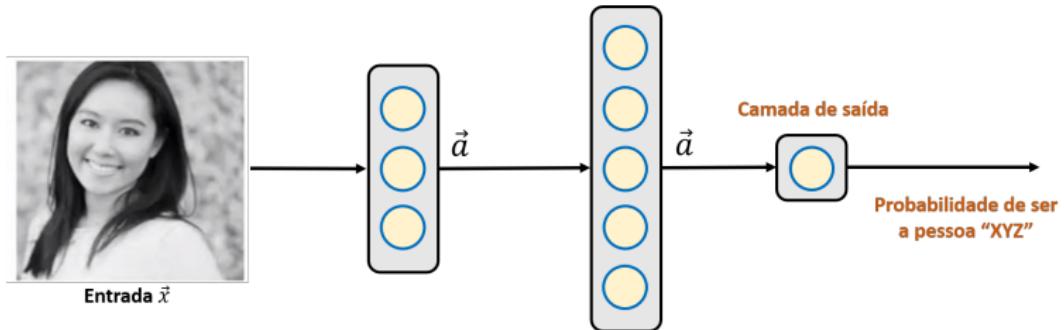
Um outro Exemplo: Reconhecimento facial



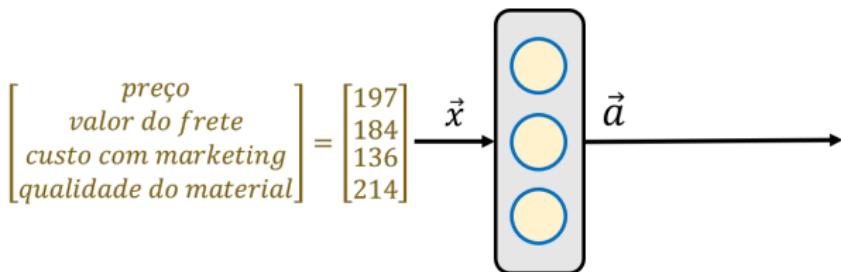
Fonte: Machine Learning Specialization, deeplearning.ai, Stanford Online, Coursera.org.

Sim, basta 'desenrolar' a matriz

Um outro Exemplo: Reconhecimento facial

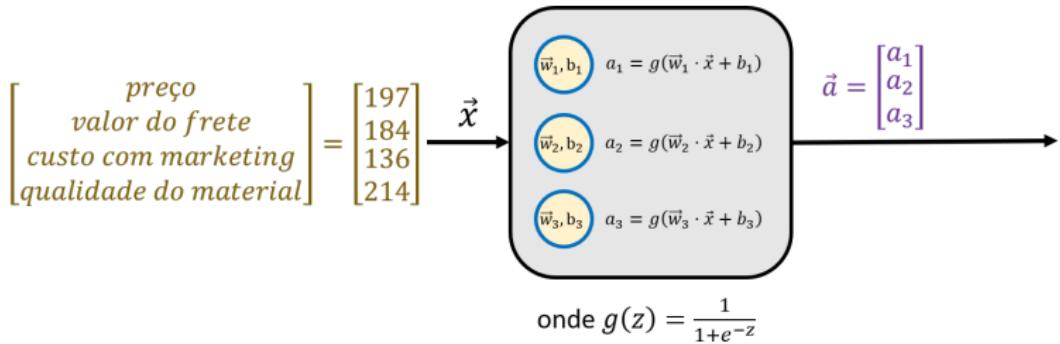


Voltando para o exemplo de previsão de demanda de camisetas (pensando apenas na camada oculta)



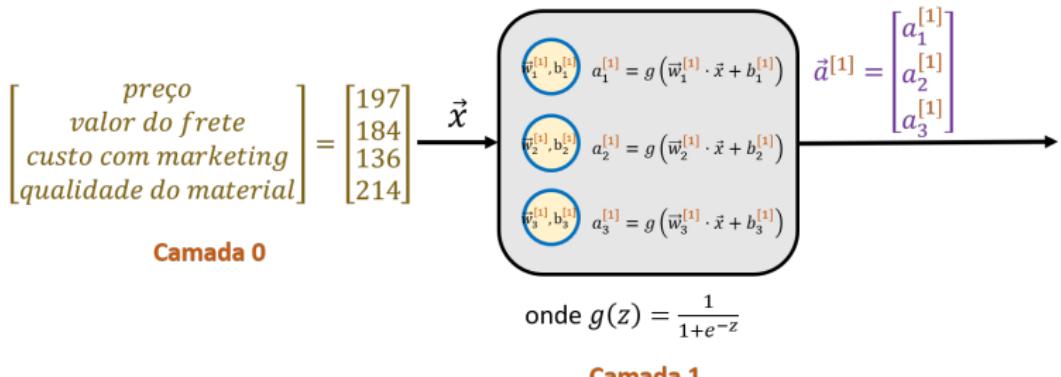
Perguntas:

- Como calcular as ativações \vec{a} dos neurônios?
- Quantos valores o vetor \vec{a} terá?



Lembre-se:

- Cada neurônio consiste em uma unidade de regressão logística



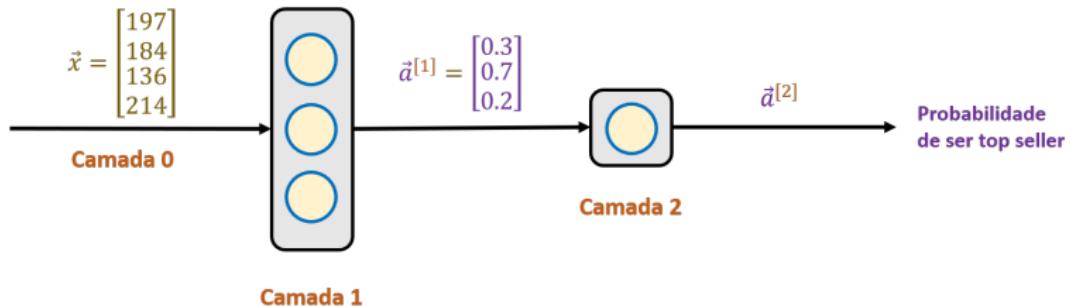
Importante!

- Usaremos o sobrescrito $[i]$ para nos referirmos a valores que correspondem à camada i .
- Pense também o seguinte: Os valores agora calculados para $\vec{a}^{[1]}$, por exemplo,

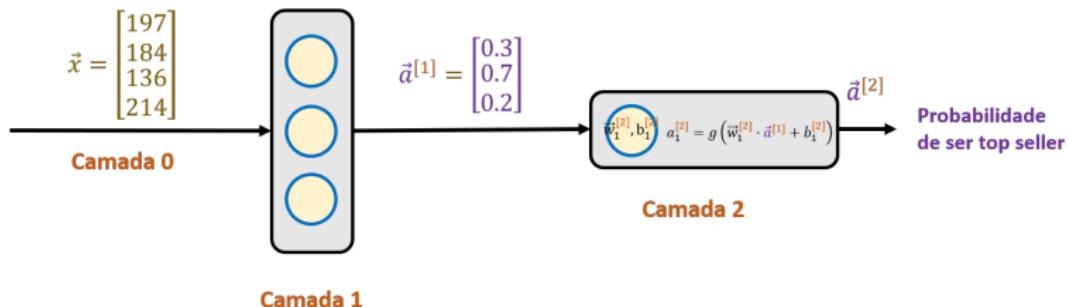
$$\vec{a}^{[1]} = \begin{bmatrix} 0.3 \\ 0.7 \\ 0.2 \end{bmatrix}$$

servem como valores de entrada para a camada que viria na sequência (camada 2).

Ou seja, por enquanto nós temos:



- Como calcular a ativação $\vec{a}^{[2]}$ do neurônio presente na Camada 2?

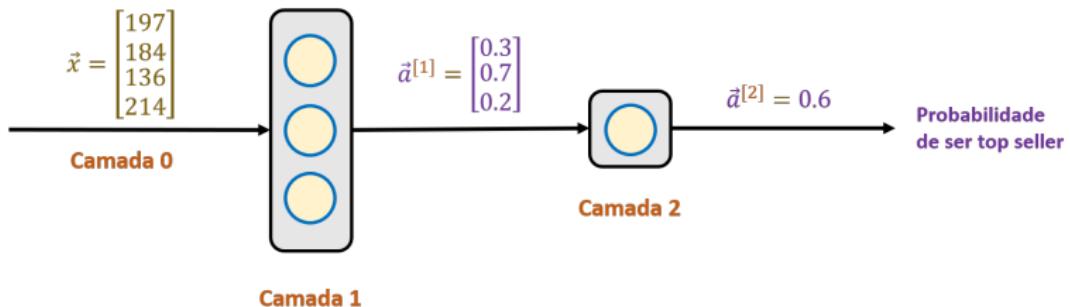


- Fazemos de forma análoga ao que havíamos feito para a Camada 1
- Note que a entrada para o neurônio da Camada 2 não é \vec{x} , mas sim $\vec{a}^{[1]}$.
- $\vec{a}^{[2]}$ vai resultar num valor entre 0 e 1, por exemplo,

$$\vec{a}^{[2]} = 0.6$$

Para o nosso exemplo, esse valor consiste na probabilidade da camiseta em questão ser top seller.

Portanto, temos o seguinte resultado final:

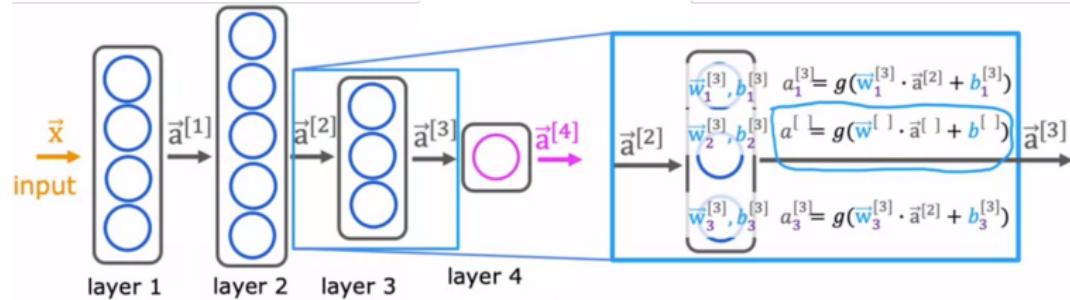


Pergunta:

E aí, essa camiseta é top seller ou não?

Resposta:

Depende. Se considerarmos um valor de limiar de 0.5, então podemos dizer que é sim top seller.

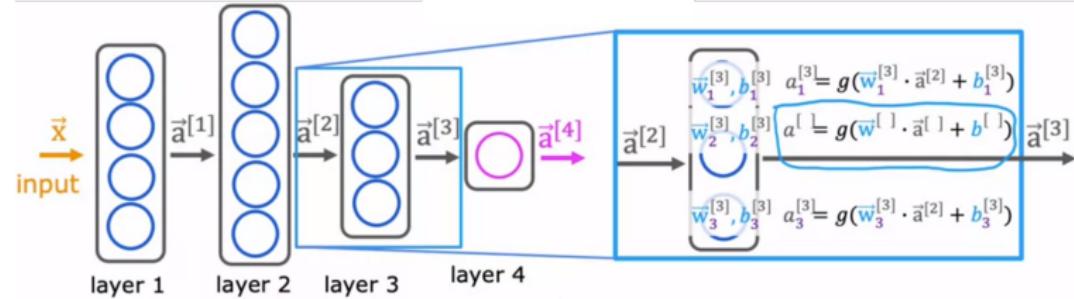


Fonte: Machine Learning Specialization, deeplearning.ai, Stanford Online, Coursera.org.

Perguntas:

- Qual é a camada de entrada dessa rede? E sua camada de saída?
- Quantas camadas ocultas essa rede possui?
- Com relação à Camada 3, quais seriam os subscritos e sobreescritos para o seu segundo neurônio?

Notação



Podemos definir que a ativação gerada pelo j -ésimo neurônio da l -ésima camada é dada por

$$a_j^{[l]} = g(\vec{w}_j^{[l]} \cdot \vec{a}^{[l-1]} + b_j^{[l]})$$

Para que essa notação possa valer também para a Camada 1, definimos também

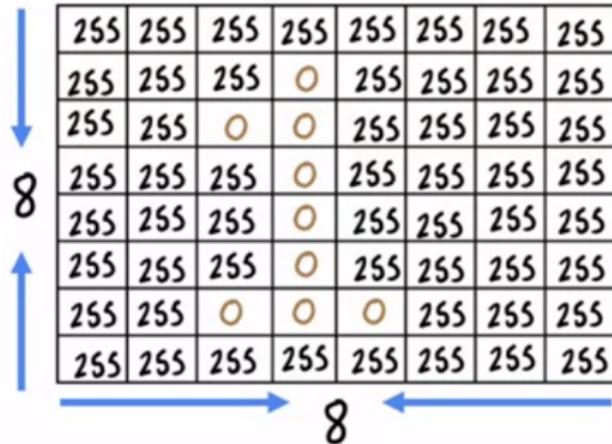
$$\vec{x} = \vec{a}^{[0]}$$

Vamos agora sistematizar o processo de realizar previsões a partir de uma rede neural já treinada.

Quando usamos uma rede já treinada para fazermos previsões, trata-se de um problema de **inferência**.

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão

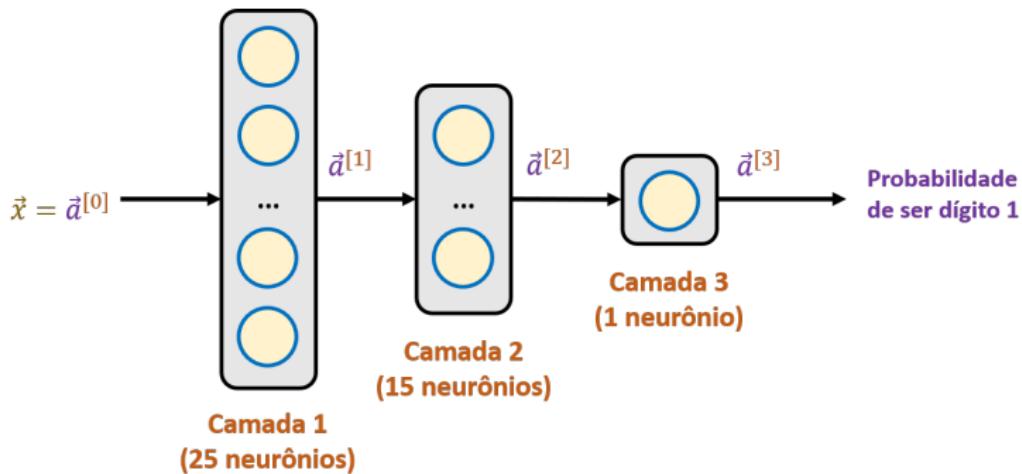
Considere que você tem uma imagem ($8 \times 8 = 64$ pixels) que possui o dígito 0 ou 1 escrito à mão:



- Valor 0 significa cor preta
- Valor 255 significa cor branca
- Valores entre 0 e 255 implicam em diferentes tons de cinza
- \vec{x} é o vetor (matriz desenrolada) contendo o valor de brilho para os 64 pixels

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão

Usaremos uma rede neural com a seguinte arquitetura:



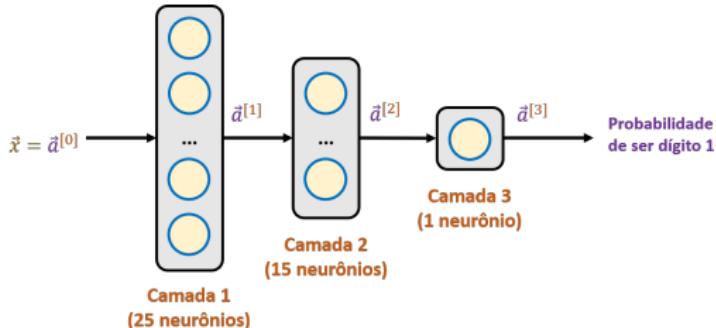
Pergunta:

Como calcular $\vec{d}^{[3]}$ a partir de $\vec{d}^{[0]} = \vec{x}$?

Resposta:

Usaremos a estratégia que acabamos de aprender (*forward propagation*) → iremos da esquerda para a direita, propagando os cálculos!

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão



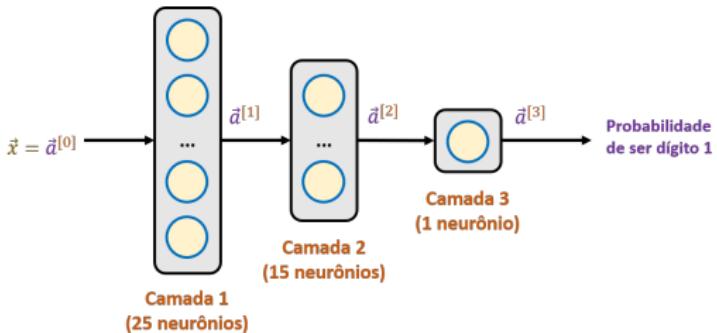
Para a Camada 1, temos:

$$\vec{a}^{[1]} = \begin{bmatrix} g(\vec{w}_1^{[1]} \cdot \vec{x} + b_1^{[1]}) \\ \vdots \\ g(\vec{w}_{25}^{[1]} \cdot \vec{x} + b_{25}^{[1]}) \end{bmatrix}$$

Para a Camada 2, temos:

$$\vec{a}^{[2]} = \begin{bmatrix} g(\vec{w}_1^{[2]} \cdot \vec{a}^{[1]} + b_1^{[2]}) \\ \vdots \\ g(\vec{w}_{15}^{[2]} \cdot \vec{a}^{[1]} + b_{15}^{[2]}) \end{bmatrix}$$

Exemplo: Reconhecendo dígitos 0 e 1 escritos à mão



Finalmente, para a Camada 3, temos:

$$\vec{a}^{[3]} = g(\vec{w}_1^{[3]} \cdot \vec{a}^{[2]} + b_1^{[3]})$$

se $a^{[3]} \geq 0.5$, $\hat{y} = 1$ (imagem contém dígito 1).

se $a^{[3]} < 0.5$, $\hat{y} = 0$ (imagem contém dígito 0).

Vamos agora explorar o conhecimento básico adquirido sobre neurônios e camadas usando código.

Nome do arquivo que trabalharemos agora:

codigo - neuronios e camadas.ipynb

OBS: Nesse código, vamos aproveitar para iniciar nossos estudos em cima do pacote **TensorFlow**, que consiste numa ferramenta de machine learning bastante utilizada.

Parte 1

Rode todo o “codigo - neuronios e camadas.ipynb” sem fazer qualquer tipo de alteração. Certifique-se de que você o compreendeu.

OBS: Caso esse seja seu primeiro contato com **TensorFlow**, é normal que demore um tempo até que você concorde com o que está sendo feito e explicado no código.

Parte 2

- 1 Adicione amostras tanto ao problema de regressão como ao problema de classificação (sem comprometer de forma significativa a capacidade do modelo de explicar esses dados)