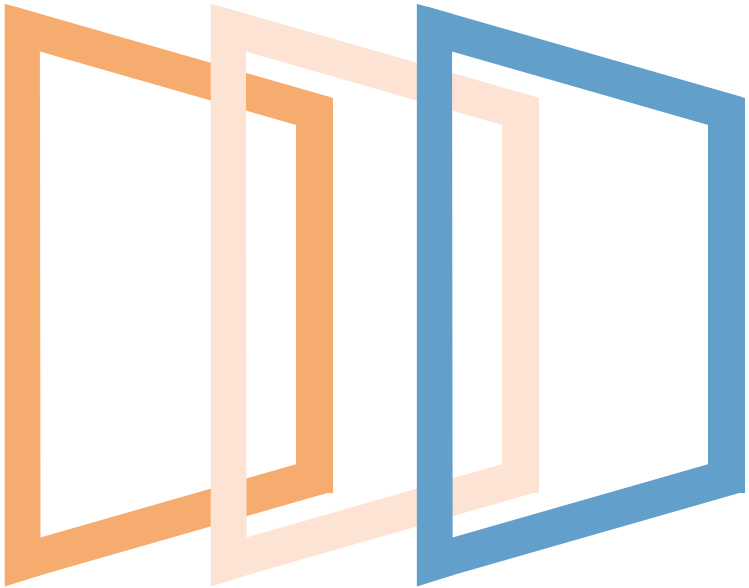
Modelos Clássicos e Redes Neurais

Thaís Ratis

Inteligência Artificial Brasil, 20.08.2025



minsait

An Indra company

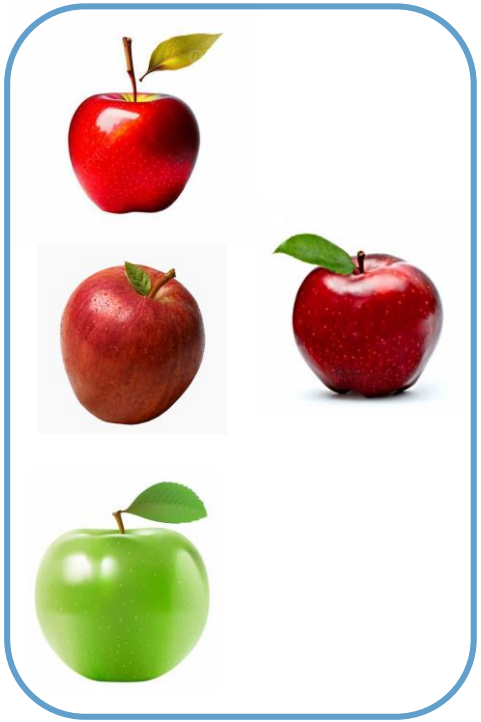
Conteúdo programático

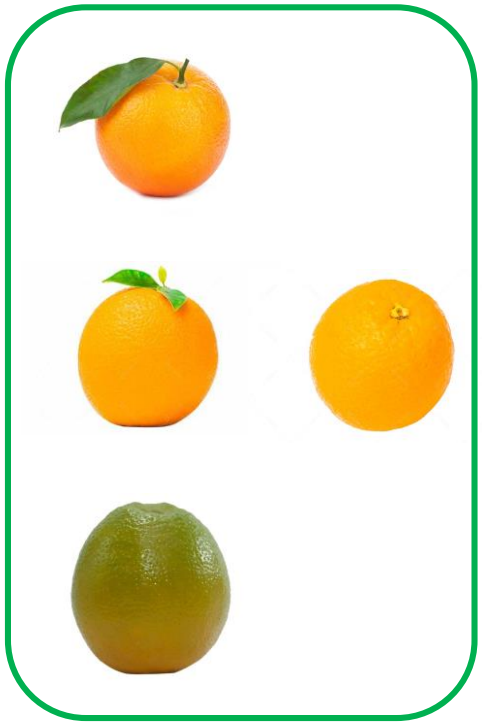
1. Modelos não supervisionados
2. Perceptron
3. Rede Neural
4. Correção de erros
5. Backpropagation
6. Modos de treinamento
7. Critérios de parada

Não Supervisionado

01

Aprendizagem Não Supervisionada



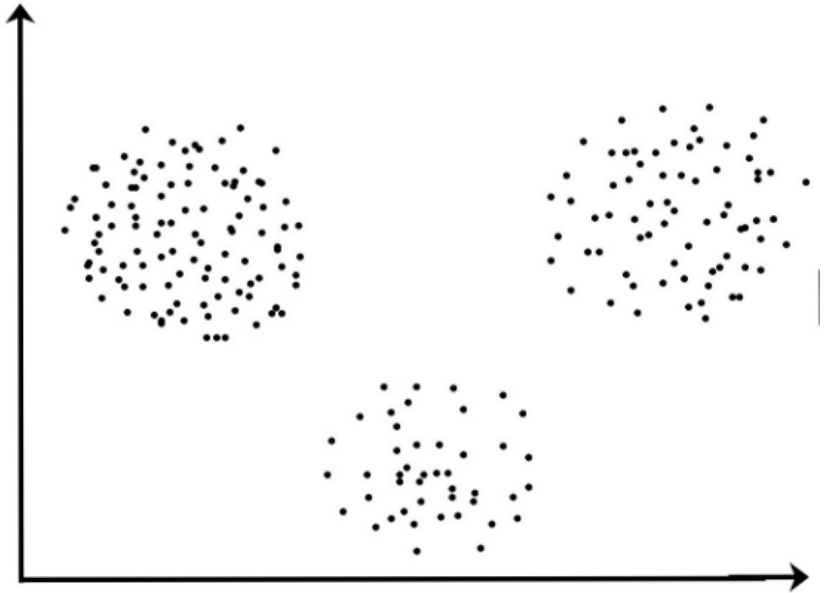


Aprendizagem Não Supervisionada

Os objetos pertencentes a cada cluster compartilham alguma característica.

Cluster: uma coleção de objetos próximos ou que satisfazem alguma relação espacial.

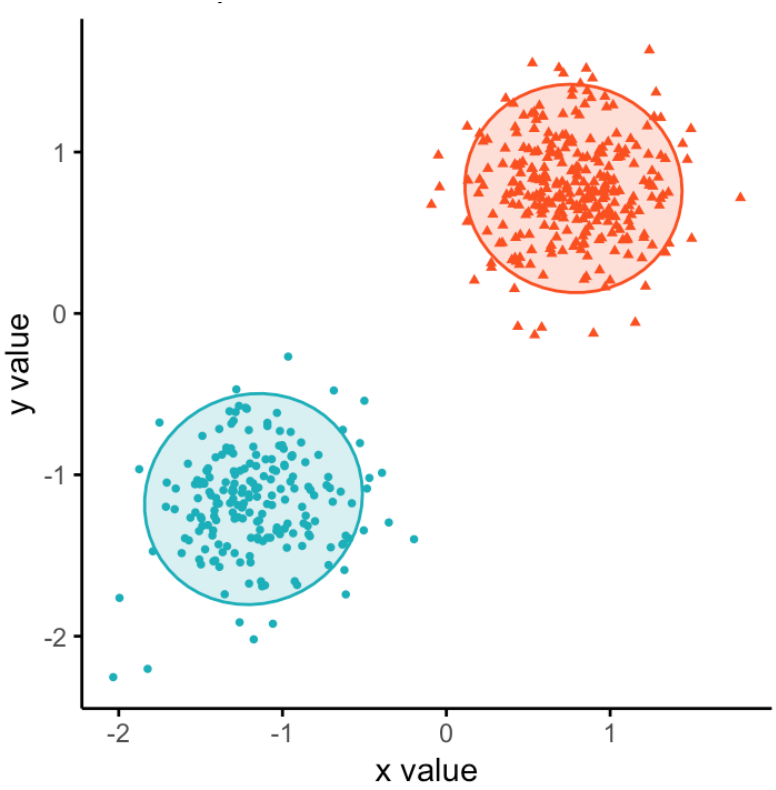
Cluster bem separado: conjunto de pontos tal que qualquer ponto em um determinado cluster está mais próximo (ou é mais similar) dos prontos de seu cluster do que de qualquer ponto de outros clusters.



Aprendizagem Não Supervisionada

Cluster baseado em centro: conjunto de pontos tal que qualquer ponto em um dado cluster está mais próximo (ou é mais similar) ao centro do cluster do que ao centro de qualquer outro cluster.

O centro de um cluster pode ser um centróide, como a média aritmética dos pontos do cluster.



Principais Características Não Supervisionado

1.1

Matriz de Similaridade e Dissimilaridade

Representa a similaridade ou a dissimilaridade entre cada par de objetos.

Cada elemento da matriz S_{nxn}, s_{ij}, é dado pela distância. $d(x_i,x_j)$, ou pela similaridade, $s(x_i,x_j)$, entre os objetos x_i e x_j.

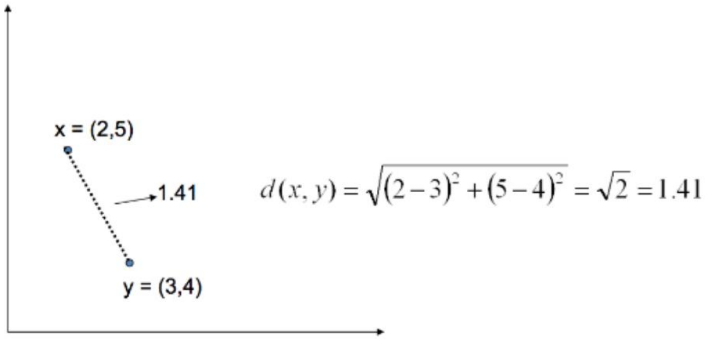
Objeto	Coordenada X1	Coordenada X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

Proximidade

Uma das medidas de dissimilaridade mais comum para objetos cujos atributos são todos contínuos é a distância euclidiana;

Uma das medidas de similaridade mais usadas é a correlação.




```
corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	identificador	idade	peso	temperatura	internacoes
identificador	1	0.48618	0.0620799	-0.319808	-0.817134
idade	0.48618	1	0.560835	-0.191917	-0.512349
peso	0.0620799	0.560835	1	0.0593191	-0.28261
temperatura	-0.319808	-0.191917	0.0593191	1	-0.035277
internacoes	-0.817134	-0.512349	-0.28261	-0.035277	1

Validação

Determina se os clusters são significativos, ou seja, se a solução é representativa para o conjunto de dados analisado.

Determina o número apropriado de clusters para um conjunto de dados, que em geral não é conhecido previamente.

Alguns fatores têm grande influência no desempenho das técnicas de agrupamento:

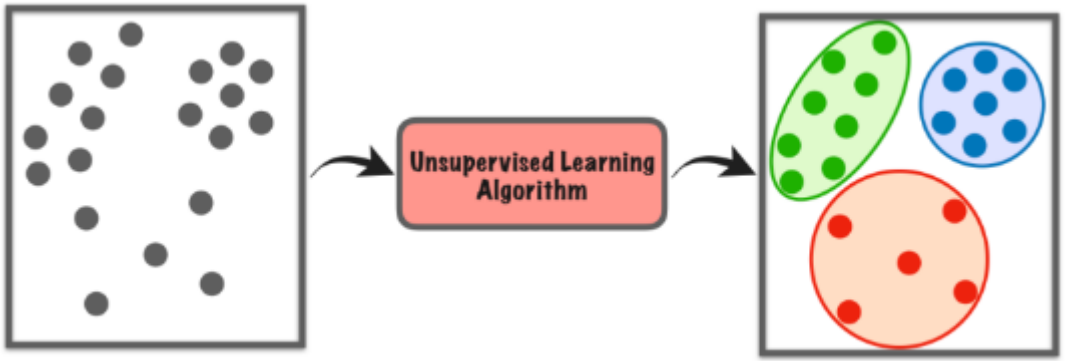
- Estrutura dos clusters (forma, tamanho, número de clusters);
- Presença de *outliers*;
- Grau de sobreposição dos clusters;
- Escolha da medida de similaridade.

K-médias (K-Means)

1.2

K-Means

O K-Means é um dos principais algoritmos de clusterização utilizados em *Machine Learning* pois é um modelo de simples **interpretabilidade** e apresenta uma boa **eficiência computacional**, ou seja, consegue modelar com facilidade utilizando o *K-Means*.



towardsdatascience

K-Means - Algoritmo

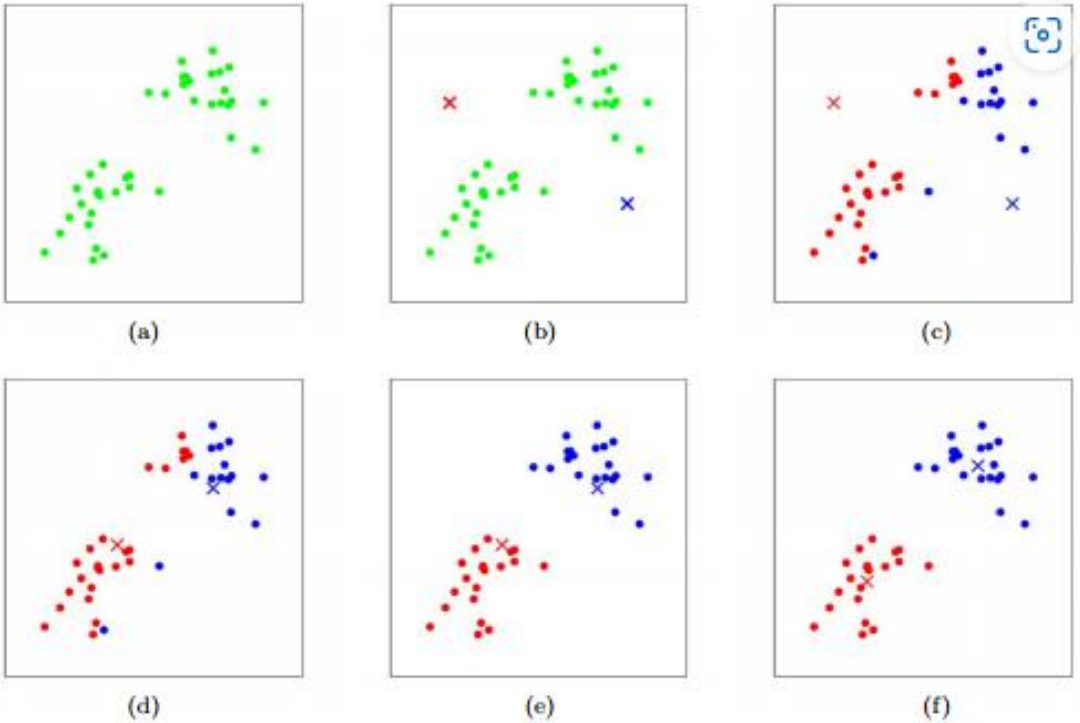
1) **Definição dos Centroides:** Supondo que deve-se separar os dados em *K* grupos (também chamados de *clusters*), para inicializar o modelo define-se *K* centroides iniciais aleatoriamente, que serão utilizados para os cálculos do modelo;

2) **Agrupamentos dos *K* grupos:** Dado os *K* grupos utilizados no modelo, cada uma das observações será associada ao **centroide mais próximo** utilizando de cálculos de distância (no caso, distância euclidiana);

3) **Reposicionamento dos Centroides:** Separados as observações em cada um dos *K* grupos, a partir das observações recalcula-se a **posição dos centroides** como a média da posição das observações dentro de determinado grupo (isto fazendo referência ao nome de *K-Means*);

4) **Processo Iterativo:** Os passos 2 e 3 serão repetidos até que o modelo considere que não houve mais alterações na **posição dos centroides**, levando em considerando uma margem de erro para as variações do posicionamento do centroide, ou mesmo quando o número máximo de iterações é atingido.

K-Means - Funcionamento



stanford.edu

Método do cotovelo

1.2.1

Método do cotovelo

A ideia é executar o KMeans para várias quantidades diferentes de clusters e dizer qual dessas quantidades é o **número ótimo de clusters**.

O que geralmente acontece ao aumentar a quantidade de clusters no KMeans é que as diferenças entre clusters se tornam muito pequenas, e as diferenças das observações intra-clusters vão aumentando. Então é preciso achar um equilíbrio em que as observações que formam cada agrupamento sejam o mais homogêneas possível e que os agrupamentos formados sejam o mais diferentes um dos outros.

Método do cotovelo

Como o KMeans calcula a distância das observações até o centro do agrupamento que ela pertence, o ideal é que essa distância seja a menor viável. Matematicamente falando, nós estamos buscando uma quantidade de agrupamentos em que a soma dos quadrados intra-clusters (ou do inglês *within-clusters sum-of-squares* (wcss)) seja a menor possível, sendo zero o resultado ótimo.

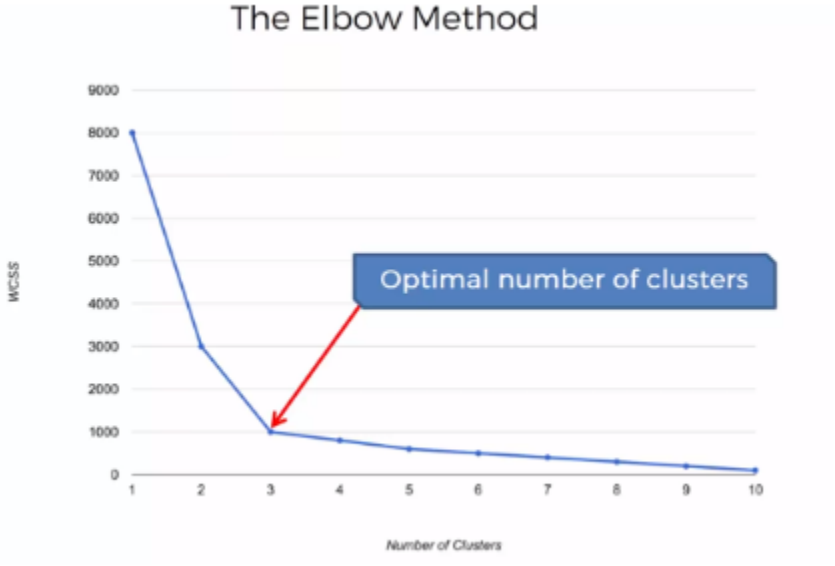
Método do cotovelo

Por exemplo, quando K = 3, a distância de soma (p, c) é a soma da distância dos pontos em um cluster do centróide.

$$\text{WCSS} = \sum_{P_i \text{ in Cluster 1}} \text{distance}(P_i, C_1)^2 + \sum_{P_i \text{ in Cluster 2}} \text{distance}(P_i, C_2)^2 + \sum_{P_i \text{ in Cluster 3}} \text{distance}(P_i, C_3)^2$$

Método do cotovelo

Na representação abaixo podemos ver que após 3 não há diminuição significativa no WCSS, então 3 é o melhor aqui. Portanto, há um formato de cotovelo que se forma e geralmente é uma boa ideia escolher o número onde esse cotovelo é formado. Muitas vezes o gráfico não seria tão intuitivo, mas com a prática fica mais fácil.



Método de silhueta

1.2.2

Método de silhueta

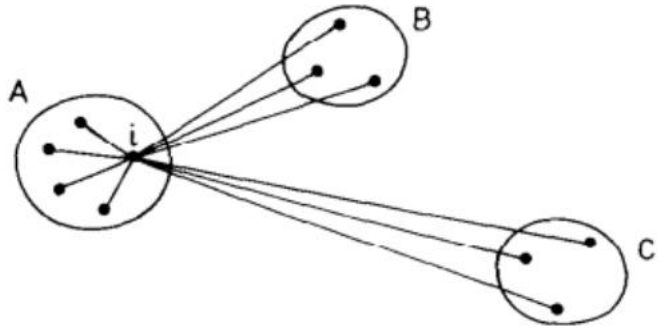
Silhueta refere-se a um método estatístico utilizado para interpretação e validação de consistência dentro dos clusters formados. O valor da silhueta (também chamado de coeficiente de silhueta) é uma medida de quão semelhante um objeto é ao seu próprio cluster (coesão), em comparação com outros clusters (separação). Este valor pode ser calculado com qualquer métrica de distância, como a euclidiana ou a de Manhattan.

Os coeficientes de silhueta variam de -1 a +1. Os valores próximos a +1 indicam que a amostra está longe dos clusters vizinhos, ou seja, indica que o objeto está no grupo que deveria se encontrar e que não deveria ser agrupado aos grupos vizinhos. Um valor 0 indica que a amostra está dentro ou muito perto do limite de decisão entre dois clusters vizinhos. Valores negativos indicam que essas amostras podem ter sido atribuídas ao cluster errado.

Método de silhueta

1. Para cada ponto *i*, calcula-se a distância intra-cluster, denominada de *a(i)*. Esta distância é calculada através da distância média de *i* para todos os outros pontos que foram agrupados dentro deste mesmo *cluster*. Pode-se interpretar *a(i)* como quão bem *i* foi atribuído ao seu grupo (quanto menor o valor, melhor a atribuição). Desta forma, a média de dissimilaridade do ponto *i* para um grupo *c* é definida como a média das distâncias de *i* para todos os pontos em *c*.

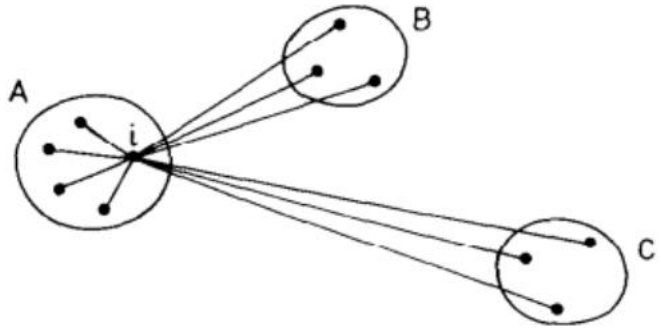
$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$



Método de silhueta

2. Para cada ponto *i*, calcula-se a distância inter-cluster, denominada por *b(i)*. Esta distância é calculada através da distância média de *i* para todos os outros pontos que foram agrupados em *clusters* distintos do ponto *i*. Assim, o valor de *b(i)* será denominado pela menor distância média de *i* para todos os pontos pertencentes a outros grupos, do qual *i* não é um membro. O grupo com essa menor dissimilaridade média é o "grupo vizinho" de *i*.

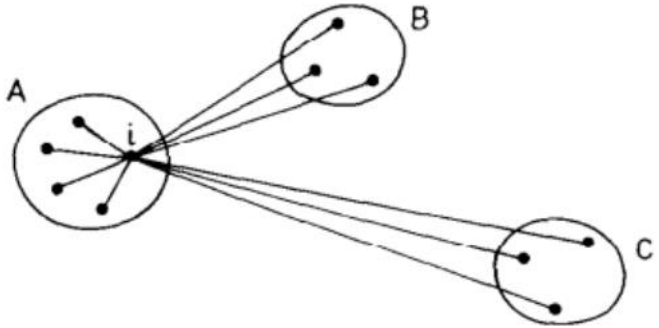
$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$



Método de silhueta

3. Para cada ponto *i*, o coeficiente de silhueta pode ser descrito pela equação abaixo. Ademais, uma ilustração correspondente aos elementos envolvidos no cálculo de *s(i)* pode ser visualizada na figura abaixo.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

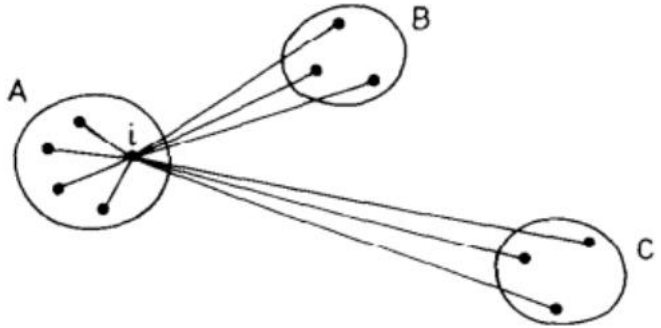


Método de silhueta

3. Para cada ponto *i*, o coeficiente de silhueta pode ser descrito pela equação abaixo. Ademais, uma ilustração correspondente aos elementos envolvidos no cálculo de *s(i)* pode ser visualizada na figura abaixo.

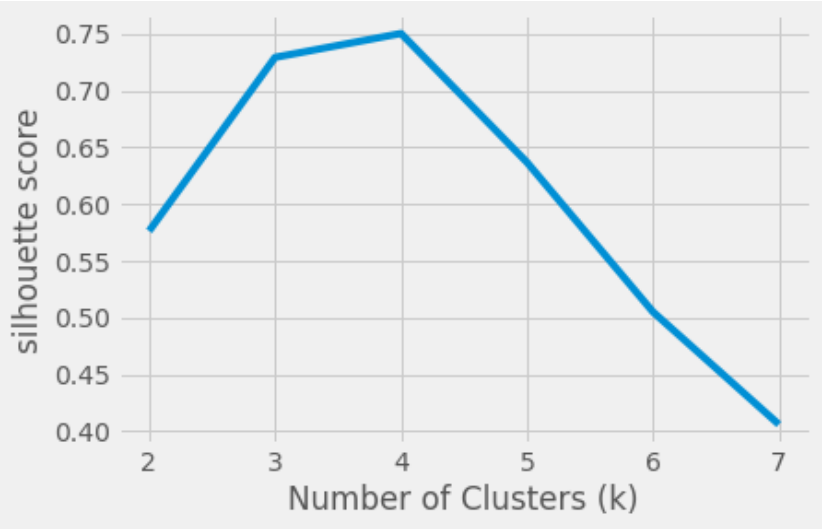
Portanto, o score de um cluster é calculado através da média dos coeficientes de silhueta de todos os pontos pertencentes a este grupo. Assim, para saber o quão bom foi o agrupamento, basta calcular o valor de silhueta do agrupamento total, ou seja, a média dos coeficientes de silhueta de todos os pontos do conjunto de dados.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

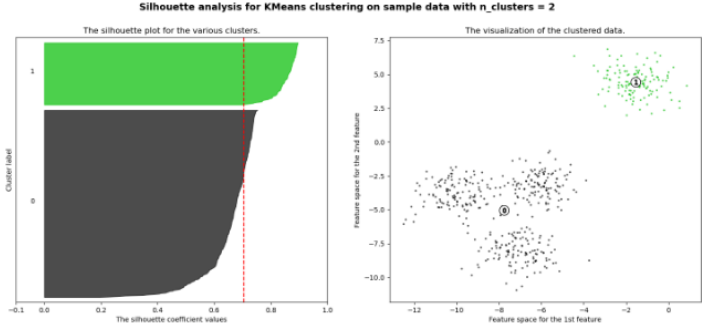


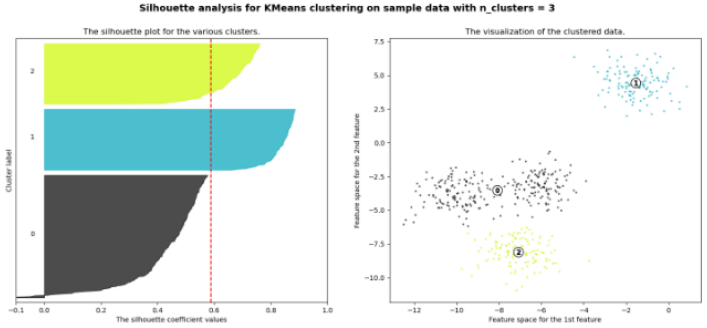
Método de silhueta

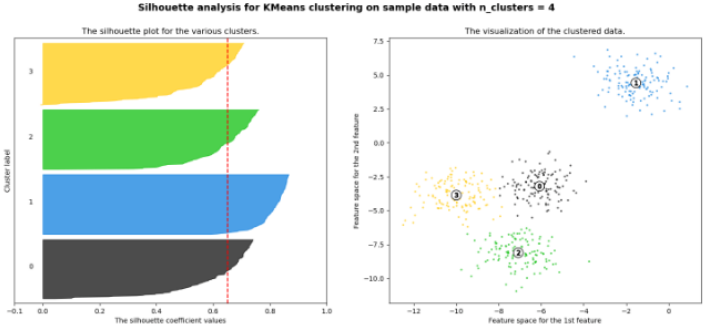
Semelhante ao método anterior, escolhemos um intervalo de valores candidatos de k (número de clusters) e, em seguida, treinamos o agrupamento K-Means para cada um dos valores de k. Para cada modelo de agrupamento k-Means representamos os coeficientes de silhueta em um gráfico e observamos as flutuações de cada cluster.

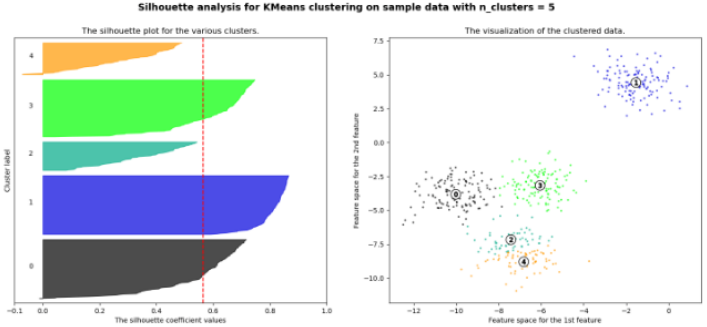


Método de silhueta









JUPYTER NOTEBOOK

Perceptron

02

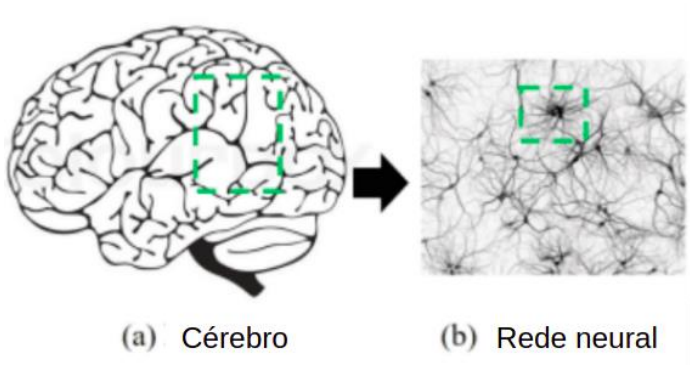
Inspiração biológica

- O cérebro é o principal órgão associado à inteligência e aprendizagem
- O cérebro é composto por uma rede complexa de aproximadamente 100 bilhões de neurônios interconectados
- Existem mais de 500 trilhões de conexões entre neurônios no cérebro humano
- Mesmo as maiores redes neurais artificiais de hoje não chegam perto do cérebro humano

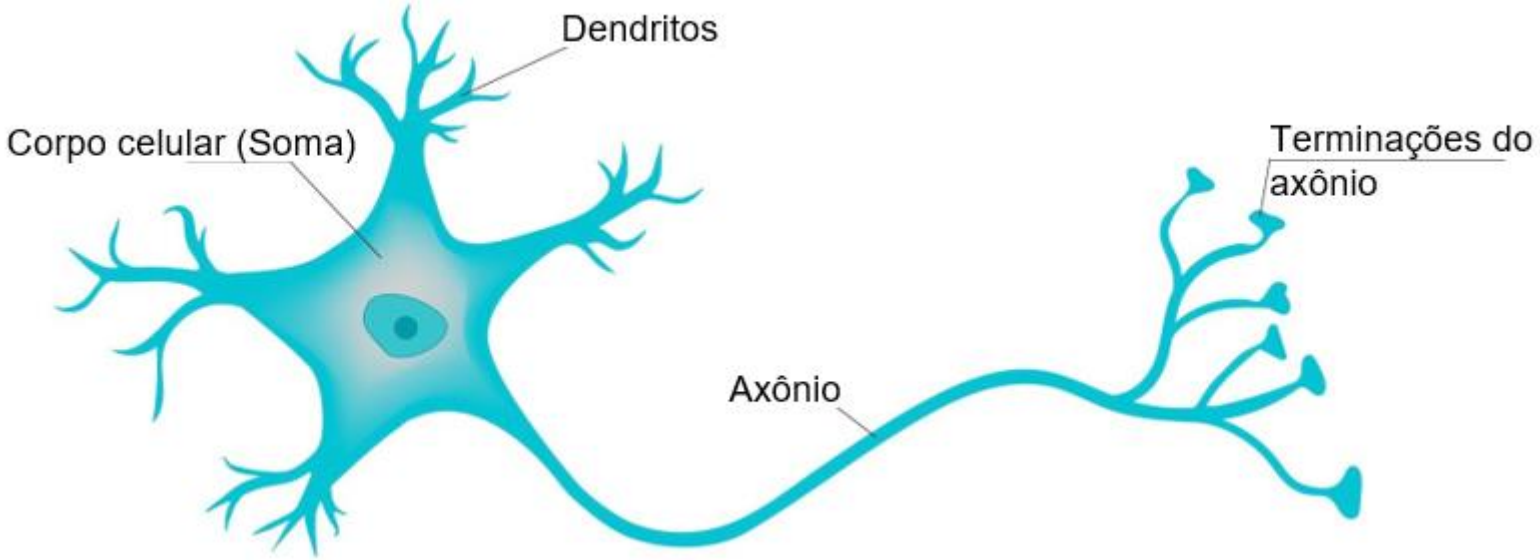
Inspiração biológica

A unidade básica de uma rede neural é o neurônio artificial

Neurônios artificiais são modelados como neurônios biológicos do cérebro, nos quais são estimulados por entradas



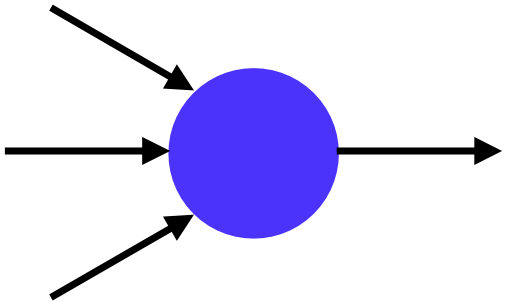
Neurônio

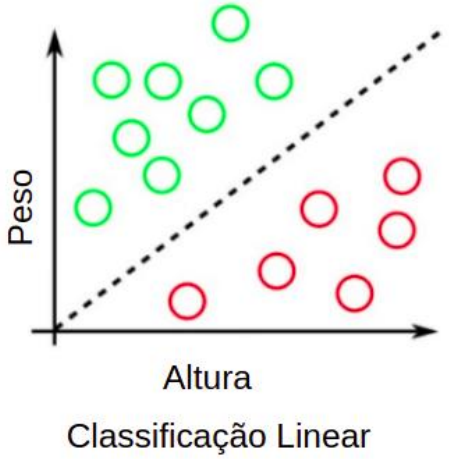

Dendritos
Corpo celular (Soma)
Terminações do axônio
Axônio

Perceptron

Em 1943, McCullock e Pitts propuseram um modelo de neurônio artificial: **perceptron**

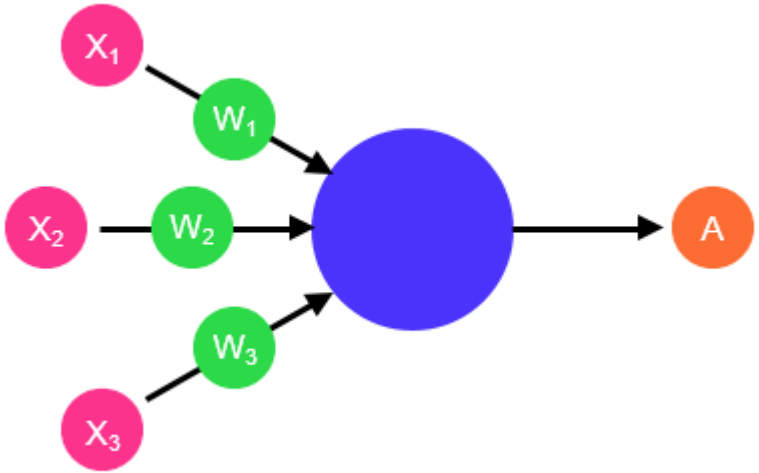
Modelo linear utilizado para classificação binária



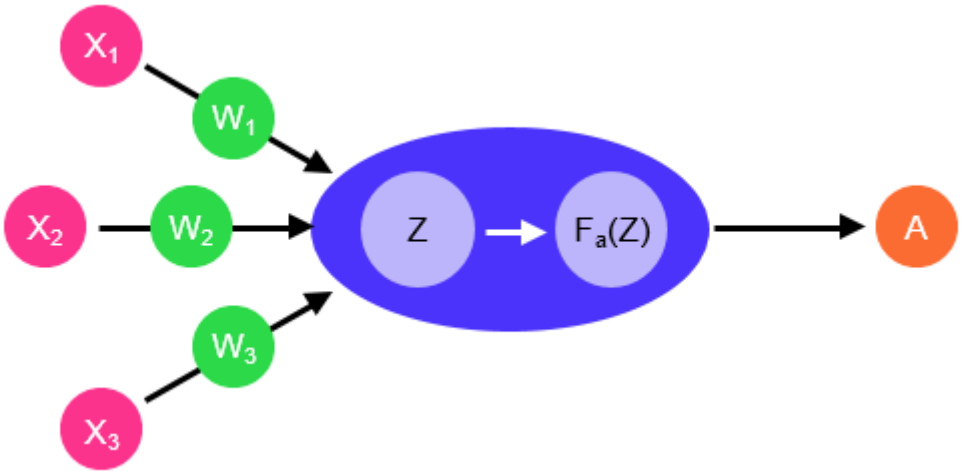


Classificação Linear

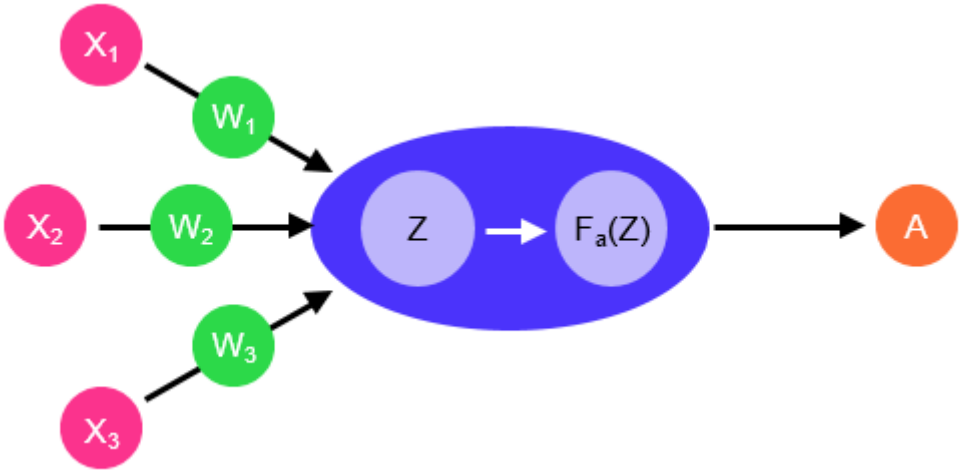
Perceptron



Perceptron

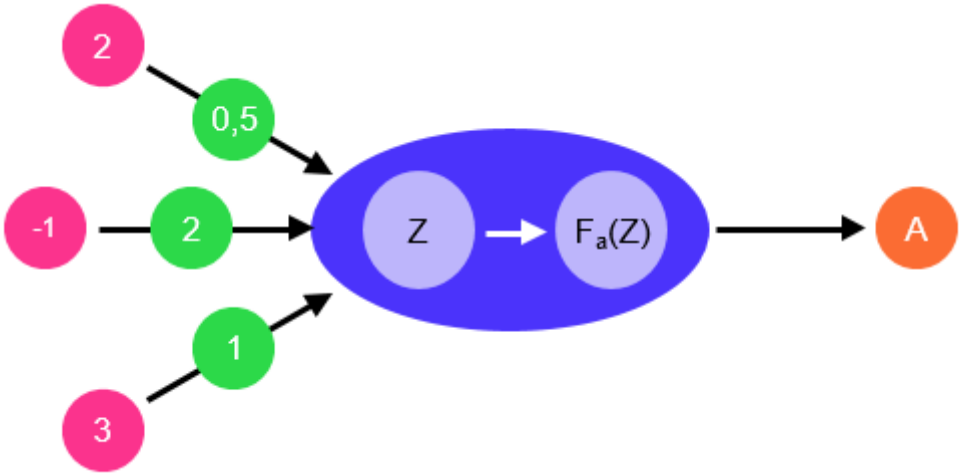


Perceptron



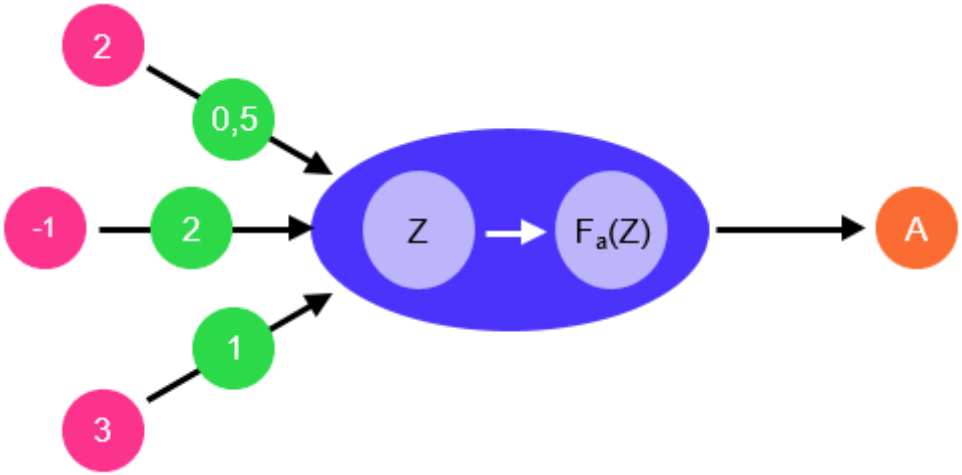
$$Z = W_1 \times X_1 + W_2 \times X_2 + W_3 \times X_3$$

Perceptron



$$Z = 0{,}5 \times 2 + 2 \times -1 + 1 \times 3 = 2$$

Perceptron



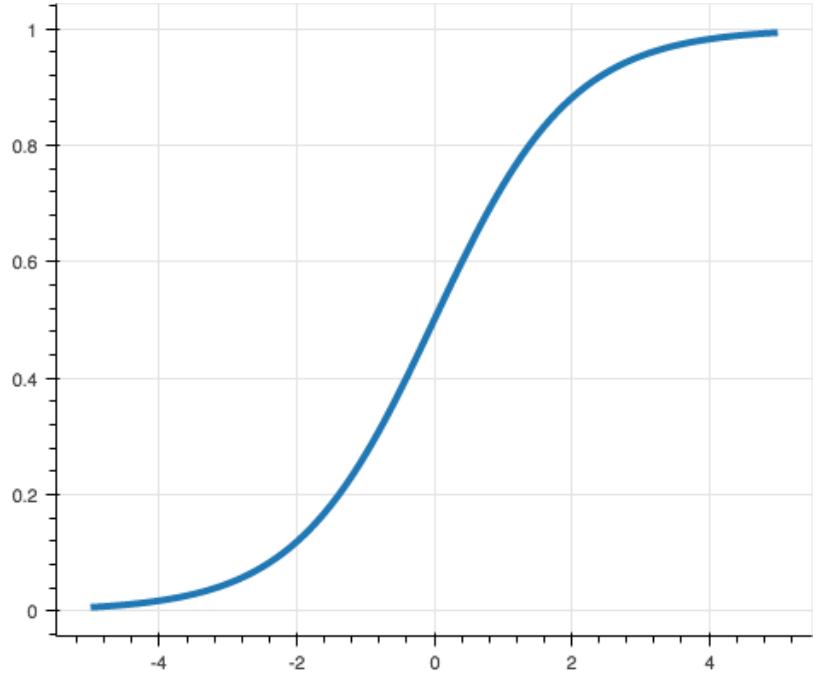
A = $F_a(2)$

Funções de ativação

Várias funções podem ser usadas como função de ativação (F_a)

Função Sigmóide:

$$f_a(z) = \frac{1}{1 + e^{-z}}$$

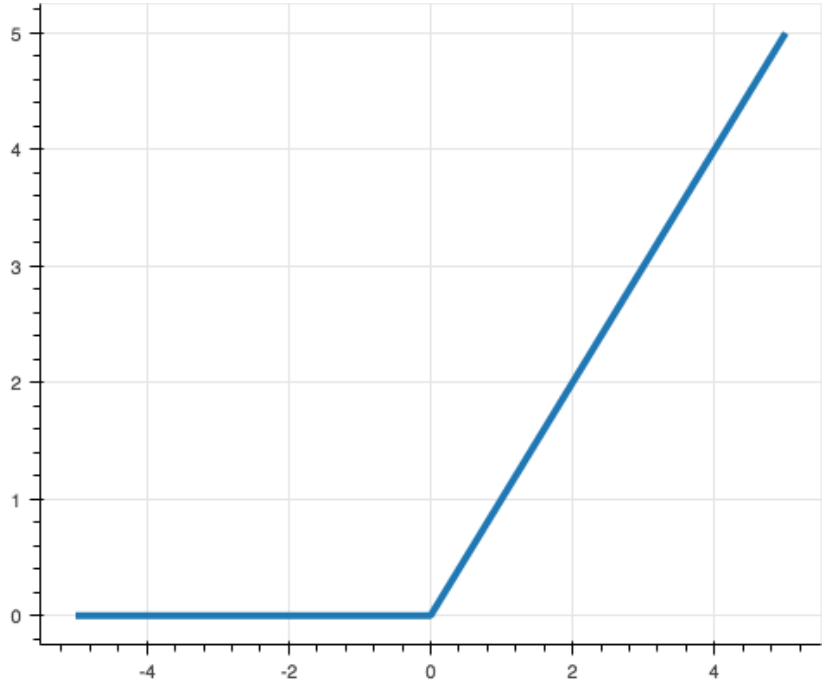


Funções de ativação

Várias funções podem ser usadas como função de ativação (F_a)

Função ReLU:

$f_a(z) = \max(0,z)$



Perceptron

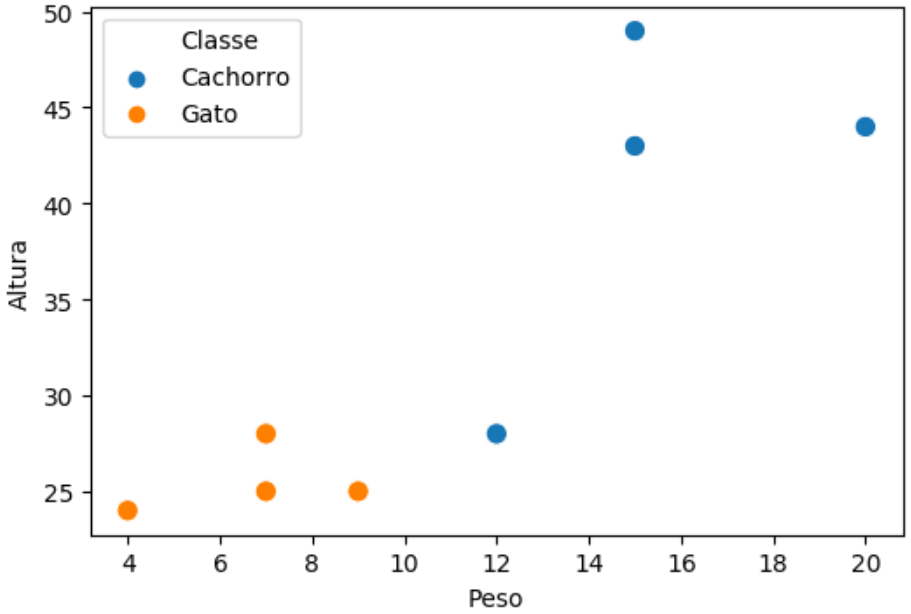
Podemos usar o neurônio artificial para diferentes problemas

Ele nos fornece uma saída de acordo com as entradas

O resultado do processamento das entradas para fornecer uma saída é determinado pela função de ativação e pelos pesos

Exemplo

Gatos e cachorros:



	Peso	Altura	Classe
0	20	44	Cachorro
1	15	43	Cachorro
2	12	28	Cachorro
3	15	49	Cachorro
4	7	28	Gato
5	7	25	Gato
6	9	25	Gato
7	4	24	Gato

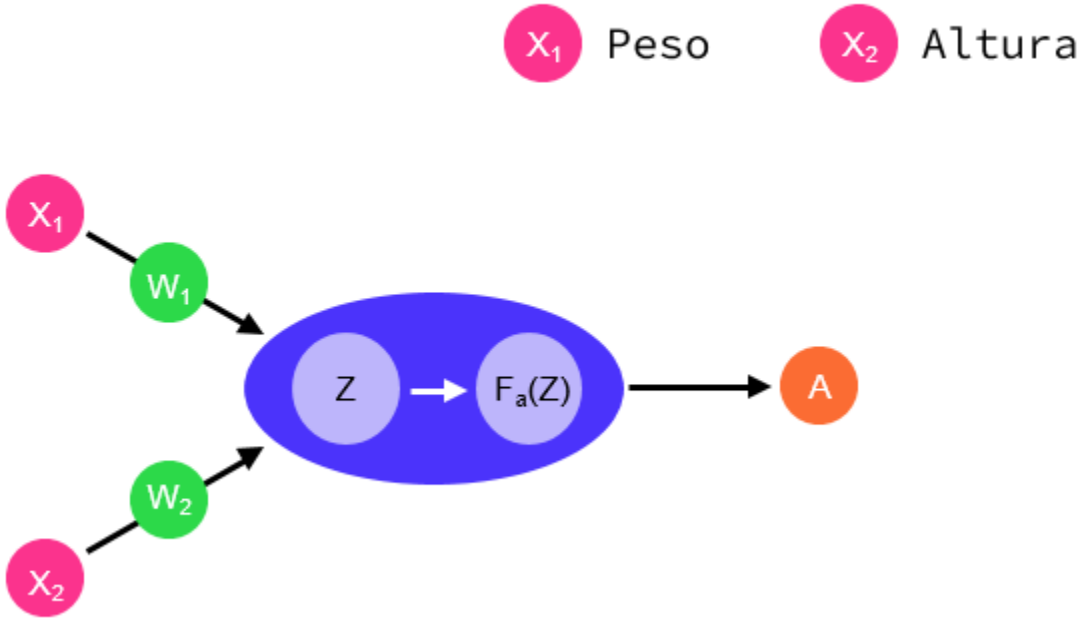
Exemplo

Dada a altura e o peso, vamos fazer o neurônio ter:

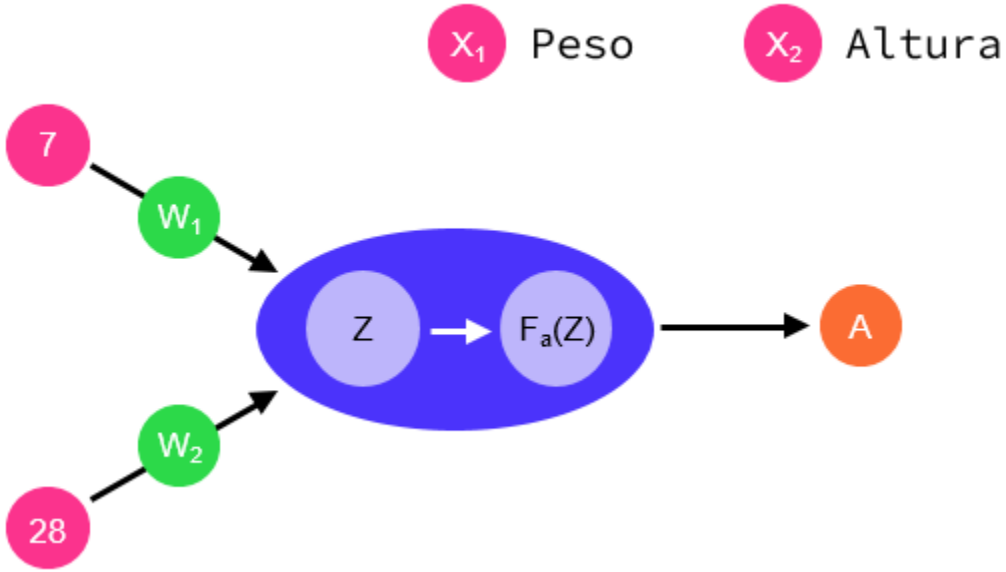
Saída **0** se for um gato

Saída **1** se for um cachorro

Exemplo



Exemplo



$$Z = W_1 \times 7 + W_2 \times 28$$

Rede Neural

03

Treinando uma rede neural

Quais são os valores de **w**?

Podemos tentar encontrá-los manualmente, porém este é um processo complexo e muitas vezes inviável

Treinando uma rede neural

Como não sabemos os pesos iniciais, vamos iniciá-los aleatoriamente

$w_1 = 0,25$

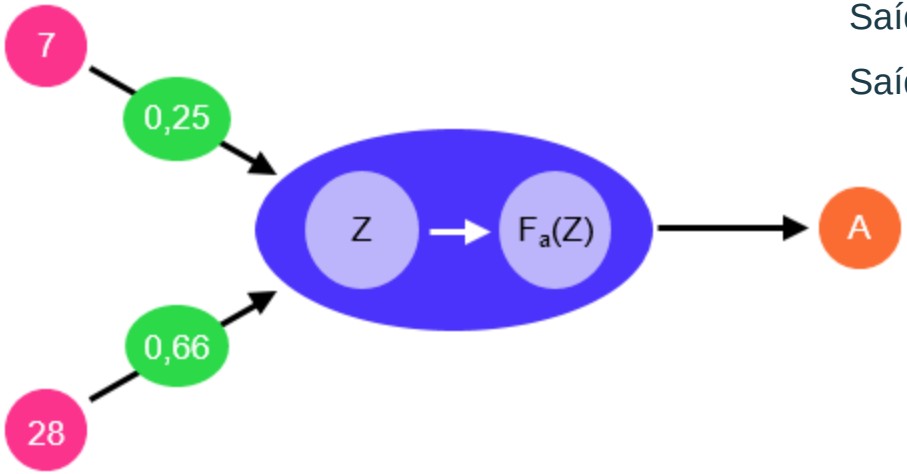
$w_2 = 0,66$

Em seguida, vamos calcular a saída para

$x_1 = 7$

$x_2 = 28$

Treinando uma rede neural



Saída 0: gato

Saída 1: cachorro

$$Z = 0{,}25 \times 7 + 0{,}66 \times 28 = 20{,}23$$

$$A = F_a(20{,}23) = 0{,}99$$

Treinando uma rede neural

Esperado	Obtido	Ajuste	Esperado - Obtido
0	0	-	0
0	1	Diminuir W	-1
1	0	Aumentar W	1
1	1	-	0

Treinando uma rede neural

Para isso, eu preciso saber o erro, ou seja a diferença entre o valor esperado e o valor obtido

Quanto maior o erro, maior precisa ser o ajuste dos valores de **w**

Então, vamos atualizar $\mathbf{w_1}$ da seguinte forma:

$$w_1 \leftarrow w_1 + \eta\,(esperado - obtido)\,x_1$$

Taxa de aprendizagem: controla o tamanho da atualização

Treinando uma rede neural

Cada ajuste vai melhorando um pouco a rede neural

Vamos ajustar os pesos para cada um dos exemplos dos dados de treinamento

$$w_1 \leftarrow w_1 + \eta(esperado - obtido)x_1$$

$$w_2 \leftarrow w_2 + \eta(esperado - obtido)x_2$$

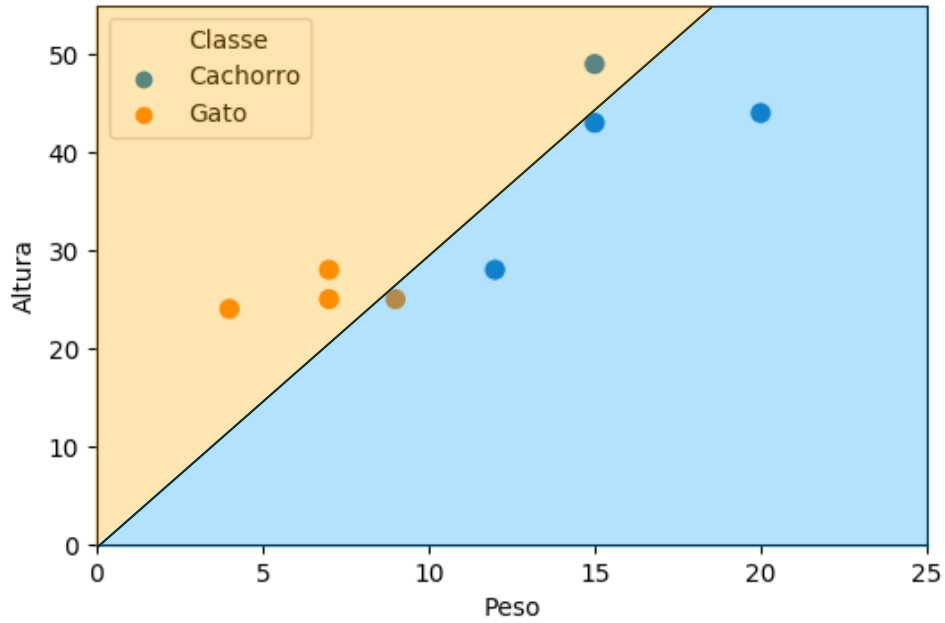
Treinando uma rede neural

Treinamento de todas as amostras do conjunto.

Esse número de vezes é chamado de **épocas**

Exemplo

Temos como resultado:



Treinando uma rede neural

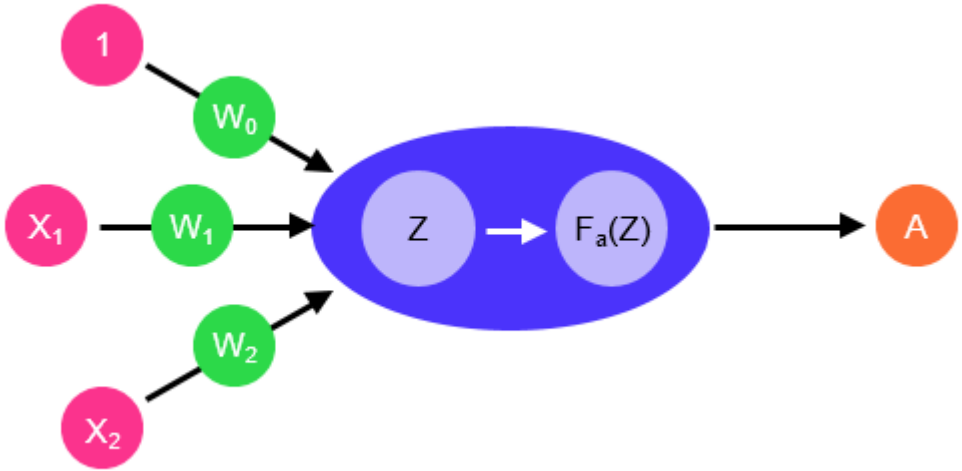
- Usando o neurônio com a estrutura que fizemos, sempre teremos uma fronteira linear partindo da origem

- Para permitir que a fronteira possa se deslocar no eixo X, vamos adicionar uma entrada extra no neurônio chamada de **viés**

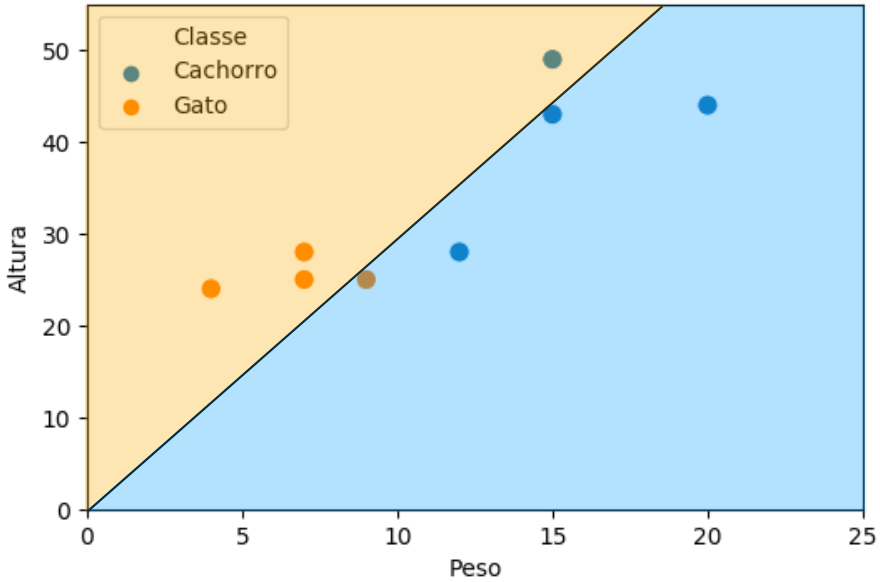
- O viés geralmente recebe 1 como entrada

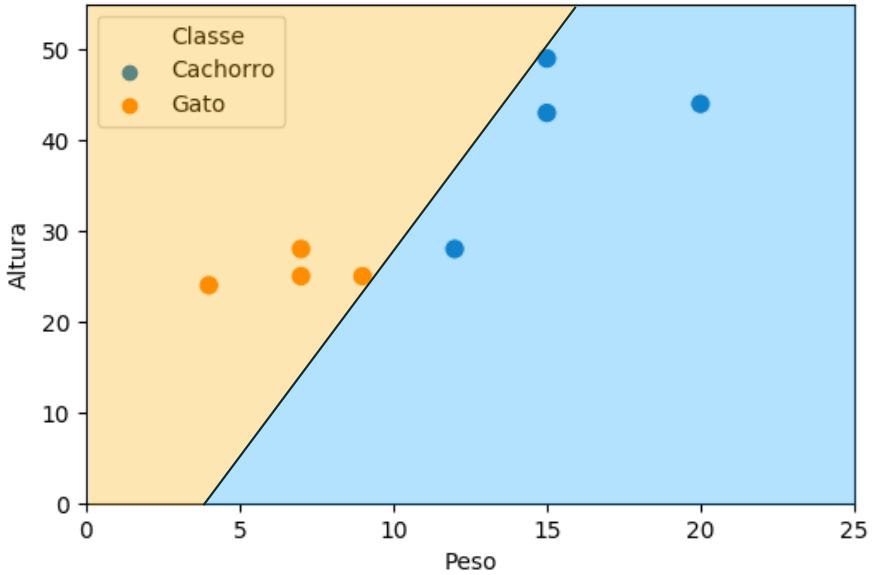
- O peso dessa entrada segue o comportamento das outras

Treinando uma rede neural



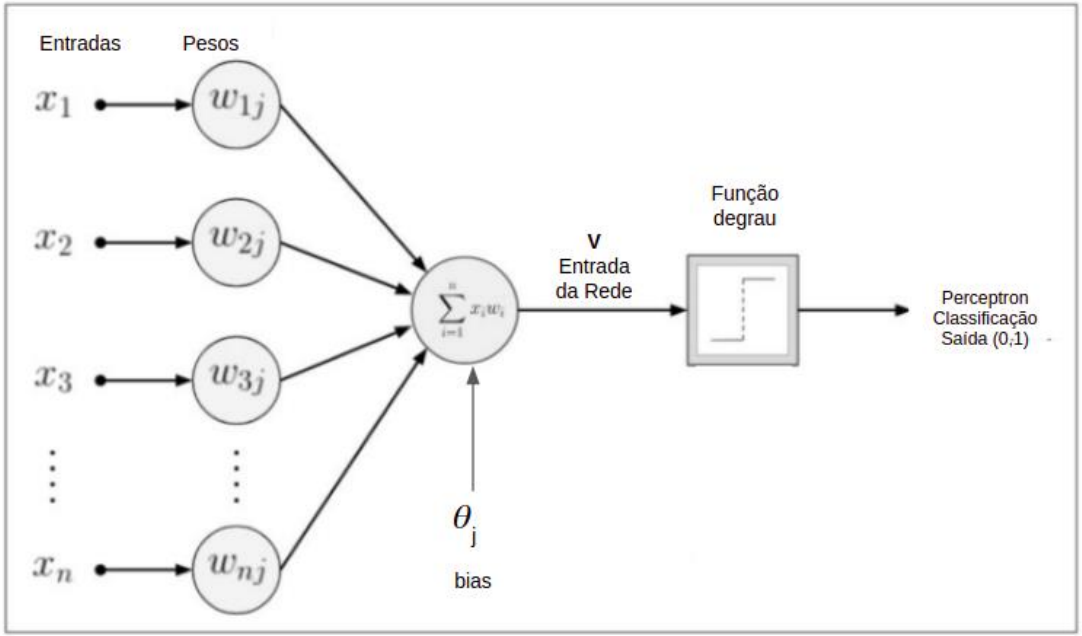
Viés (bias)


Classe
Cachorro
Gato
Altura
Peso
0
10
20
30
40
50
0
5
10
15
20
25


Classe
Cachorro
Gato
Altura
Peso
0
10
20
30
40
50
0
5
10
15
20
25

Estrutura do perceptron

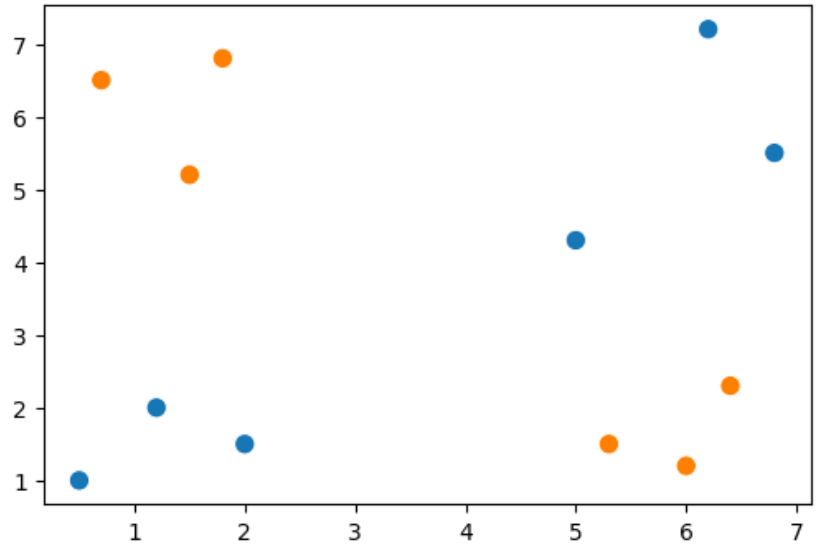
Função de ativação de Heaviside: $f(v) = \begin{cases} 0 & v < 0 \\ 1 & v >= 0 \end{cases}$



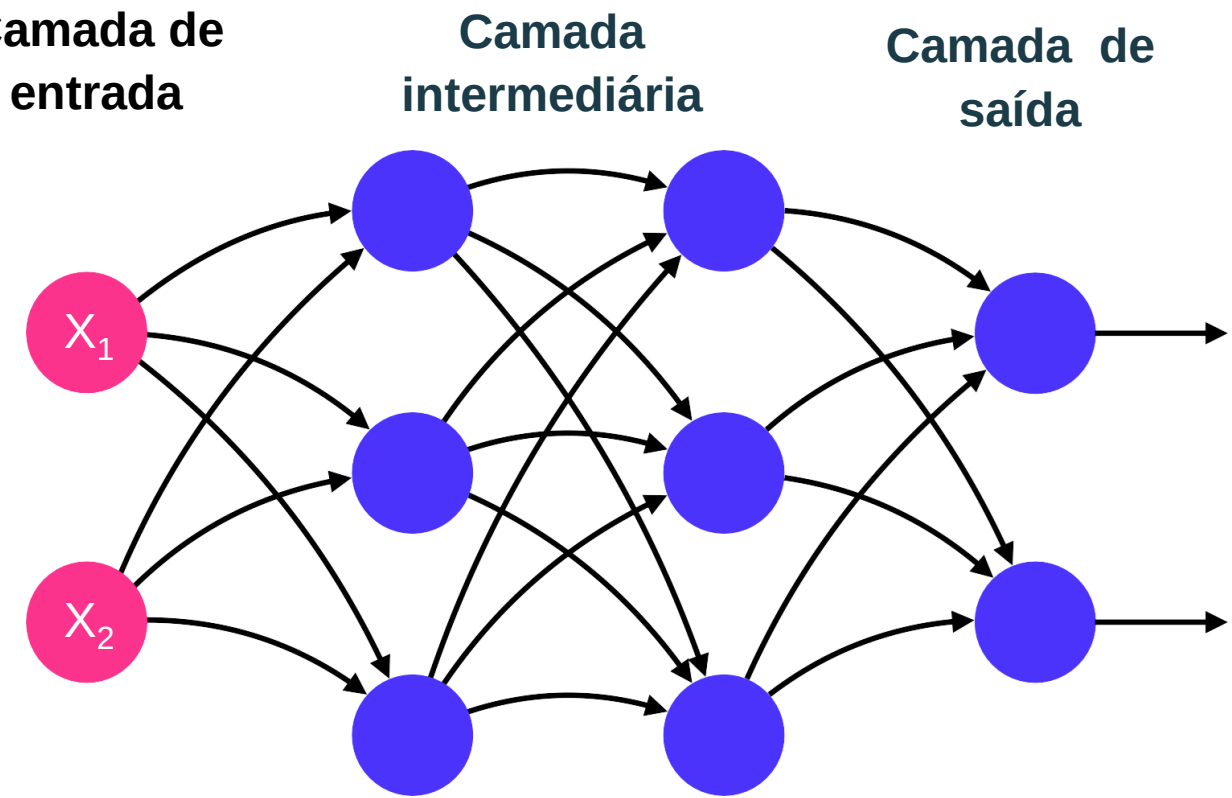
Treinando uma rede neural

Como vimos, o perceptron possui a limitação de criar apenas fronteiras lineares

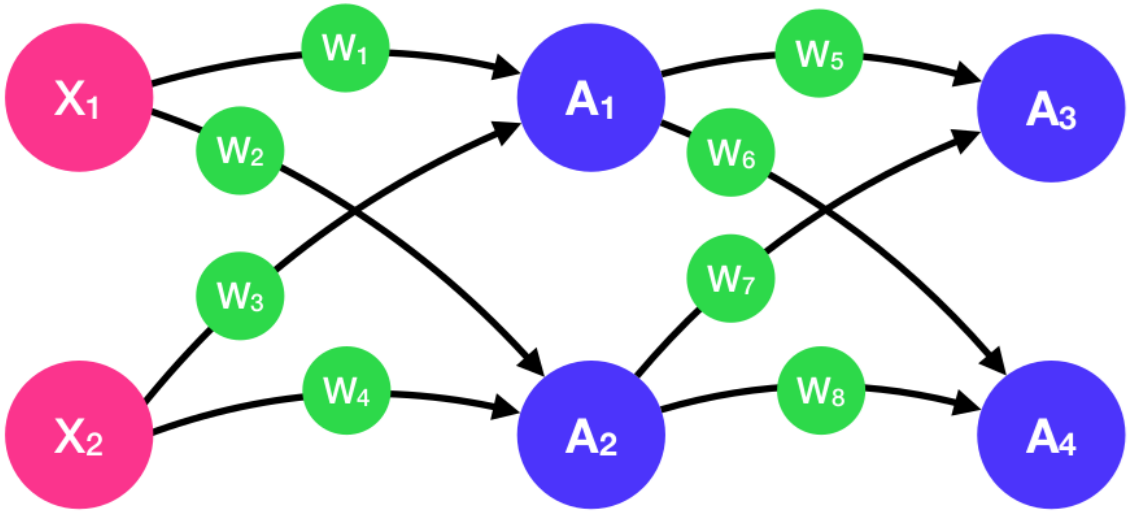
Ele não consegue classificar bem um conjunto de dados dessa forma:



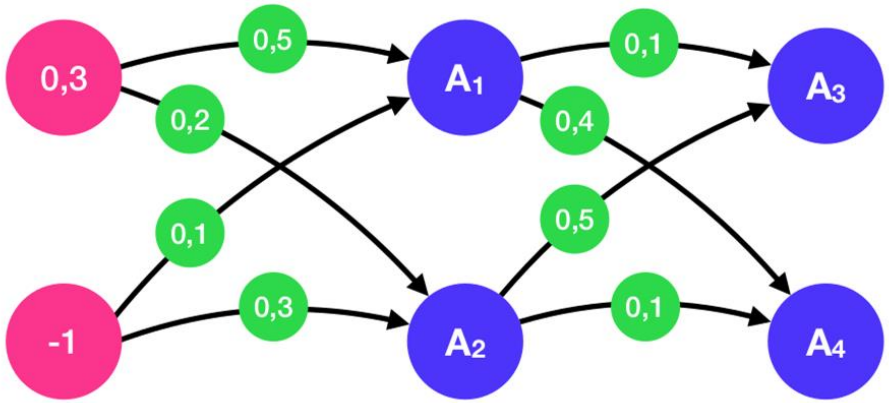
Rede neural


Camada de entrada
Camada intermediária
Camada de saída
X_1
X_2

Rede neural



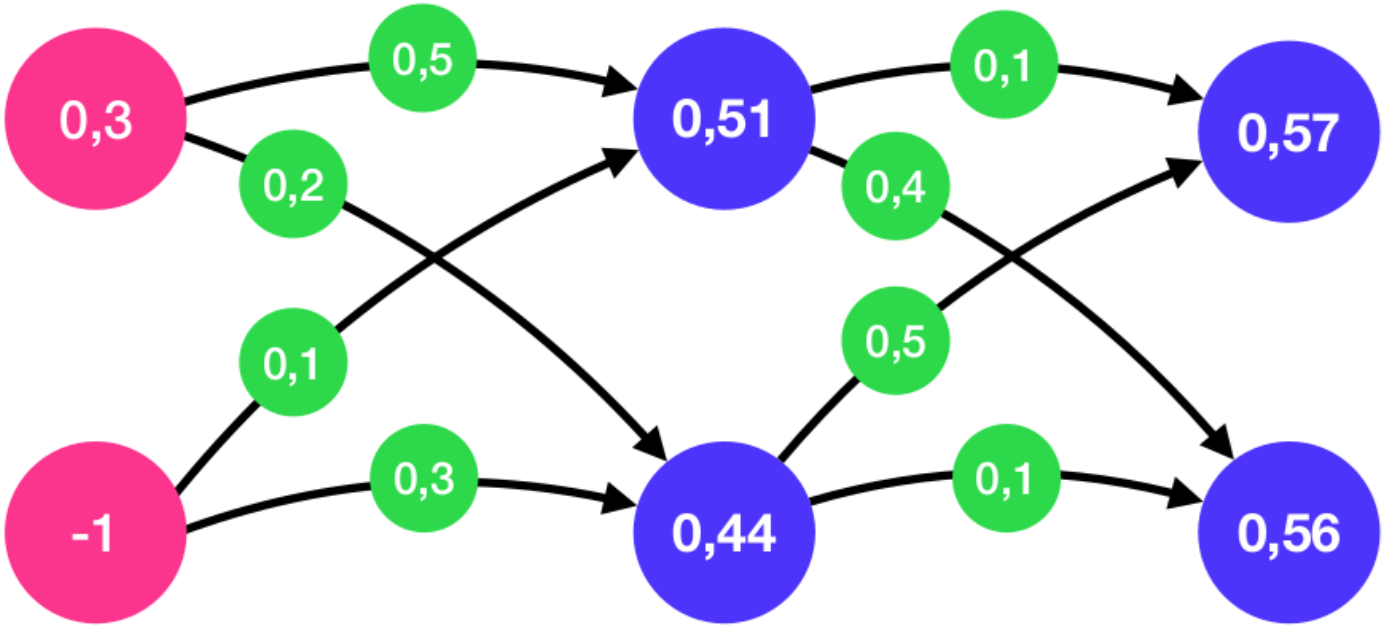
Rede neural



$$Z_1 = 0{,}5 \times 0{,}3 + 0{,}1 \times -1 = 0{,}049$$

$$A_1 = F_a(0{,}049) = 0{,}51$$

Rede neural



Treinando uma rede neural

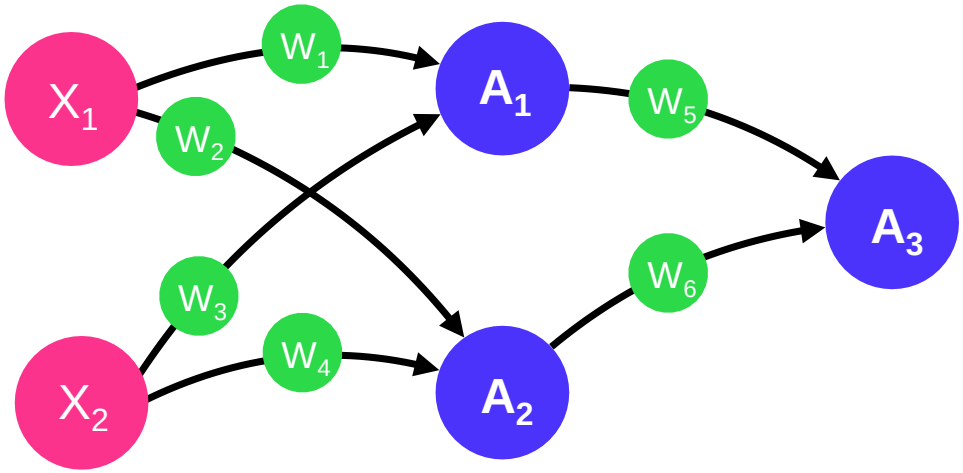
O algoritmo é semelhante ao que fizemos para apenas um neurônio

Iniciamos os pesos de forma aleatória

Entramos com os dados de treinamento e comparamos o resultado da rede com o resultado esperado

Ajustamos os pesos de acordo com o erro

Treinando uma Rede neural



O valor de A_3 é a saída da rede. Podemos compará-lo diretamente com o resultado dos dados de treinamento

E para A_1 e A_2? Quais são os valores esperados? Não temos esses dados

Treinando uma Rede neural

Para treinar uma rede com multicamadas usamos o algoritmo *backpropagation*

Nele, vamos definir uma função que mede o erro da saída da rede

Também conseguimos calcular através do gradiente da função de erro, em qual sentido devemos ajustar os pesos para que o erro diminua

Correção de Erros

04

Correção de erros

Calculamos o erro para um exemplo através da função:

$$e_k = d_k - y_k$$

Onde:

e - Sinal de erro
d - Saída desejada apresentada durante o treinamento
y - Saída real da rede

Aprendizado por Correção de erros

- O processo de aprendizado por correção de erros utiliza algoritmos para caminhar sobre a curva de erros, com o intuito de alcançar o menor valor de erro possível, o mínimo global
- Muitas vezes o algoritmo não alcança este mínimo global, atingindo o que chamamos de mínimo local. Caso este erro alcançado seja desfavorável, é necessário recomeçar o processo de aprendizagem

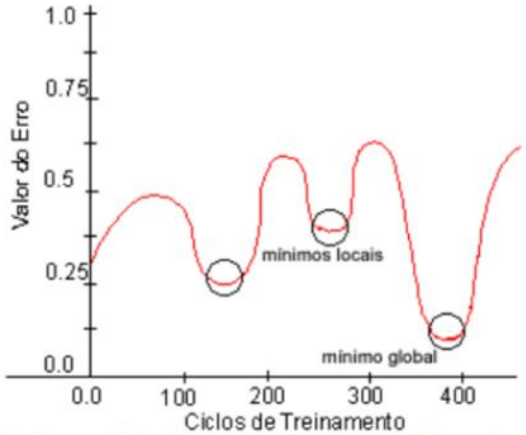


Gráfico de uma possível superfície de erro mostrando os mínimos locais e o mínimo global

Aprendizado por Correção de erros

Para correção do erro, os pesos devem ser ajustados de forma a aproximar a saída real à desejada

$$\Delta wi(n) = \eta\, e(n)\, xi(n)$$

Onde:

△wi(n) - Valor de ajuste a ser acrescido ao peso wi;

n - Taxa de aprendizagem (constante positiva);

e(n) - Valor do erro;

xi - Valor da entrada

Correção de erros

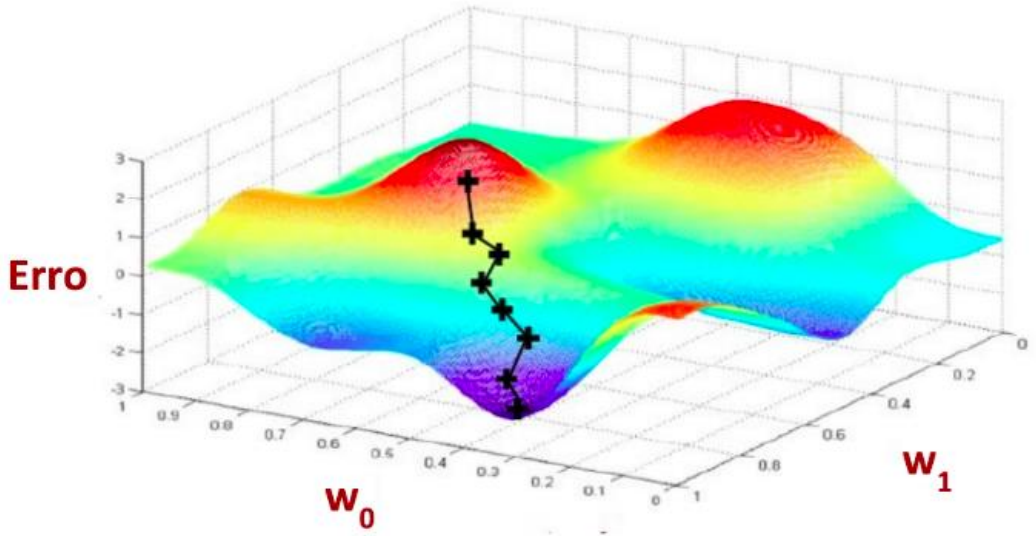
Encontrar o vetor (matriz) de pesos sinápticos (w*) que minimizem o erro (e) entre a saída da rede neural e a saída desejada

Backpropagation

05

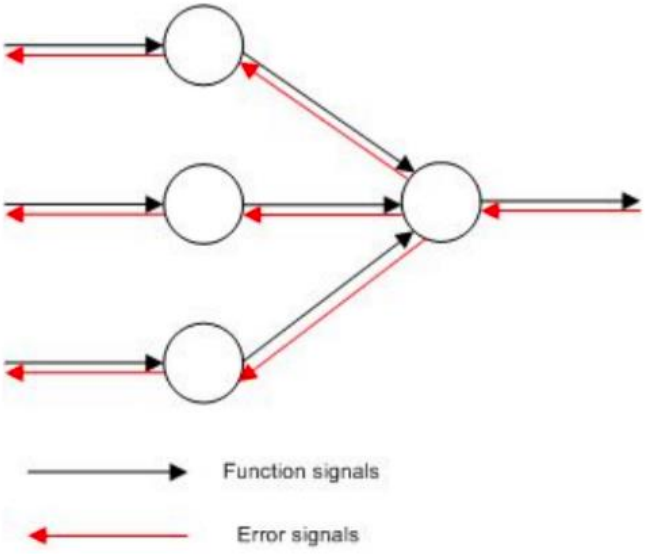
Método do Gradiente

O método do gradiente (ou método do máximo declive) é um método numérico usado em otimização. Para encontrar um mínimo (local) de uma função usa-se um esquema iterativo, onde em cada passo se toma a direção (negativa) do gradiente, que corresponde à direção de declive máximo.



Backpropagation

Sinal de erro se origina em um neurônio de saída e se propaga para trás (camada por camada) através da rede



Correção dos pesos - Regra delta

$$\Delta w_{ji}(n) = \eta * \delta j(n) * y_i(n)$$

- $\Delta w_{ji}(n)$: **Correção de peso**
- η: **Taxa de aprendizagem**
- $\delta j(n)$: **Gradiente local**
- $y_i(n)$: **Sinal de entrada do neurônio *j***

Gradiente Local

Camada de saída

$$\delta_j(n) = e_j * \varphi'(v_j(n))$$

- Gradiente Local
- Sinal de erro na saída de *j*
- Derivada da função de ativação no neurônio j

Camada oculta

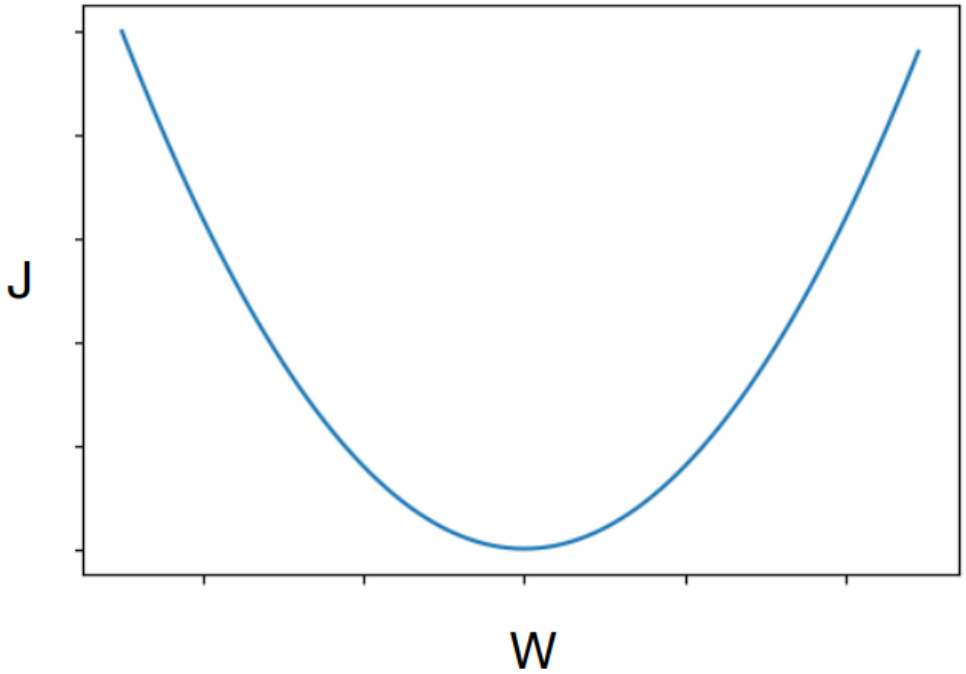
$$\delta_j(n) = \varphi'\left(v_j(n)\right) \sum_k \delta_k(n) w_{kj}(n)$$

- Gradiente Local
- Derivada associada ao neurônio j
- Soma ponderada dos gradientes locais da camada seguinte k

Treinando uma Rede Neural

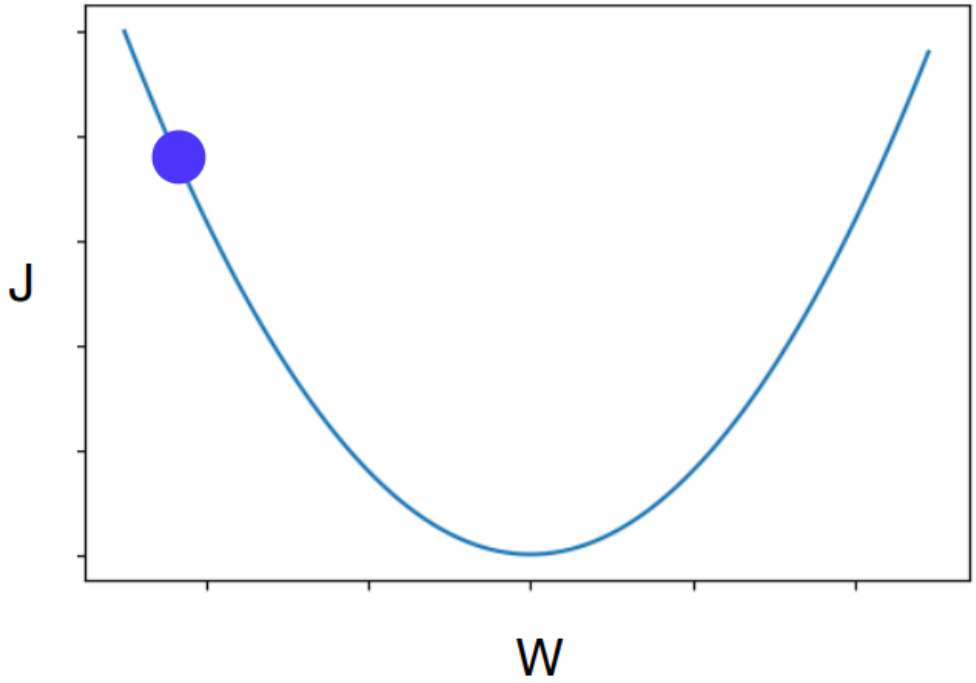
A função de erro tem uma forma como essa:

$$w_i \leftarrow w_i - \eta \frac{\partial J}{\partial w_i}$$



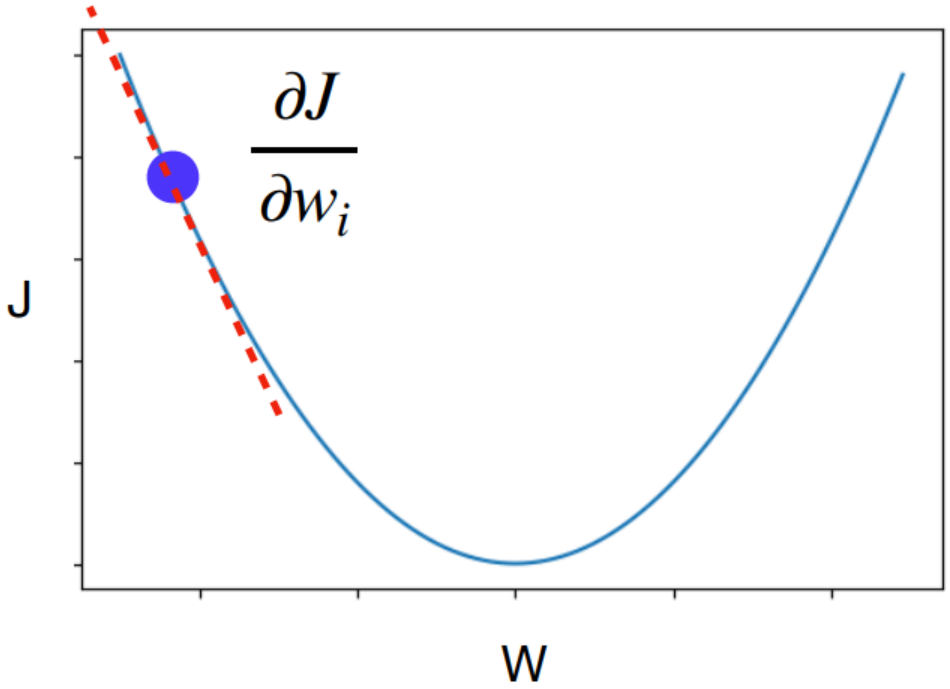
Treinando uma Rede Neural

Inicializamos o valor de W aleatoriamente



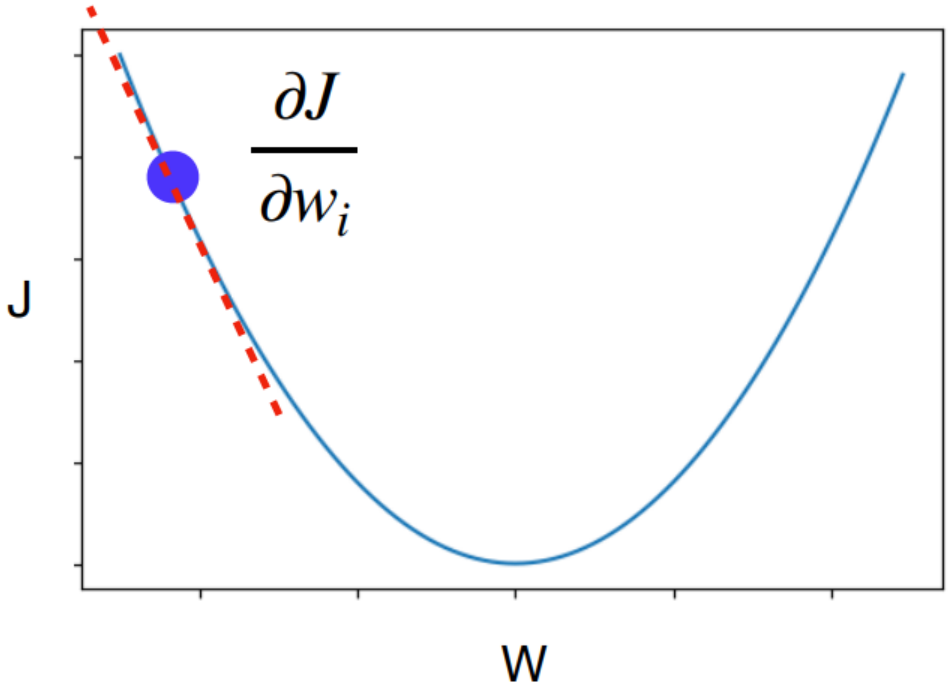
Treinando uma Rede Neural

Inicializamos o valor de W aleatoriamente



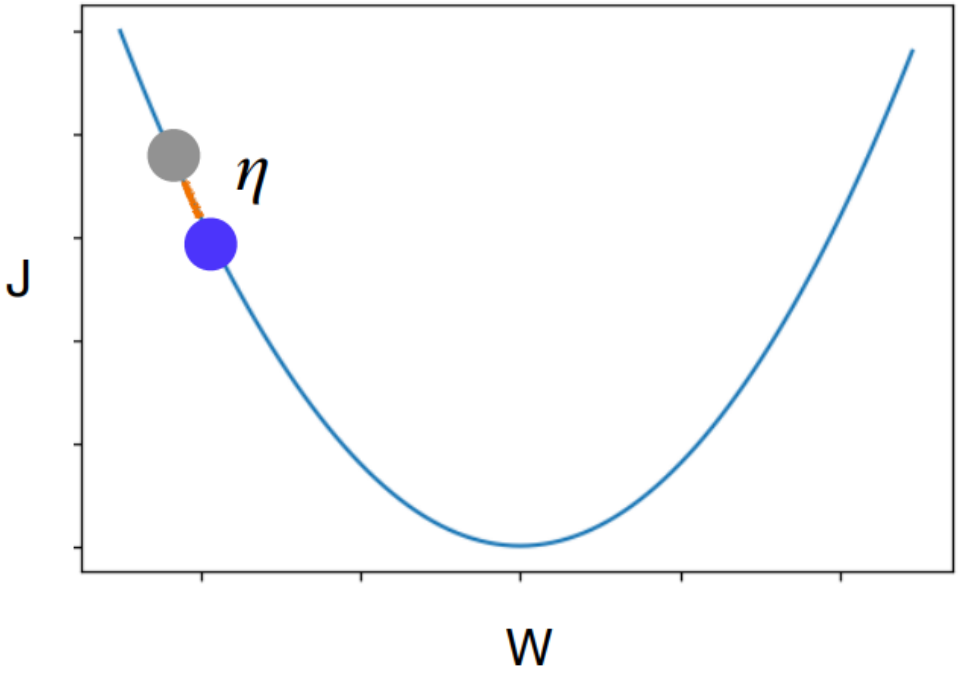
Treinando uma Rede Neural

A derivada nos fornece a inclinação do ponto



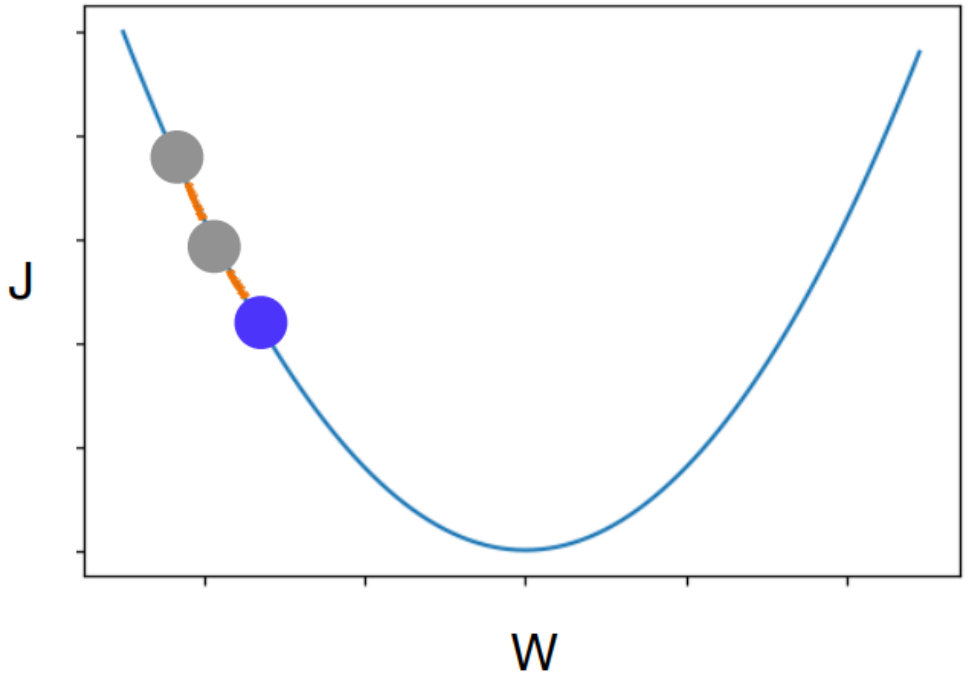
Treinando uma Rede Neural

A taxa de aprendizagem determina o tamanho do passo de atualização



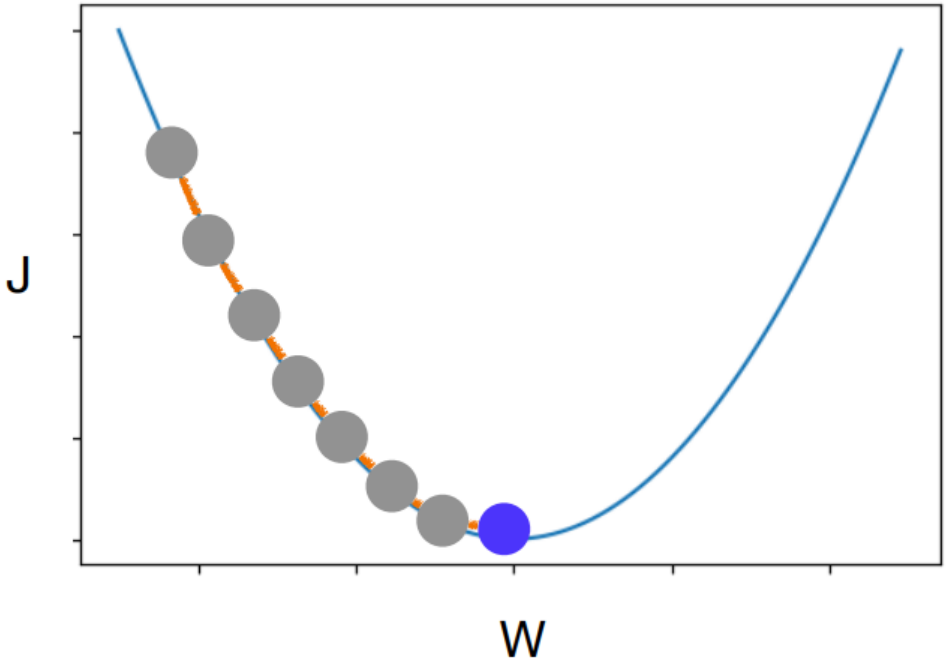
Treinando uma Rede Neural

Repetimos atualização de W múltiplas vezes

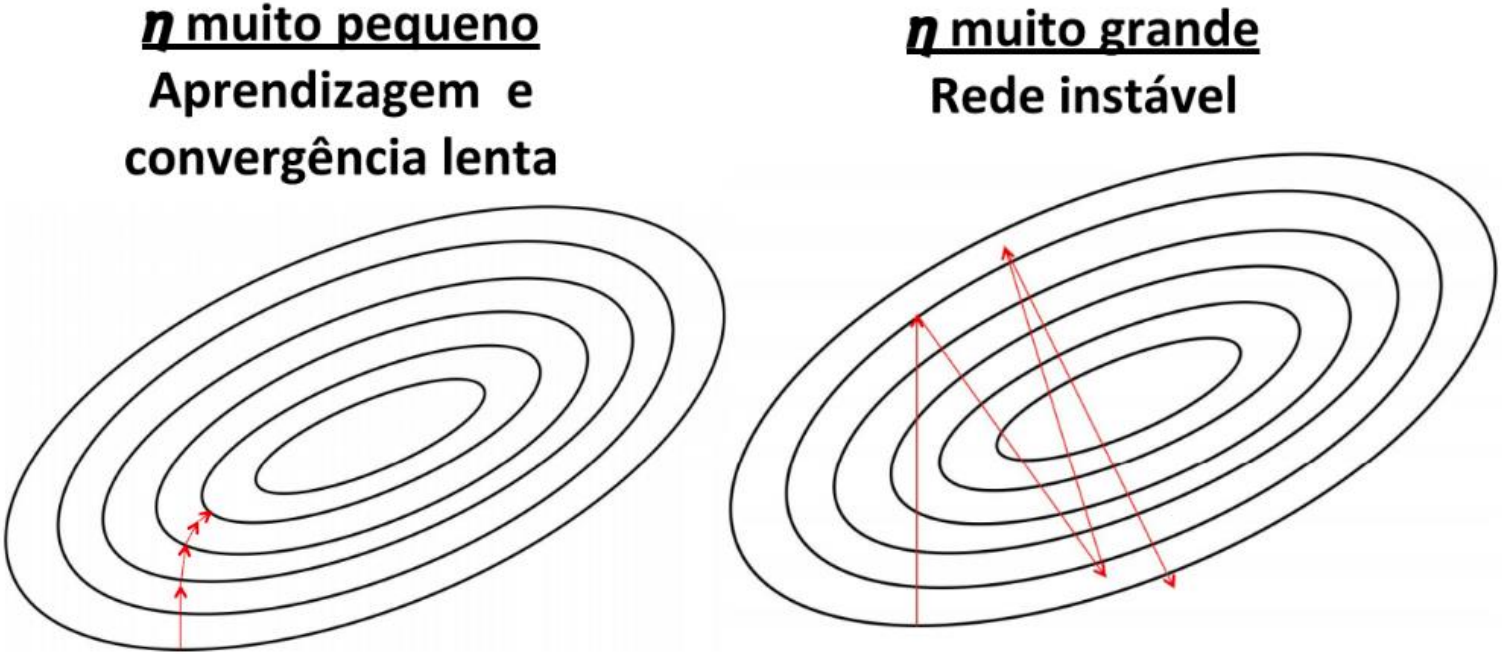


Treinando uma Rede Neural

Repetimos atualização de W múltiplas vezes

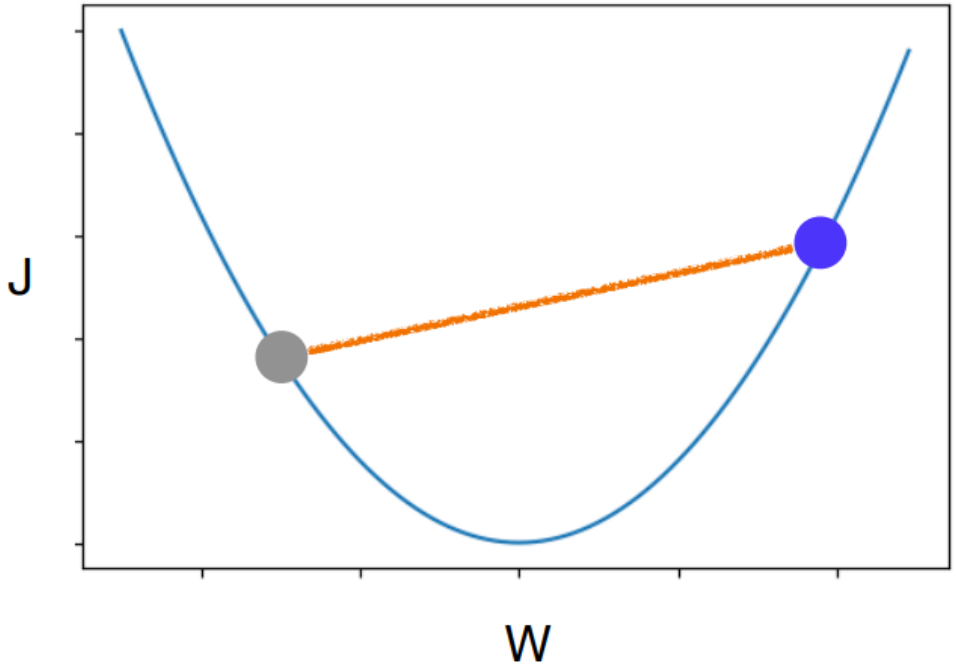


Taxa de aprendizagem



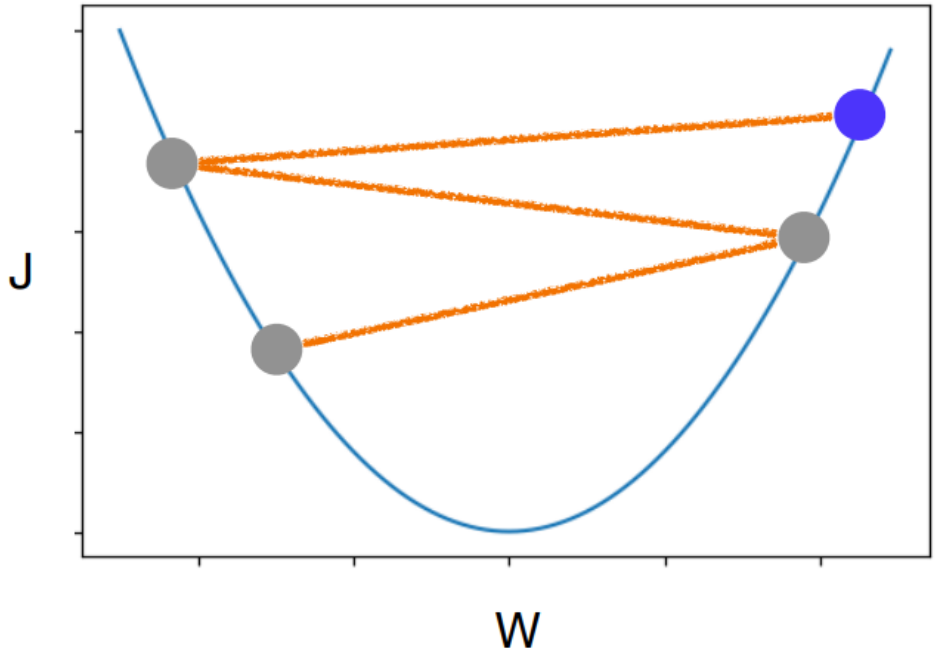
Treinando uma Rede Neural

O passo de atualização foi tão grande que o erro aumentou



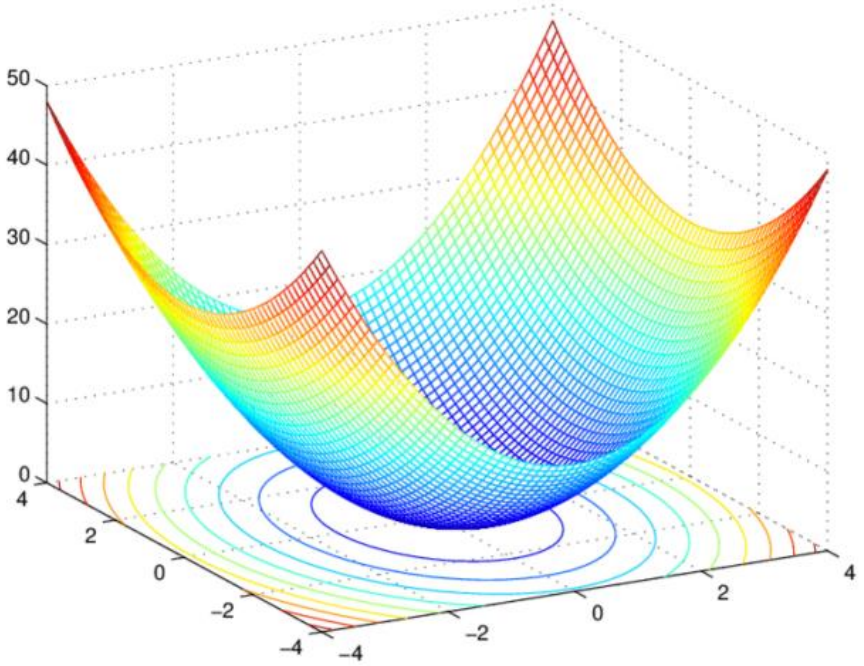
Treinando uma Rede Neural

O erro continua aumentando



Treinando uma Rede Neural

Na prática a função de J é calculada usando vários Ws



Modos de treinamento: Estocástico, por lote e mini-lote

06

Modos de treinamento

Baseado na forma de atualização dos pesos, temos 3 modos de treinamento:

- Estocástico - Ajuste de pesos é realizado após a apresentação de cada exemplo
- Por lote (batch) - Ajuste de pesos é realizado após a apresentação de todo os exemplos à rede (fim da época)
- Mini-lote (mini-batch) - Ajuste de pesos é realizado após a apresentação de um subconjunto de exemplos

Estocástico

Vantagens:

- Ajuda a escapar de mínimos locais
- Mais simples de implementar
- Pode tirar vantagens de dados (exemplos) redundantes

Desvantagens:

- Estimar o erro baseado em um único exemplo não é uma boa aproximação do erro real
- Treinamento muito lento
- Mais difícil provar teoricamente que o algoritmo converge

Por lote (batch)

Vantagens:

- Estimativa precisa do gradiente
- Convergência mais rápida sob condições simples
- Mais fácil de paralelizar

Desvantagens:

- Pode ficar preso em mínimos locais

Por mini-lote (mini-batch)

Bom balanço entre o modo estocástico e o modo por lote:

- Convergência mais rápida (modo por lote)
- Evita mínimos locais

Critérios de parada

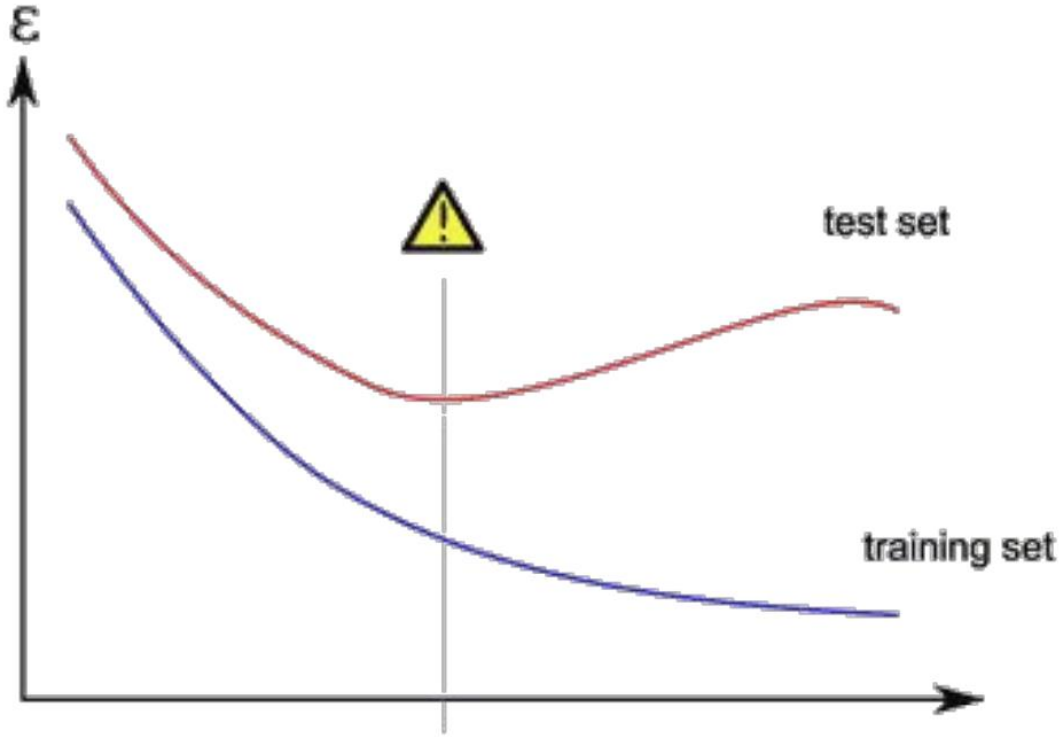
07

Critérios de parada

Existem alguns critérios razoáveis para a convergência:

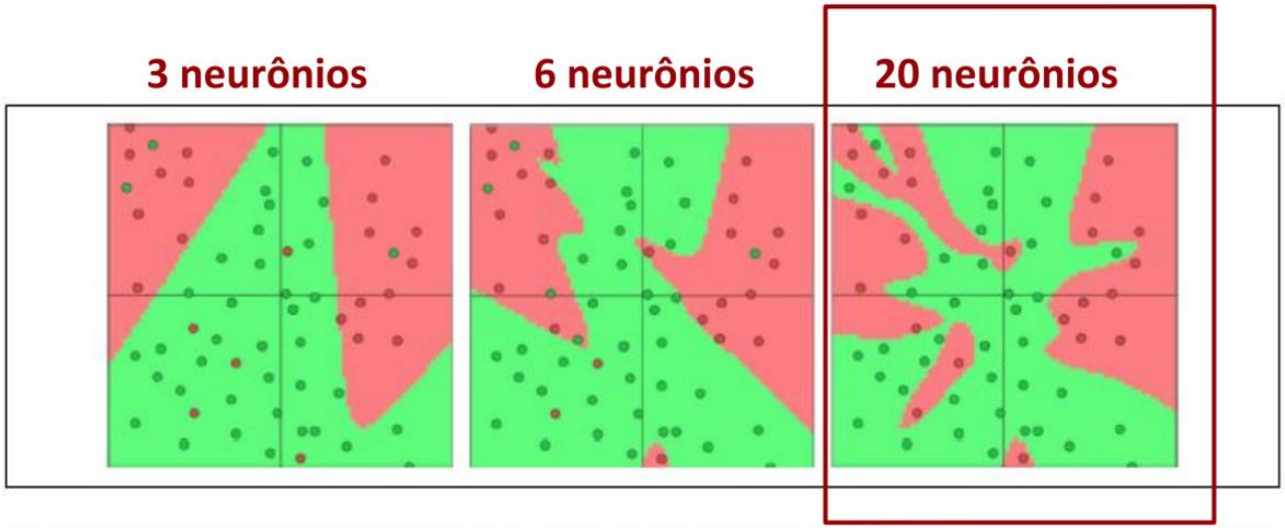
- Vetor gradiente alcançar um limiar suficiente pequeno
- Taxa de variação do erro muito pequena entre as épocas (ex: menor que 1%)
- Rede apresenta um bom desempenho de generalização, ou seja, funciona bem com um outro conjunto de exemplos (conjunto de validação)

Overfitting



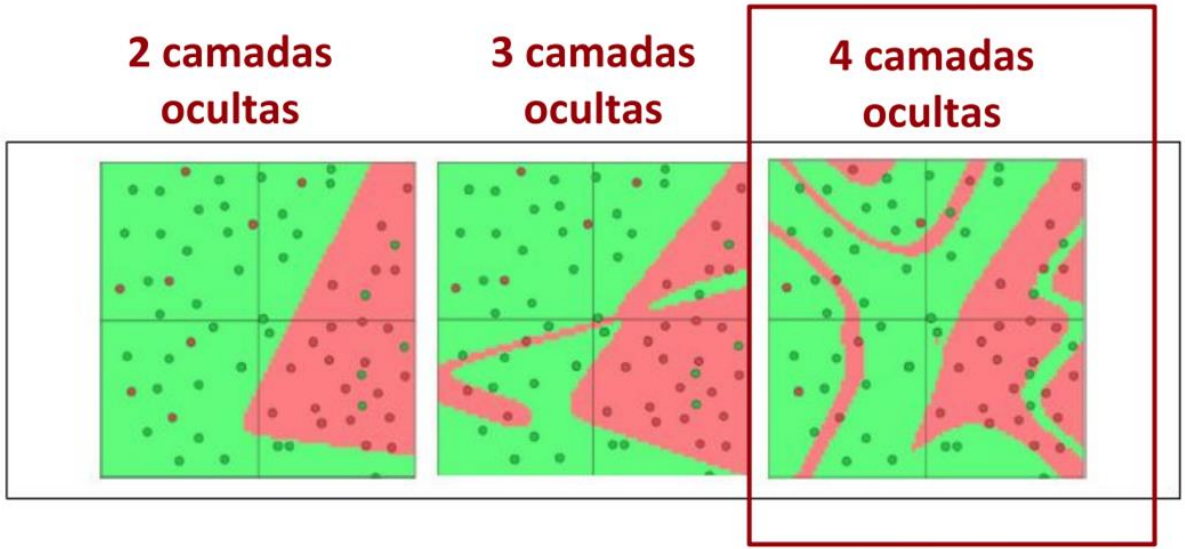
Overfitting

- Aumento no número de conexões implica em maior propensão a **overfitting.**
- Ex: RNA formada por 1 camada oculta contendo:



Overfitting

- Aumento no número de camadas também aumenta a propensão a **Overfitting**.



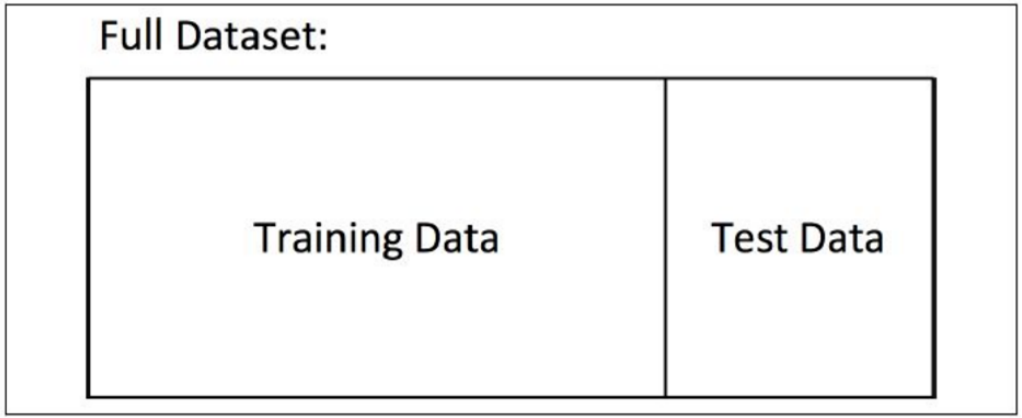
Overfitting

- Overfitting é um problema comum em DL.

- DL procura resolver **problemas muito complexos** com **modelos complexos**
  - Necessário tomar medidas para prevenir o **overfitting**.
  - Veremos algumas dessas medidas

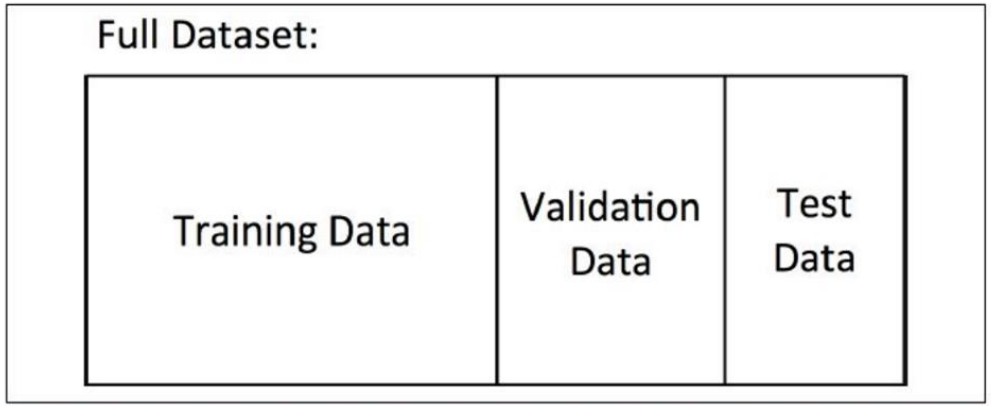
Prevenção de Overfitting

- **Medida 1: Não avalie seu modelo com os dados que você usou para treinar.**
  - Divida seu conjunto em conjunto de treinamento e conjunto de testes.



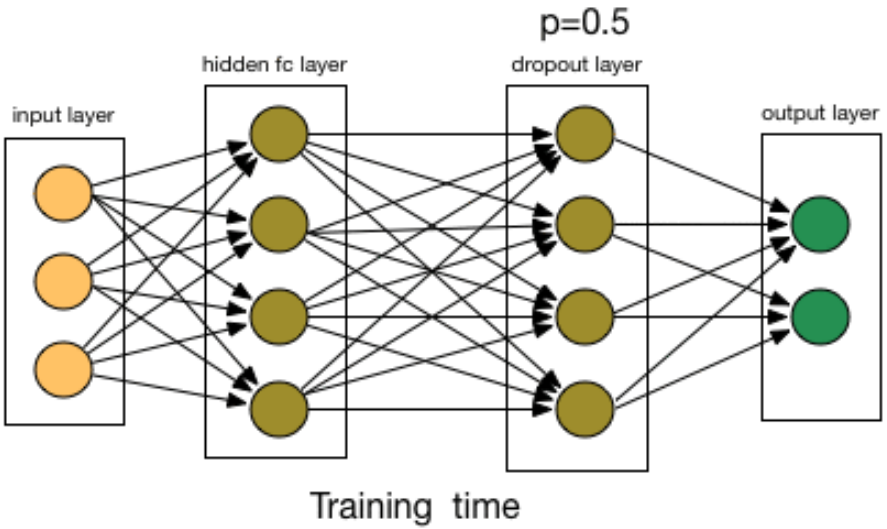
Prevenção de Overfitting

- **Medida 2: Crie também um conjunto de validação.**
  - Use-o para avaliar o treinamento de época em época;
  - Se o desempenho continuar **melhorando no treinamento** mas **piorando no conjunto de validação**, é hora de parar.
    - **Você está em OVERFITTING.**



Dropouts

- Durante o treinamento, mantém um neurônio ativo com uma probabilidade *p*.



Dropout

- **Uma das estratégias preferidas, em DL, para prevenir overfitting.**
- Força a rede a ter uma boa acurácia mesmo na ausência de certas informações.
- Evita que a rede se torne muito dependente de um (ou um conjunto de) neurônios.

Referências

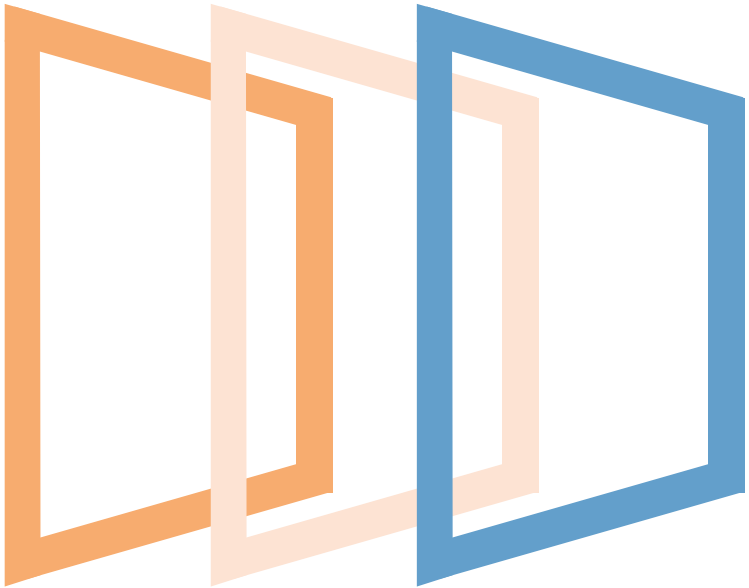
Inteligência Artificial - Aulas de Inteligência Artificial (google.com)

ARIA - YouTube

Modelos Clássicos e Redes Neurais

Thaís Ratis

Inteligência Artificial Brasil, 20.08.2025



minsait

An Indra company