Introdução a Redes Neurais

Deep Learning com Tensorflow / Keras

Estrutura Geral da Aula

Parte teórica:

- Estrutura de uma Rede Neural
- O Perceptron
- Funções de ativação
- Como uma rede neural aprende?
- Hiper-parâmetros

Parte prática:

- Carregando e preparando DataSet (usando Pandas e Numpy)
- Rede Neural para classificar flores de íris
- Ajustando a rede para uma melhor performance

Estrutura Geral da Aula

Parte teórica:

- **Estrutura de uma Rede Neural**
- O Perceptron
- Funções de ativação
- Como uma rede neural aprende?
- Hiper-parâmetros

Parte prática:

- Carregando e preparando DataSet (usando Pandas e Numpy)
- Rede Neural para classificar flores de íris
- Ajustando a rede para uma melhor performance

Estrutura de uma Rede Neural

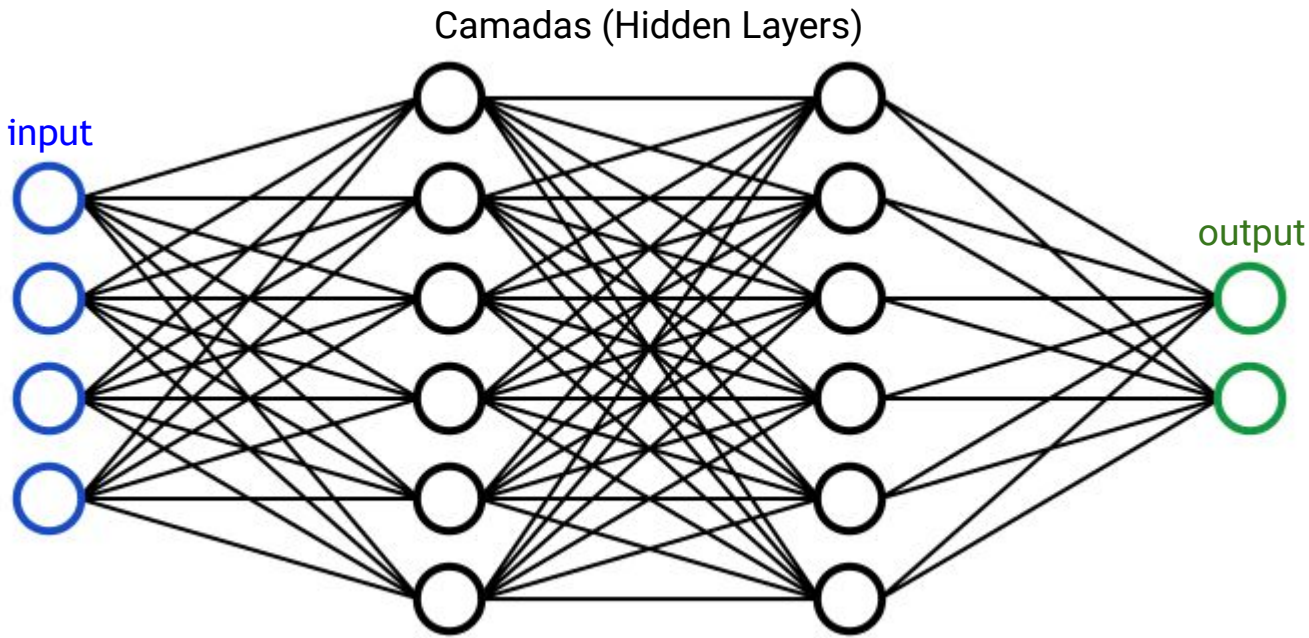


Imagem: https://medium.com/swlh/an-introduction-to-neural-networks-de70cb4305f9

Estrutura de uma Rede Neural

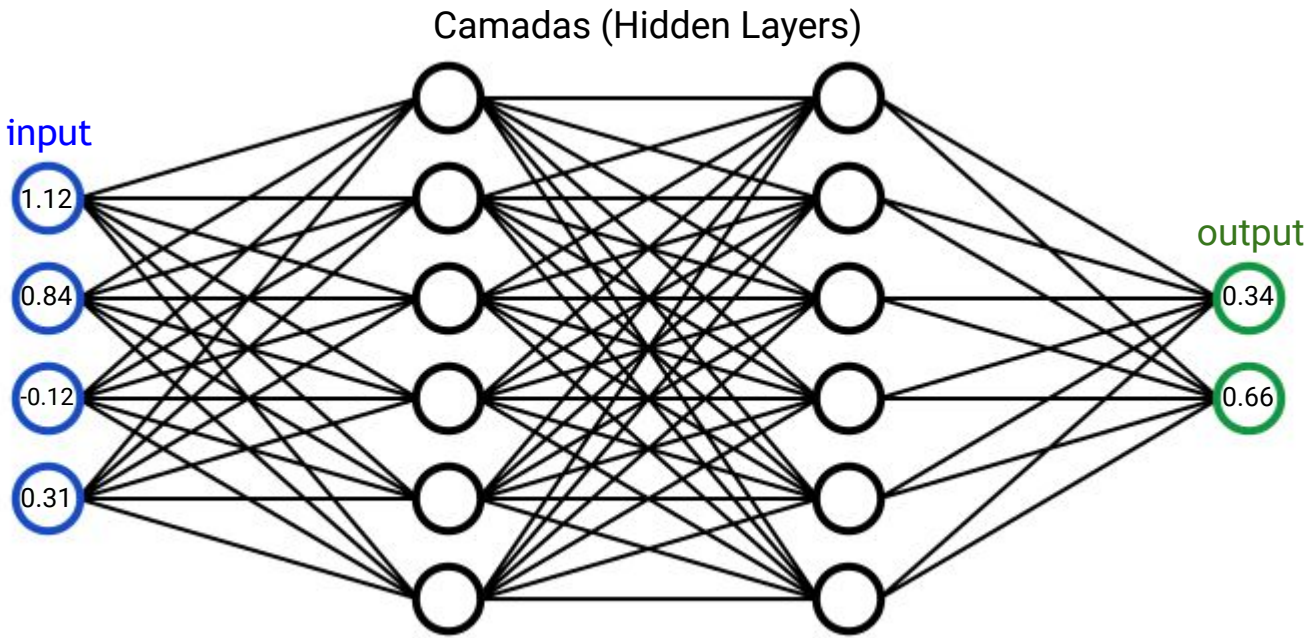


Imagem: https://medium.com/swlh/an-introduction-to-neural-networks-de70cb4305f9

Estrutura de uma Rede Neural

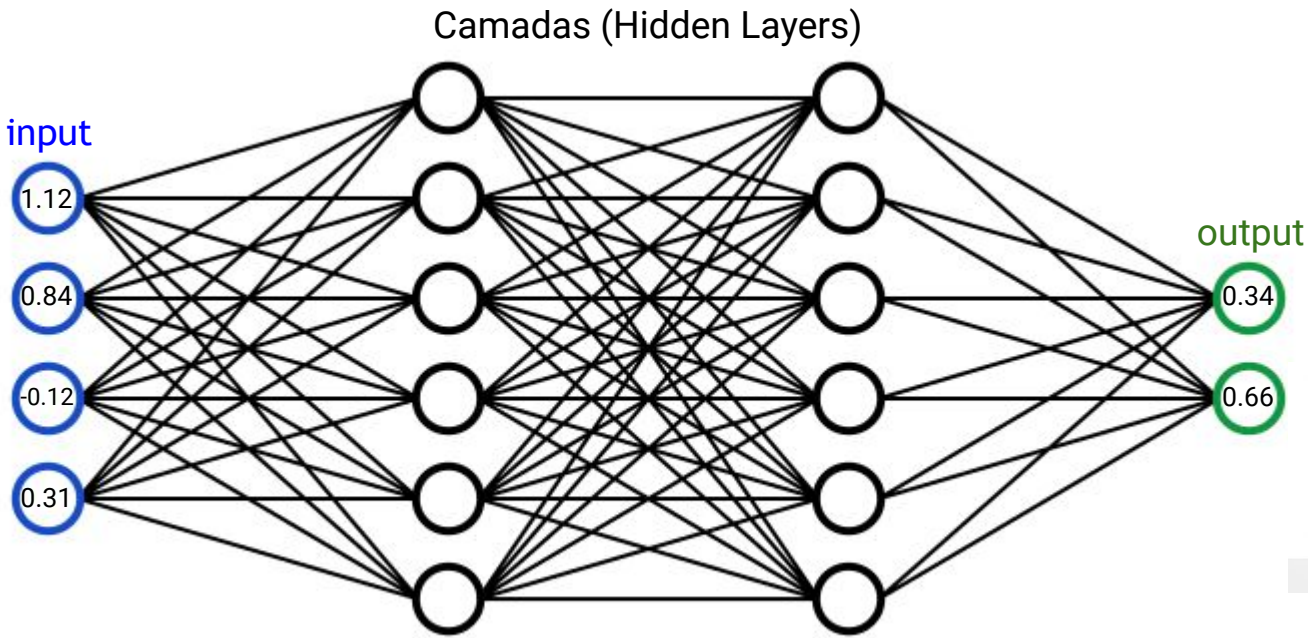


Imagem: https://medium.com/swlh/an-introduction-to-neural-networks-de70cb4305f9

Estrutura Geral da Aula

Parte teórica:

- Estrutura de uma Rede Neural
- **O Perceptron**
- Funções de ativação
- Como uma rede neural aprende?
- Hiper-parâmetros

Parte prática:

- Carregando e preparando DataSet (usando Pandas e Numpy)
- Rede Neural para classificar flores de íris
- Ajustando a rede para uma melhor performance

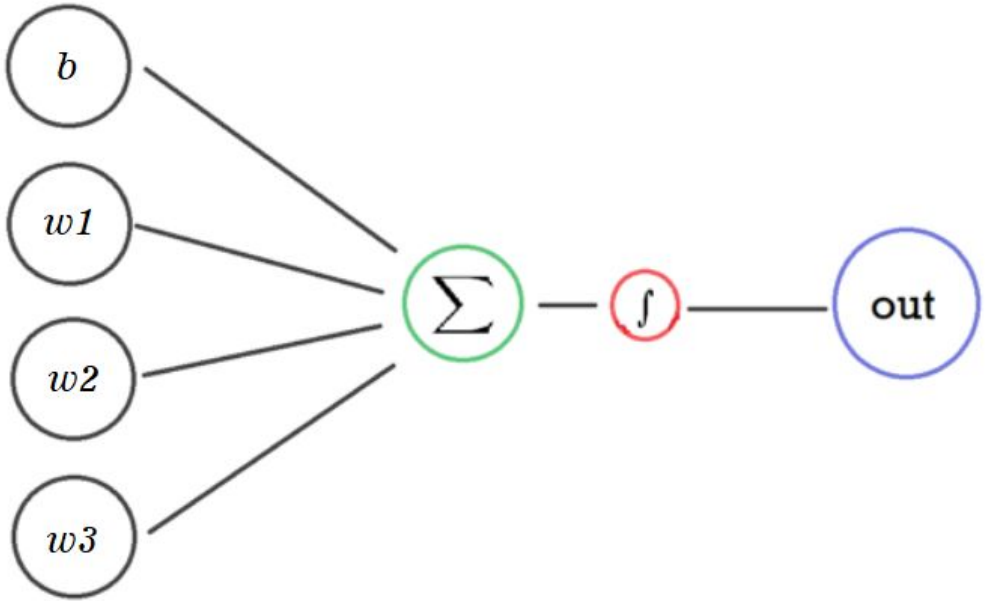
O que é um Perceptron?

No contexto das redes neurais artificiais, um neurônio é uma unidade de Perceptron.

Um perceptron é, na prática, uma forma matemática/computacional de modelar uma classificação simples.

O que é um Perceptron?

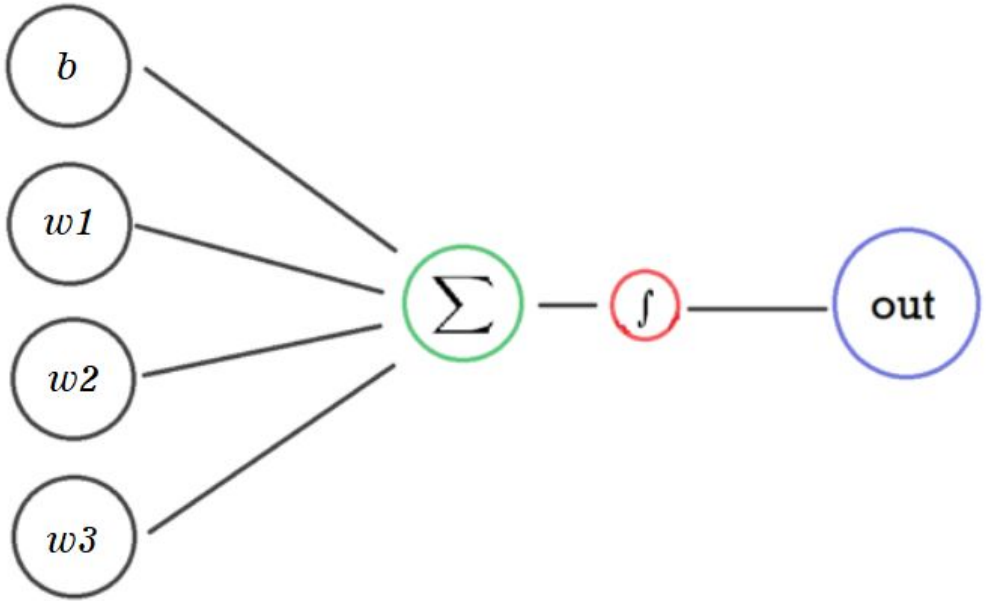
$$out = g(\, b + \sum_{i=1}^{m} w_i x_i \,)$$



O que é um Perceptron?

$$out = g(\, b + \sum_{i=1}^{m} w_i x_i \,)$$

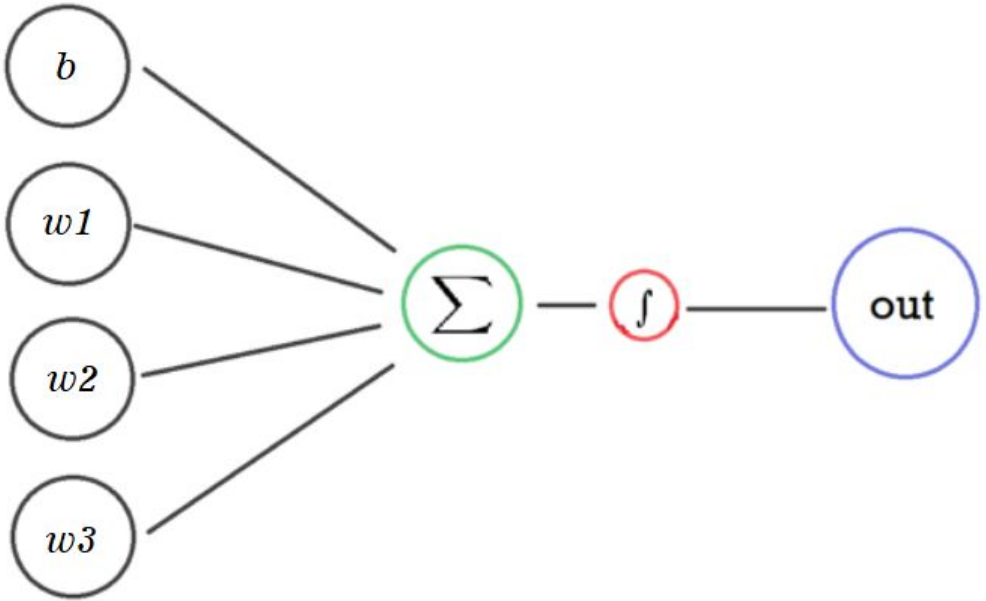
- **X** é o vetor de input



O que é um Perceptron?

$$out = g(b + \sum_{i=1}^{m} w_i x_i)$$

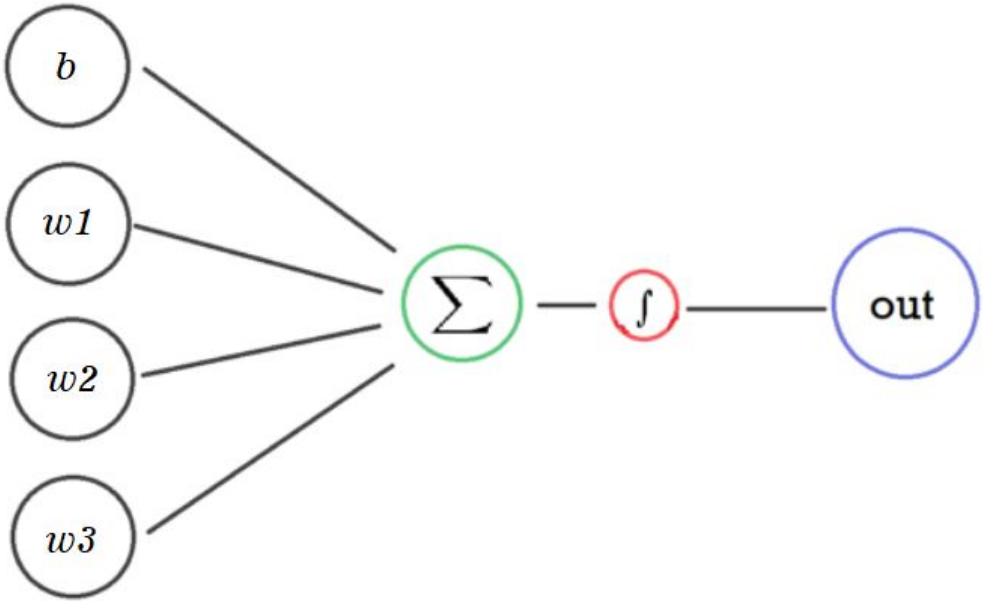
- **X** é o vetor de input
- **W** é o vetor de pesos



O que é um Perceptron?

$$out = g(\, b + \sum_{i=1}^{m} w_i x_i \,)$$

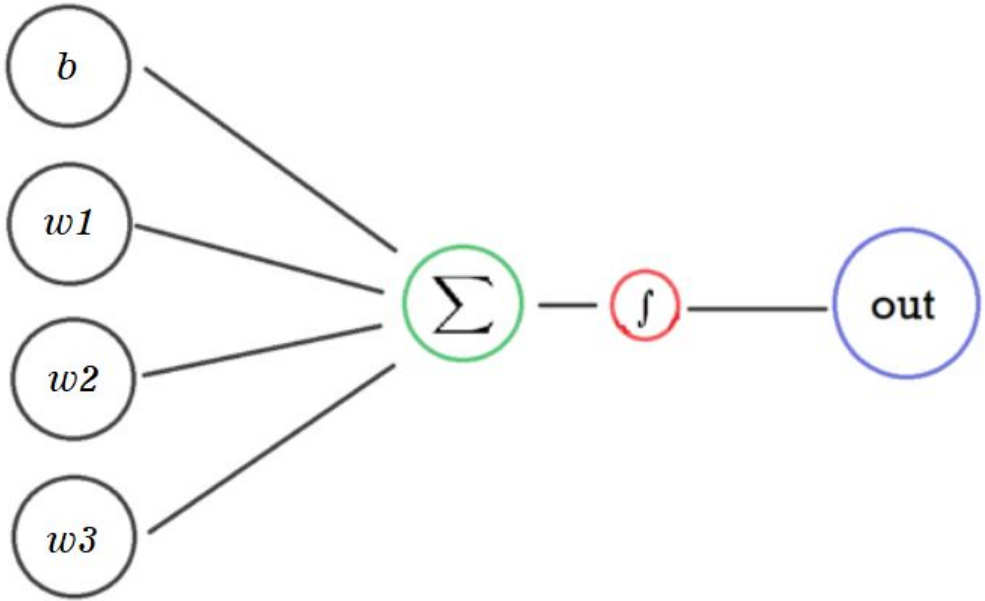
- **X** é o vetor de input
- **W** é o vetor de pesos
- **m** é a quantidade de elementos de X (que é igual a de W)



O que é um Perceptron?

$$out = g(\, b + \sum_{i=1}^{m} w_i x_i \,)$$

- **X** é o vetor de input
- **W** é o vetor de pesos
- **m** é a quantidade de elementos de X (que é igual a de W)
- **b** é o viés

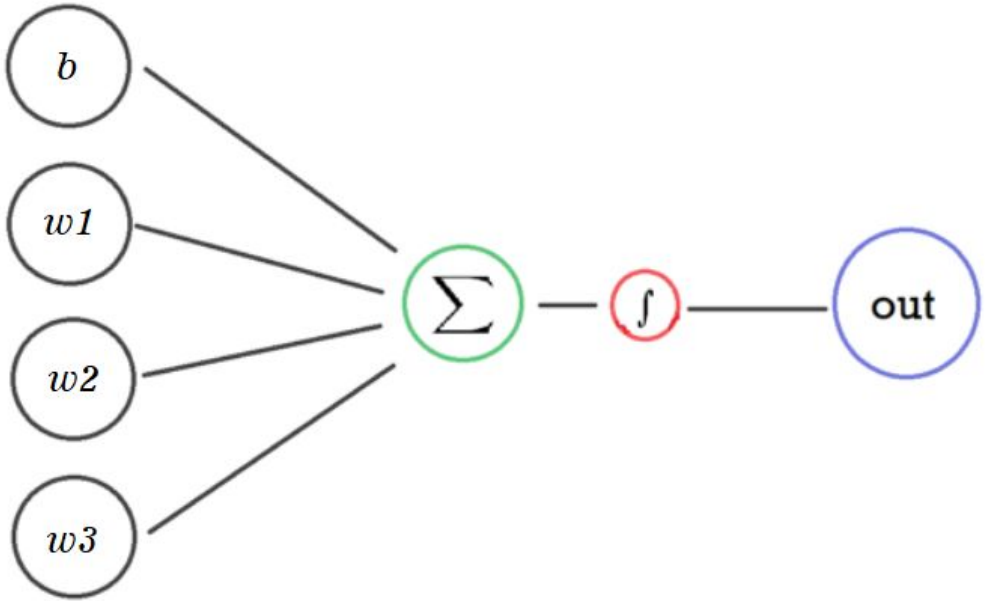


O que é um Perceptron?

$$out = g(\, b + \sum_{i=1}^{m} w_i x_i \,)$$

- **X** é o vetor de input
- **W** é o vetor de pesos
- **m** é a quantidade de elementos de X (que é igual a de W)
- **b** é o viés

- **g** é a função de ativação

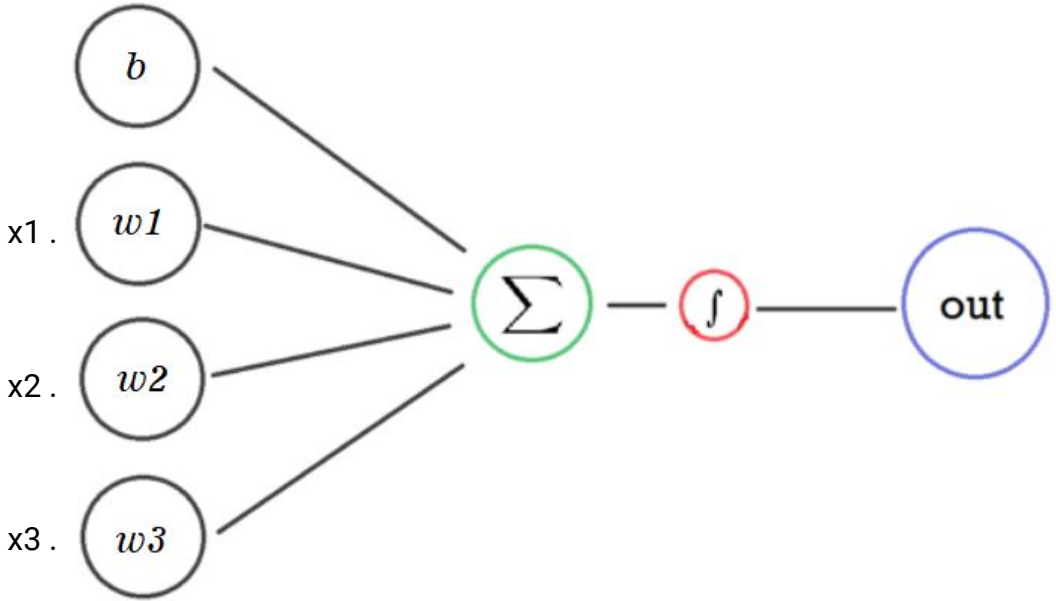


O que é um Perceptron?

$$out = g(\, b + \sum_{i=1}^{m} w_i x_i \,)$$

- **X** é o vetor de input
- **W** é o vetor de pesos
- **m** é a quantidade de elementos de X (que é igual a de W)
- **b** é o viés

- **g** é a função de ativação

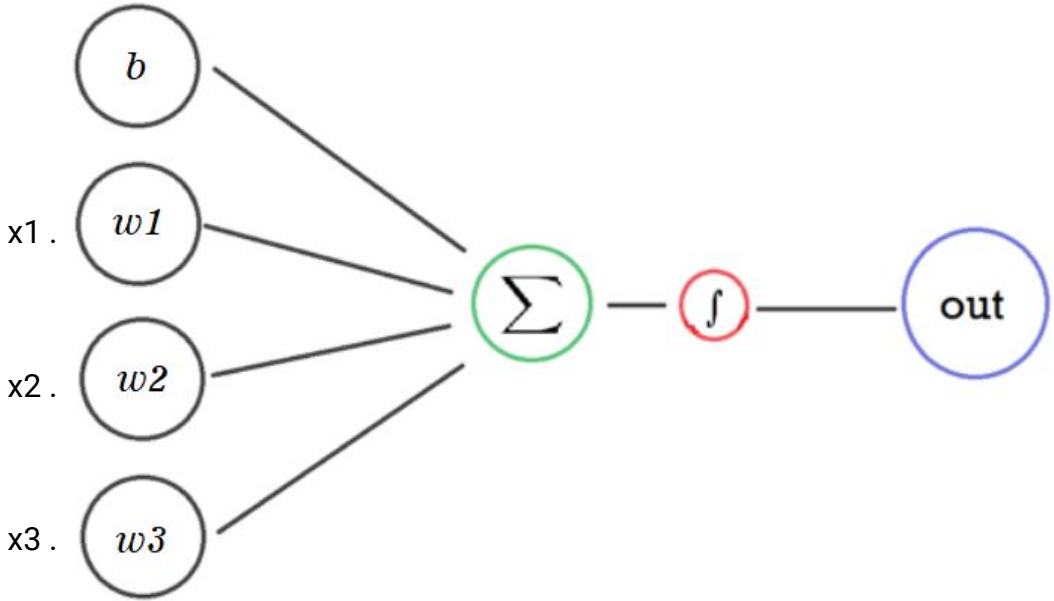


O que é um Perceptron?

$$out = g(\, b + \sum_{i=1}^{m} w_i x_i \,)$$

- **X** é o vetor de input
- **W** é o vetor de pesos
- **m** é a quantidade de elementos de X (que é igual a de W)
- **b** é o viés

- **g é a função de ativação (???)**

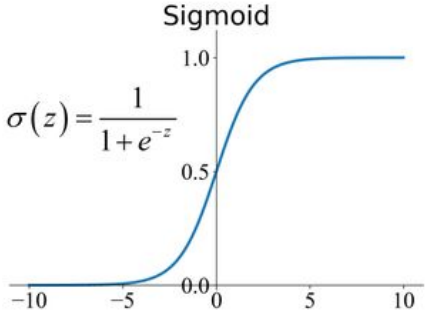


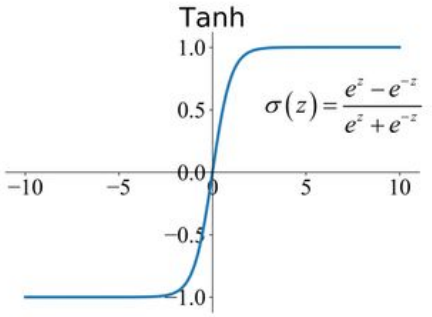
Funções de ativação

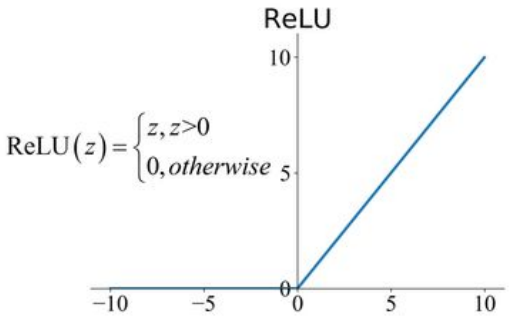
São funções unárias

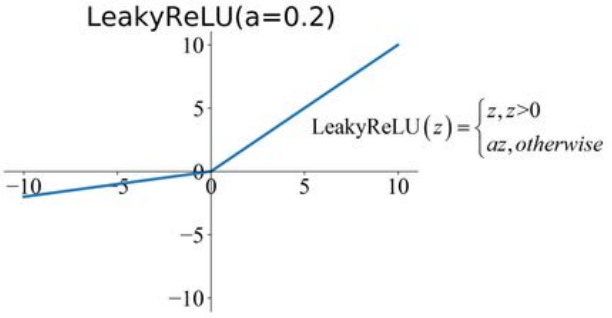
Servem para introduzir **não-linearidade**

Caso não existissem, não conseguiríamos adaptar nossos perceptrons nas redes neurais a estruturas de dados mais complexas (não-lineares).









Estrutura Geral da Aula

Parte teórica:

- Estrutura de uma Rede Neural
- O Perceptron
- Funções de ativação
- **Como uma rede neural aprende?**
- Hiper-parâmetros

Parte prática:

- Carregando e preparando DataSet (usando Pandas e Numpy)
- Rede Neural para classificar flores de íris
- Ajustando a rede para uma melhor performance

Como uma rede neural aprende?

Uma Rede Neural aprende focando na minimização da **função de perda**.

Como uma rede neural aprende?

Uma Rede Neural aprende focando na minimização da **função de perda**.

Uma **função de perda**, por sua vez, é uma forma matemática de relacionar a predição do modelo com o resultado real, através de uma função.

Como uma rede neural aprende?

Uma Rede Neural aprende focando na minimização da **função de perda**.

Uma **função de perda**, por sua vez, é uma forma matemática de relacionar a predição do modelo com o resultado real, através de uma função.

Para minimizar essa função, ajustamos os pesos dos neurônios
do modelo, para produzir resultados diferentes.

Como uma rede neural aprende?

Uma Rede Neural aprende focando na minimização da **função de perda**.

Uma **função de perda**, por sua vez, é uma forma matemática de relacionar a predição do modelo com o resultado real, através de uma função.

Para minimizar essa função, ajustamos os pesos dos neurônios do modelo, para produzir resultados diferentes.

Para ajustar os pesos utilizamos uma estratégia chamada **backpropagation**, adaptando os valores da rede a cada conjunto de dados de treino.

Como uma rede neural aprende?

Uma Rede Neural aprende focando na minimização da **função de perda**.

Uma **função de perda**, por sua vez, é uma forma matemática de relacionar a predição do modelo com o resultado real, através de uma função.

Para minimizar essa função, ajustamos os pesos dos neurônios do modelo, para produzir resultados diferentes.

Para ajustar os pesos utilizamos uma estratégia chamada **backpropagation**, adaptando os valores da rede a cada conjunto de dados de treino.

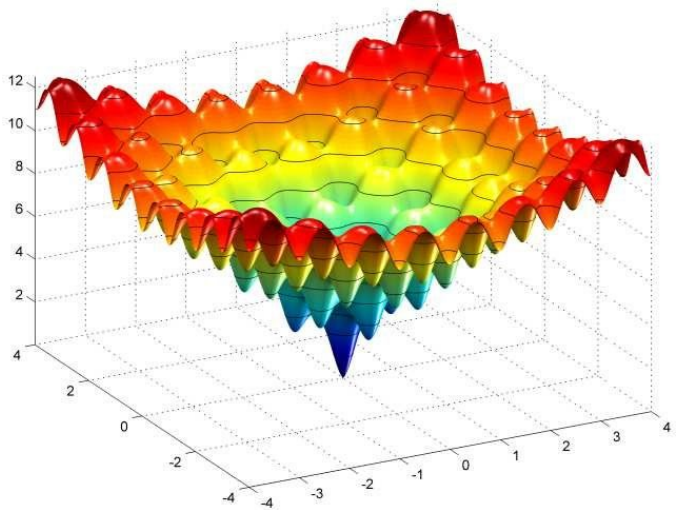
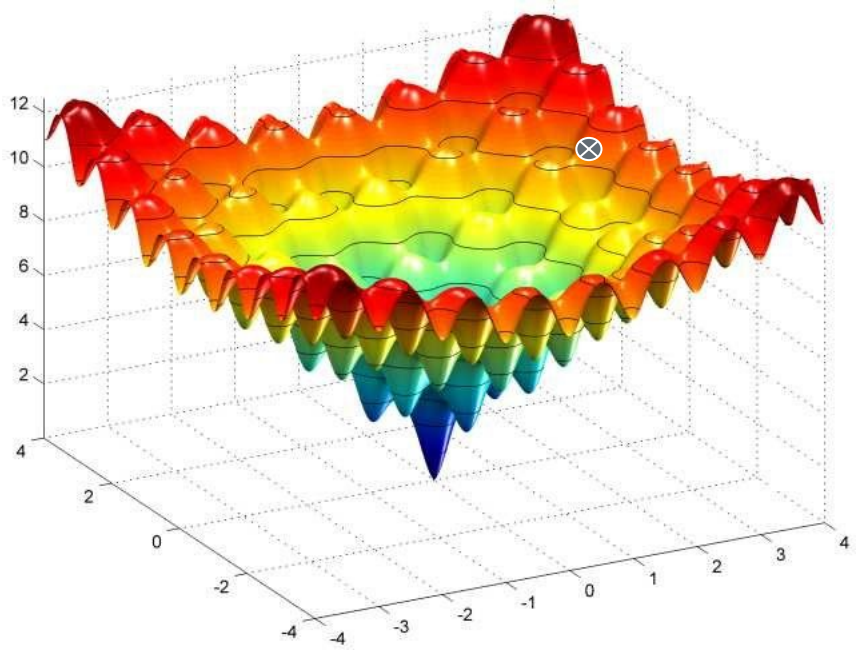


Imagem: https://www.researchgate.net/figure/Rastrigin-Function-Figure-4-Ackley-Function_fig5_324084428

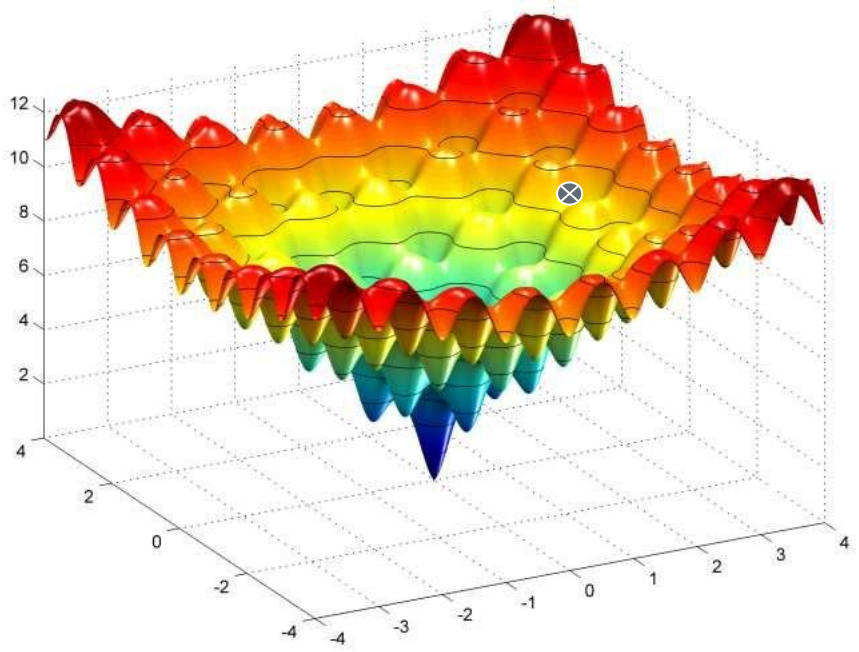
Como uma rede neural aprende?

Essa adaptação dos pesos vai nos fazer “caminhar” pela função de perda.



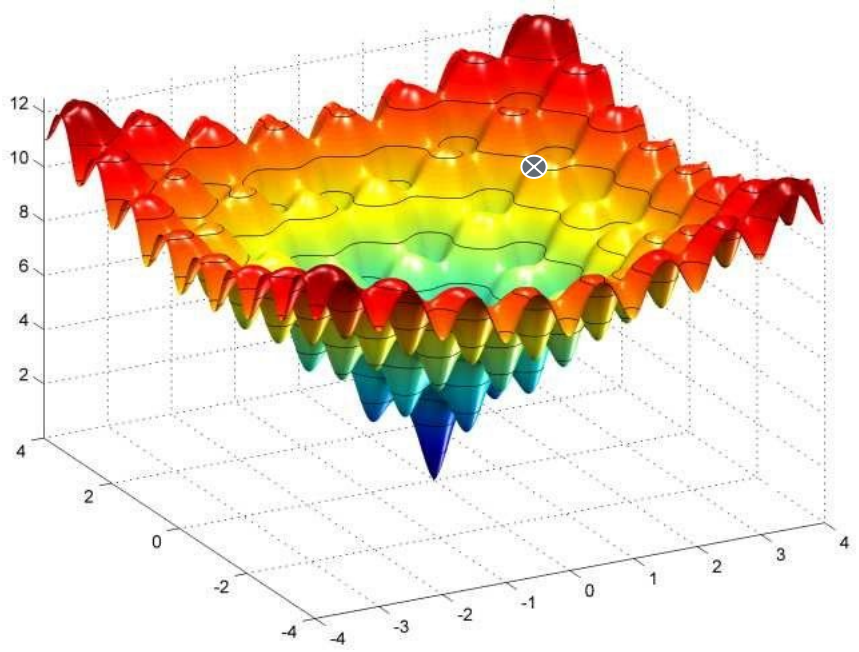
Como uma rede neural aprende?

Essa adaptação dos pesos vai nos fazer “caminhar” pela função de perda.



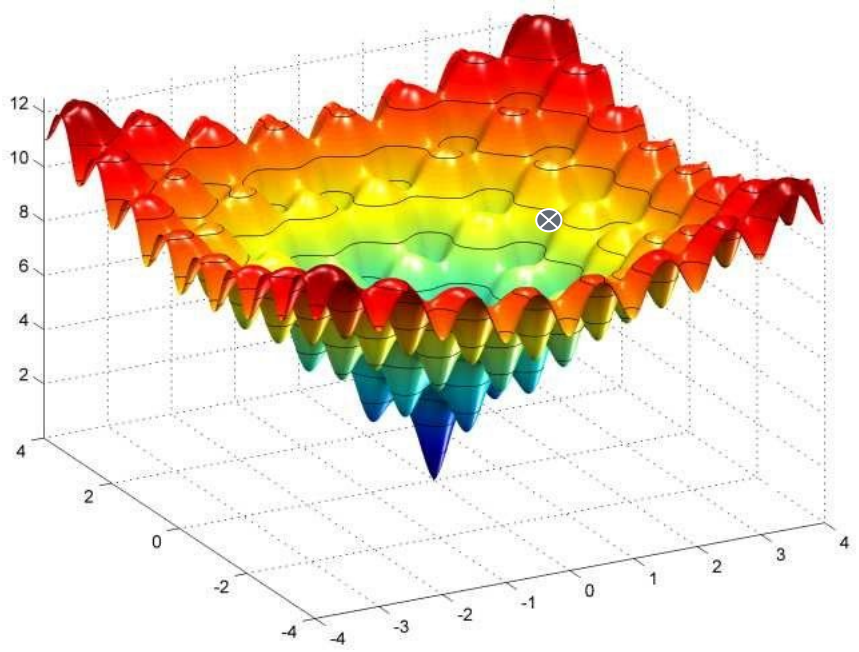
Como uma rede neural aprende?

Essa adaptação dos pesos vai nos fazer “caminhar” pela função de perda.



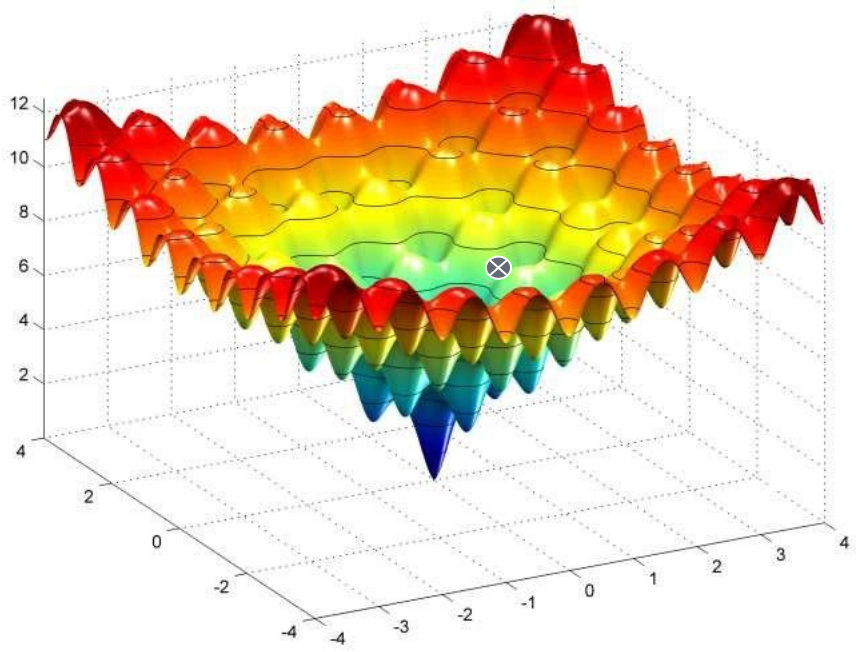
Como uma rede neural aprende?

Essa adaptação dos pesos vai nos fazer “caminhar” pela função de perda.



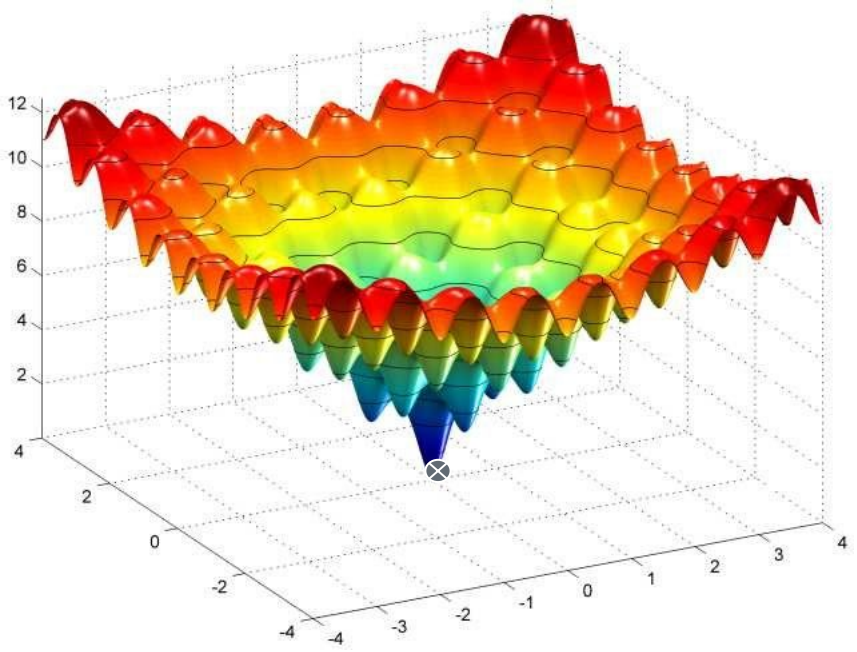
Como uma rede neural aprende?

Essa adaptação dos pesos vai nos fazer “caminhar” pela função de perda.



Como uma rede neural aprende?

Essa adaptação dos pesos vai nos fazer “caminhar” pela função de perda.



Como uma rede neural aprende?

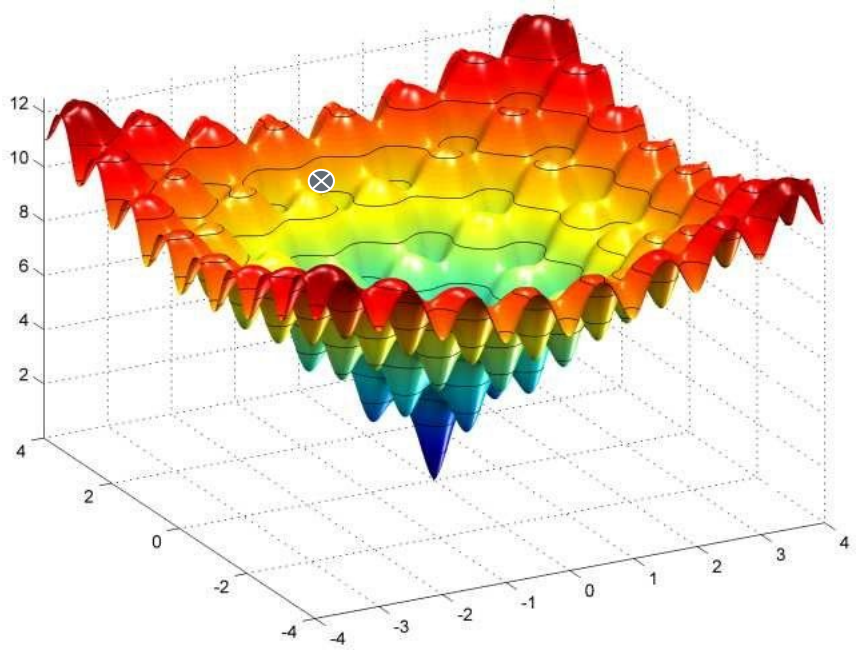
Nem sempre vai ser fácil chegar no mínimo global.

Precisamos de uma estratégia para ser menos ou mais agressivo na adaptação dos pesos.

Introduzimos então o **learning rate**.

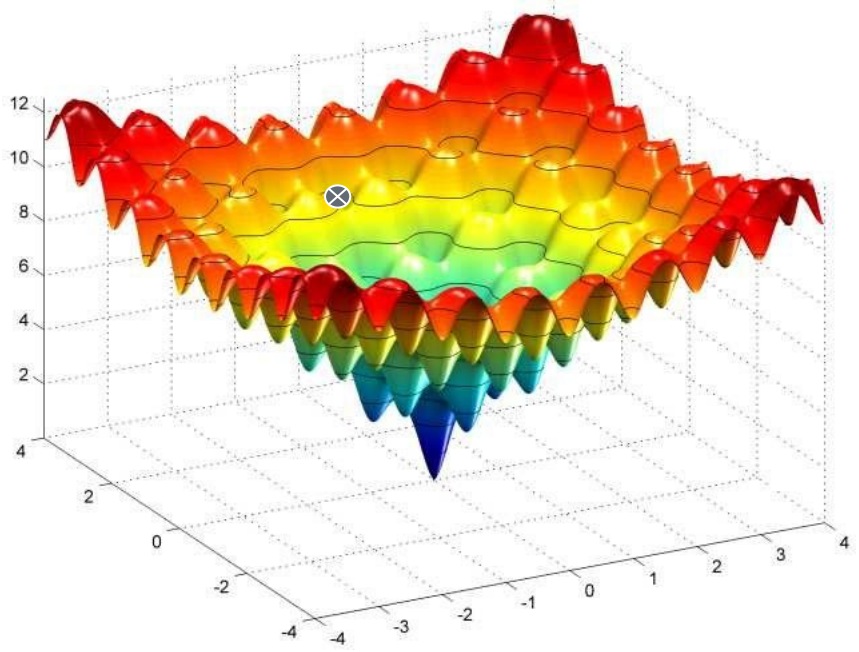
Como uma rede neural aprende?

Um **learning rate baixo** pode comprometer a nossa velocidade de aprendizado



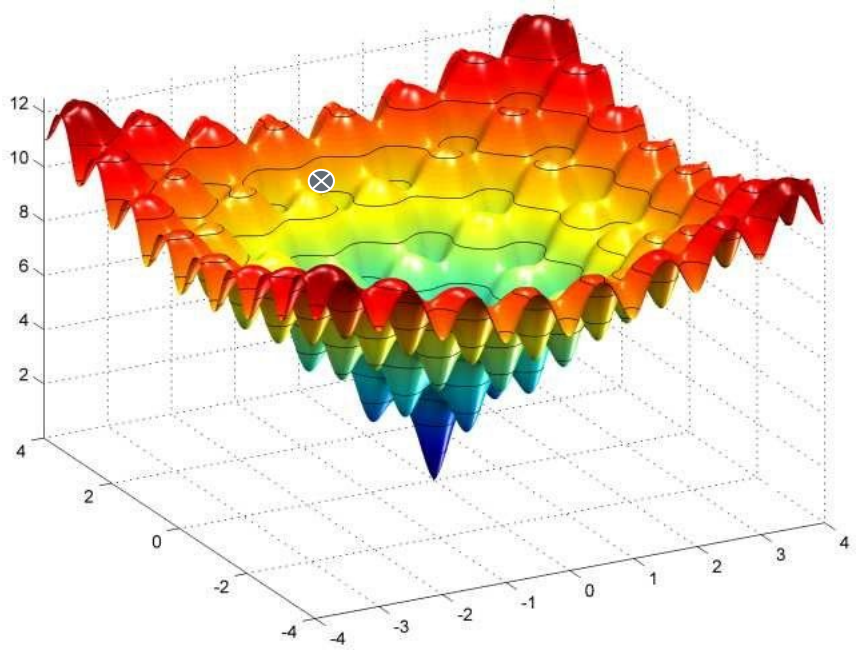
Como uma rede neural aprende?

Um learning baixo pode comprometer a nossa velocidade de aprendizado



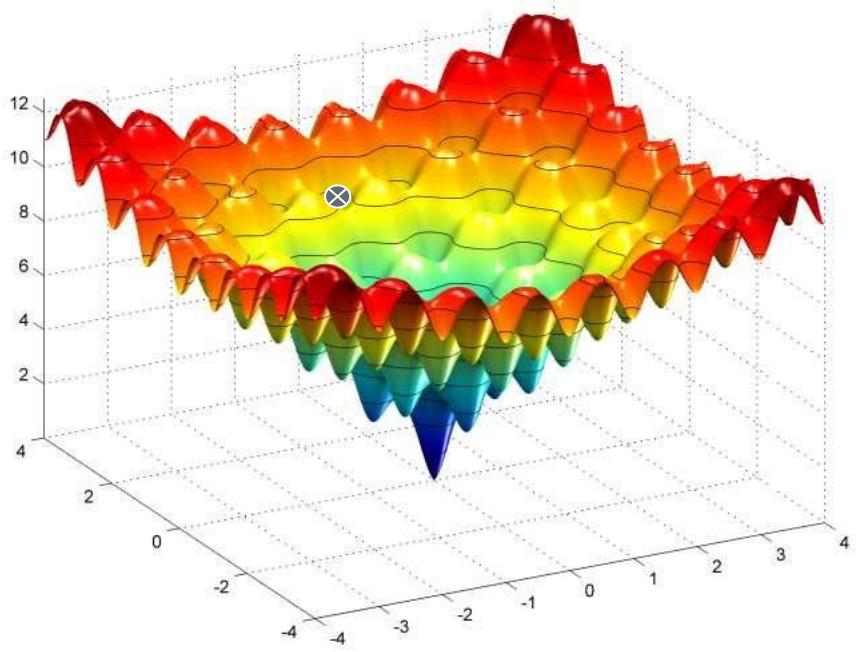
Como uma rede neural aprende?

E até nos prender em um mínimo local



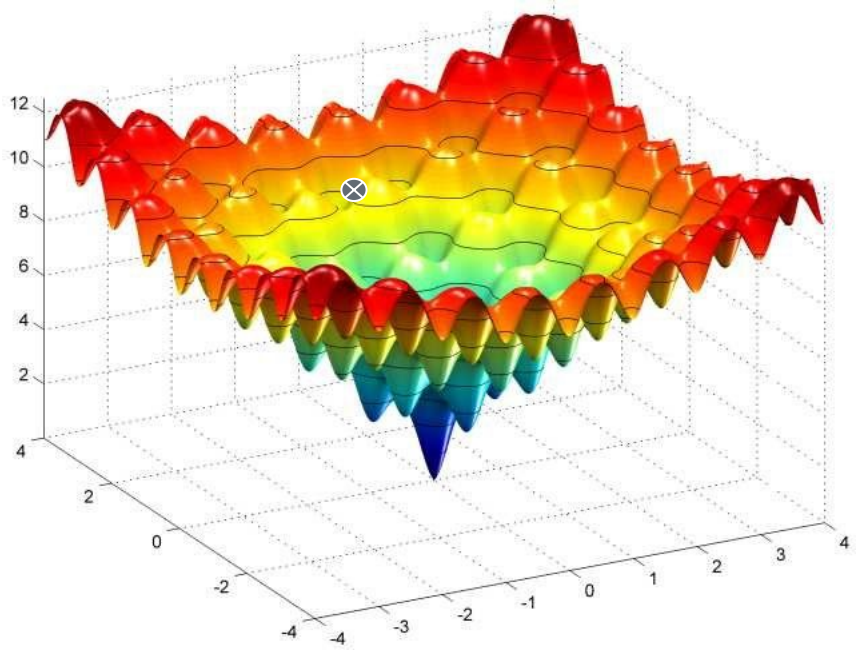
Como uma rede neural aprende?

E até nos prender em um mínimo local



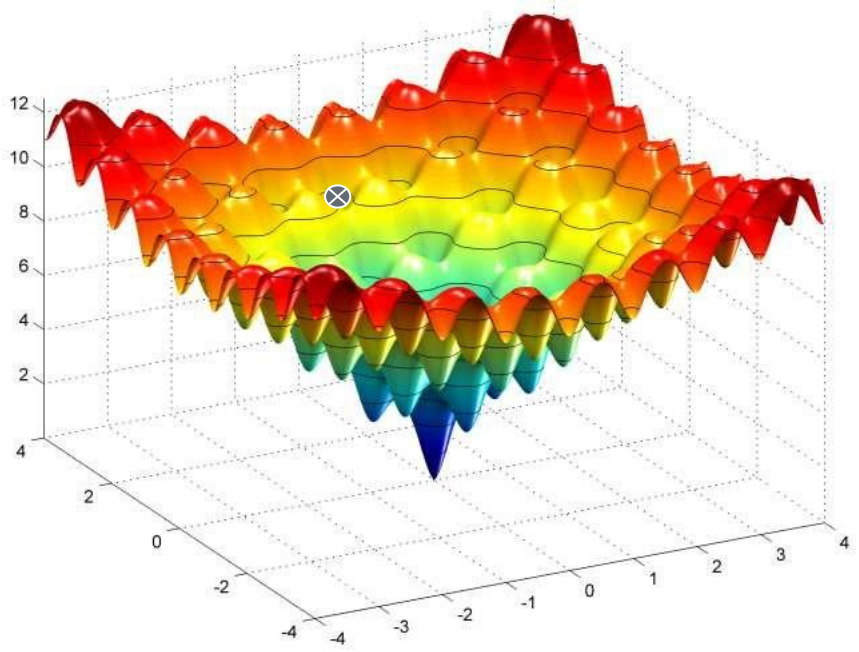
Como uma rede neural aprende?

E até nos prender em um mínimo local



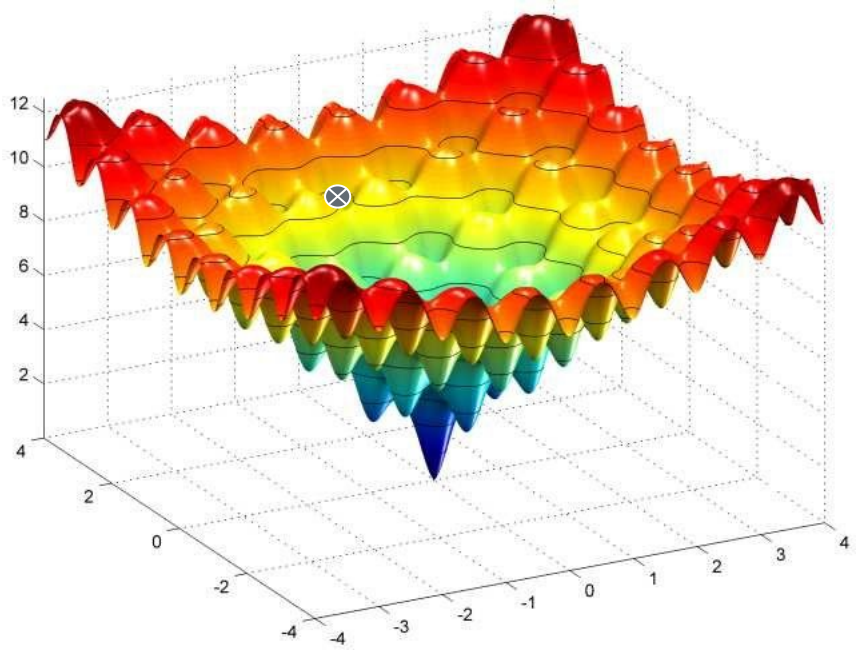
Como uma rede neural aprende?

E até nos prender em um mínimo local



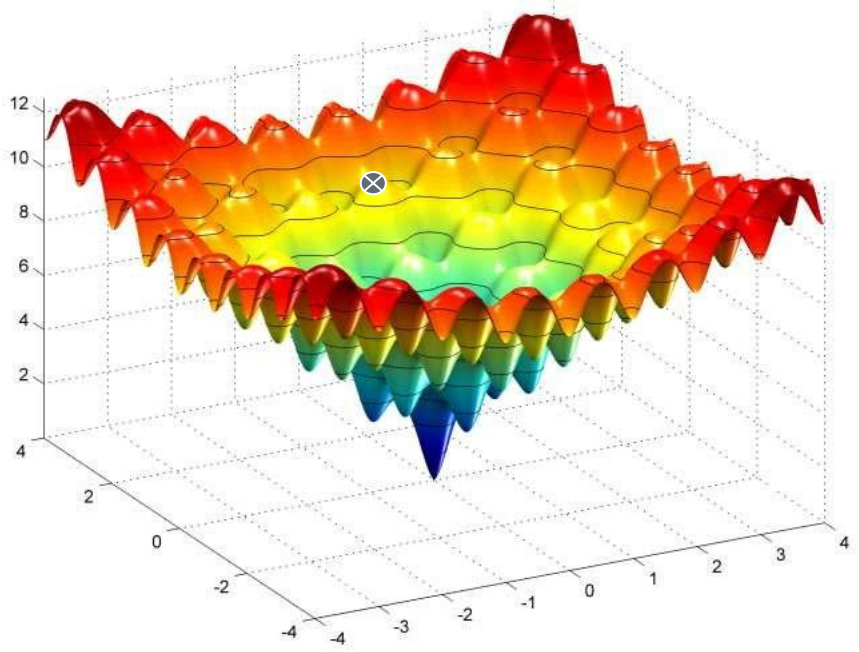
Como uma rede neural aprende?

Um **learning rate alto** pode ser bom para escapar de mínimos locais



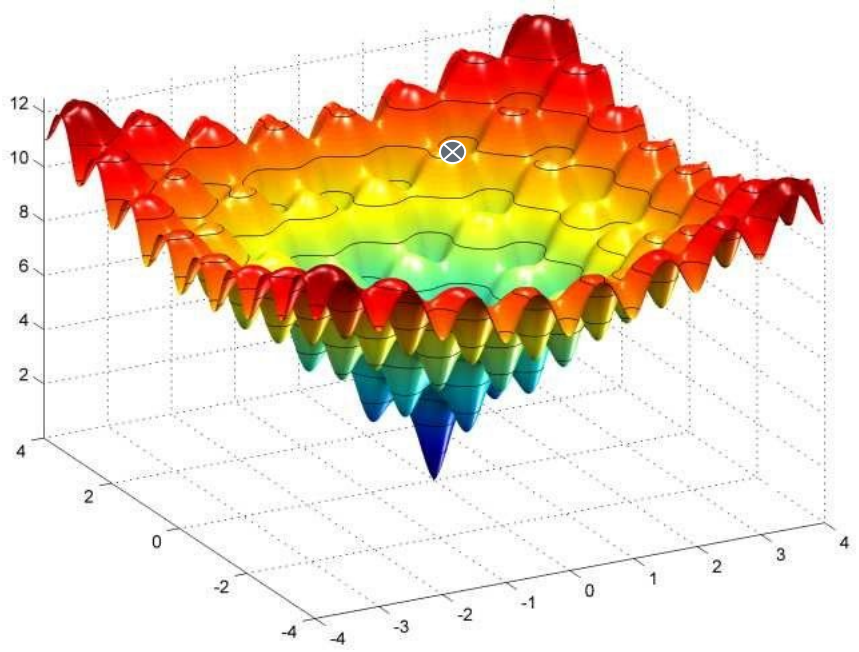
Como uma rede neural aprende?

Um learning rate alto pode ser bom para escapar de mínimos locais



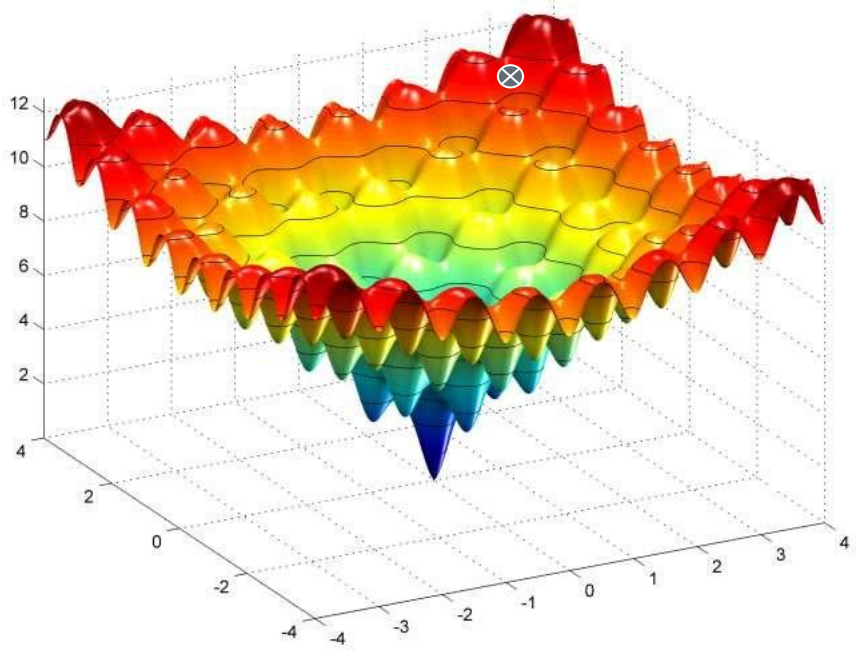
Como uma rede neural aprende?

Mas pode causar “overshoot” e nos “jogar pra cima” exageradamente



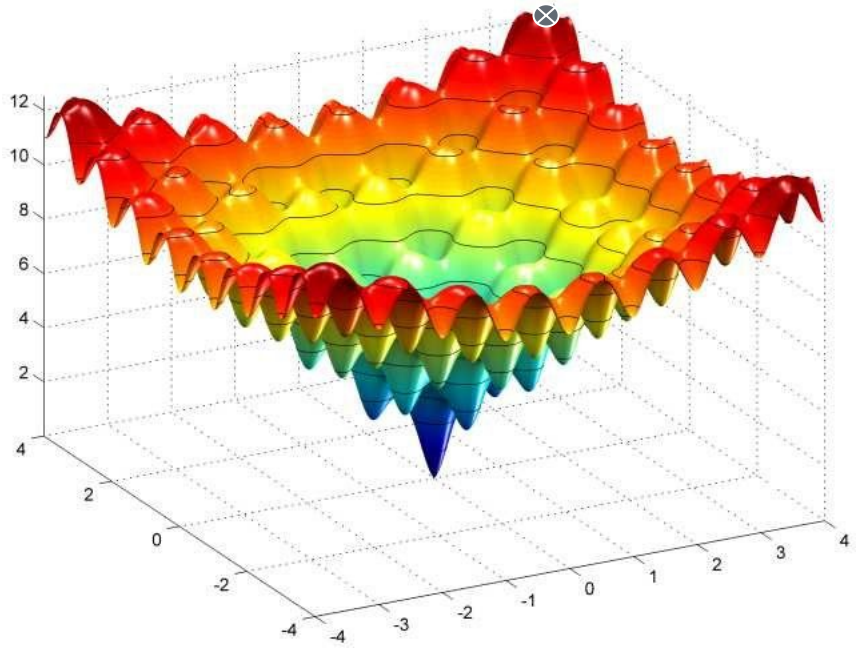
Como uma rede neural aprende?

Mas pode causar “overshoot” e nos “jogar pra cima” exageradamente



Como uma rede neural aprende?

Mas pode causar “overshoot” e nos “jogar pra cima” exageradamente



Como uma rede neural aprende?

Por isso é importante definir um learning rate que esteja de acordo com o necessário.

Definir um learning rate bom é um problema tão complexo, que normalmente utilizamos o suporte de **Optimizers** para deixar a busca do mínimo na função de perda mais flexível.

Exemplos:

- Gradient Descent
- Stochastic Gradient Descent
- ADAGRAD
- **Adam**

Hiper-parâmetros

Arquitetura geral:

- Número de **Camadas**;
- Número de **Neurônios por camada**.

Treino:

- **Batch Size:** Quantidade de dados analisados pela rede em cada passo do treinamento;
- **Dropout:** Probabilidade de desconectar um neurônio em um passo do treinamento;
- **Optimizer & Learning Rate**;
- **Epochs**: Quantidade de vezes que vamos ver todos os exemplos de treino.

Estrutura Geral da Aula

Parte teórica:

- Estrutura de uma Rede Neural
- O Perceptron
- Funções de ativação
- Como uma rede neural aprende?
- Hiper-parâmetros

Parte prática:

- Carregando e preparando DataSet (usando Pandas e Numpy)
- Rede Neural para classificar flores de íris
- Ajustando a rede para uma melhor performance

Recomendações (Inglês)

MIT DeepLearning (Perceptron e Redes Neurais):
https://www.youtube.com/watch?v=njKP3FqW3Sk

3Blue1Brown (Redes Neurais):
https://www.youtube.com/watch?v=aircAruvnKk

CodeEmporium (Optimizers):
https://www.youtube.com/watch?v=mdKjMPmcWjY

Keith Galli (Tensorflow / Keras):
https://www.youtube.com/watch?v=aBIGJeHRZLQ