CONSTELLATION
ASSET MANAGEMENT

CONSTELLATION TALKS #1

Abrindo a caixa de uma rede neural

Novembro | 2022

Quem é a **Constellation**?

SOMOS UMA GESTORA DE INVESTIMENTOS **QUE BUSCA GERAR VALOR NO LONGO PRAZO**

CONSTELLATION
ASSET MANAGEMENT

Requisitos

Precisamos garantir que quem está assistindo já possua alguns conhecimentos básicos

- Cálculo diferencial e integral (Derivada parcial)
- Noções de álgebra Linear (Matrizes e cálculo matricial básico)
- Estatística básica (Distribuições e Esperança)

Encontre o conteúdo em

https://github.com/Constellation-Dev-Team

Agenda

- O que é aprendizado de máquina?
- Como uma máquina pode aprender?
- Modelando preço de uma ação
- Definindo uma meta
- Otimizando uma função
- Como modificar o otimizador
- Perceptron e Perceptron de camada única

O que é aprendizado de máquina?

Campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados

Arthur Samuel (1959)

Tipos de aprendizado de máquina

Supervisionado	Não-Supervisionado	Por Reforço
Dados $X \in \mathbb{R}^n, y \in R$ Encontre f, *t.q.* $f(X) \mapsto y$	Dado $X \in \mathbb{R}^n$, encontre f, t.q. $f(X)$ extraia alguma informação relevante de X	Dados um conjuto de ações *A, um conjunto de estados S* e uma função de recompensa R, encontre $\pi(s, a) = max_{a \in A, s \in S} R(s)$
Você conhece X e y e quer encontrar algo que preveja y baseado em X	*Você quer encontrar algum padrão relevante em X.*	*Você quer encontrar uma solução onde você não consegue definir certos e errados, apenas recompensar o for correto*
Baseado no preço anterior, quero definir o próximo preço de uma ação	*Quero encontrar grupos semelhantes na minha amostra*	*Quero que um robô jogue xadrez*

CONSTELLATION
ASSET MANAGEMENT

Aprendizado Supervisionado

Regressão	Classificação
Dados $X \in \mathbb{R}^n, y \in R$ Encontre f, *t.q.* $f(X) \mapsto y$	Dados $X \in \mathbb{R}^n, y \in \{1,0\}^m$ Encontre f, t.q. $f(X) \mapsto y$
y é sempre um número real	*y é um conjunto de classes finitas definidos por inteiros ou um vetor "One Hot"*
Baseado no preço anterior, quero definir o próximo preço de uma ação	*Baseado no preço anterior, quero se no próximo período a ação vai subir ou cair*

Como uma máquina pode aprender?

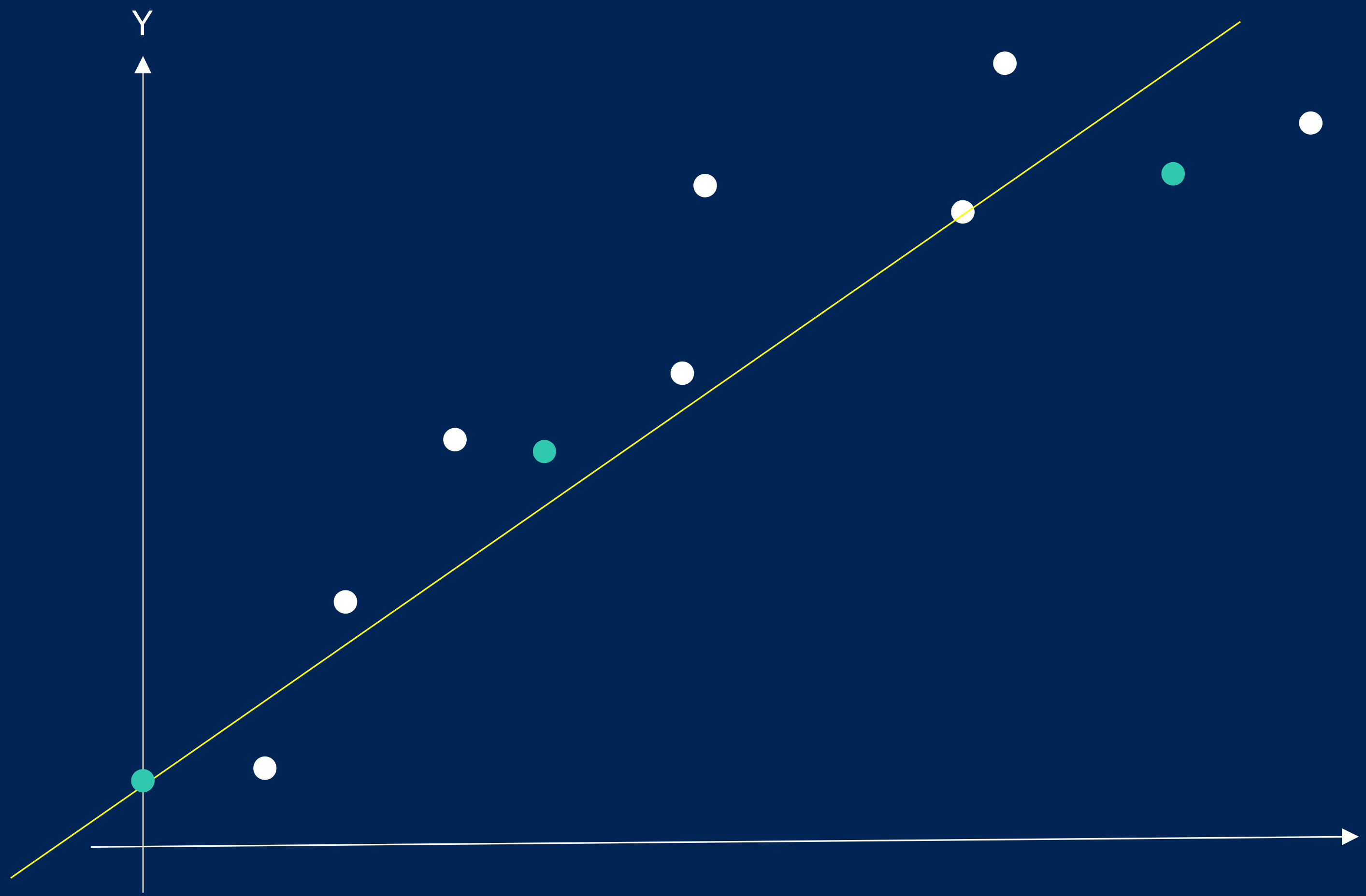
- Dado uma distribuição estatística dos dados que temos, sempre podemos dar um palpite mais acertado do que uma aposta aleatória

- Podemos assumir algumas propriedades dos dados e tentar estimar a partir dessas premissas

- Por exemplo: Ao assumir que a relação entre os dados em questão é linear, podemos criar uma regressão linear

Exemplo

Dado uma amostra de dados, podemos encontrar uma função linear



Modelando uma função linear

Dado n variáveis aleatórias, vamos criar uma função que define a variável aleatória y

$$h(x_1, x_2, x_3, \dots x_n) = \sum_{i=0}^{n} w_i x_i + b,$$

Podemos ainda definir X como uma matriz contendo todas as variáveis $x_1, x_2, x_3, \dots x_n$ de toda a amostra de tamanho m e W uma matriz com os pesos $w_1, w_2, w_3, \dots w_n$

$$X = \begin{bmatrix} x_{11} & \cdots & x_{n1} \\ x_{21} & \cdots & x_{n2} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix} = \begin{bmatrix} - x_1 - \\ - x_2 - \\ \vdots \\ - x_m - \end{bmatrix}, \mathrm{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Com isso, podemos redefinir h de forma matricial para toda a amostra:

$$h(X) = W \circ X + b$$

Modelando o mercado

Mas o que poderia ser modelado dessa forma?

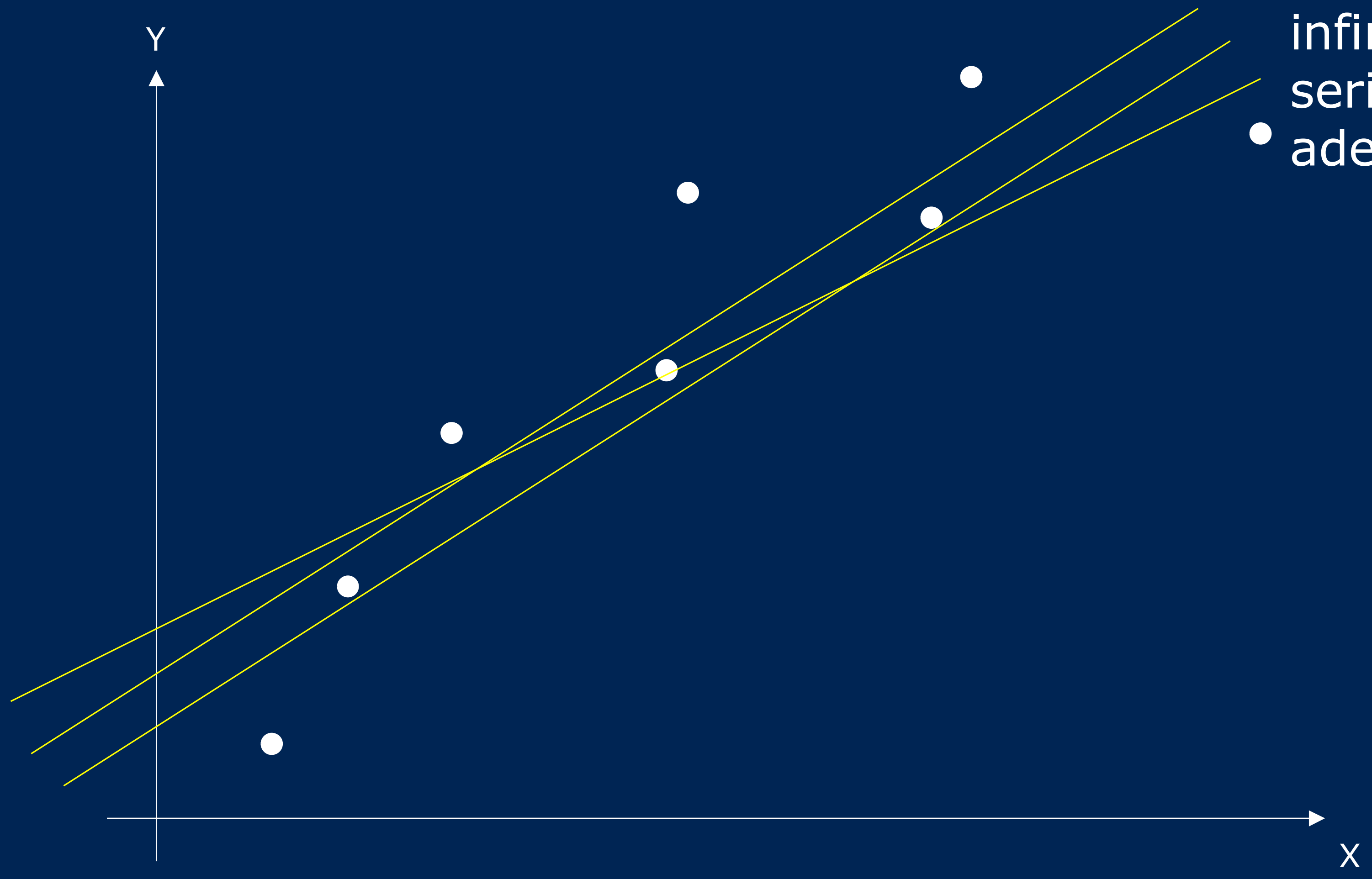
Para nosso exercício, iremos tentar modelar o preço de uma ação no dia seguinte.

O que podemos utilizar para tentar prever o preço de uma ação?

- Preços anteriores da ação
- Indicadores técnicos
    - SMA, RSI, Bollinger Bands, Donchian
- Fatores fundamentalistas
    - Lucro Bruto, Lucro líquido, Dívidas
- Indicadores econômicos
    - Juros, Inflação, IPCA, IGP-M

Exemplo

Porém, entre infinitas retas, qual seria a mais adequada?



Mas como posso encontrar a melhor função possível?

Para isso, precisamos definir uma segunda função, que vai servir para otimizar a primeira.

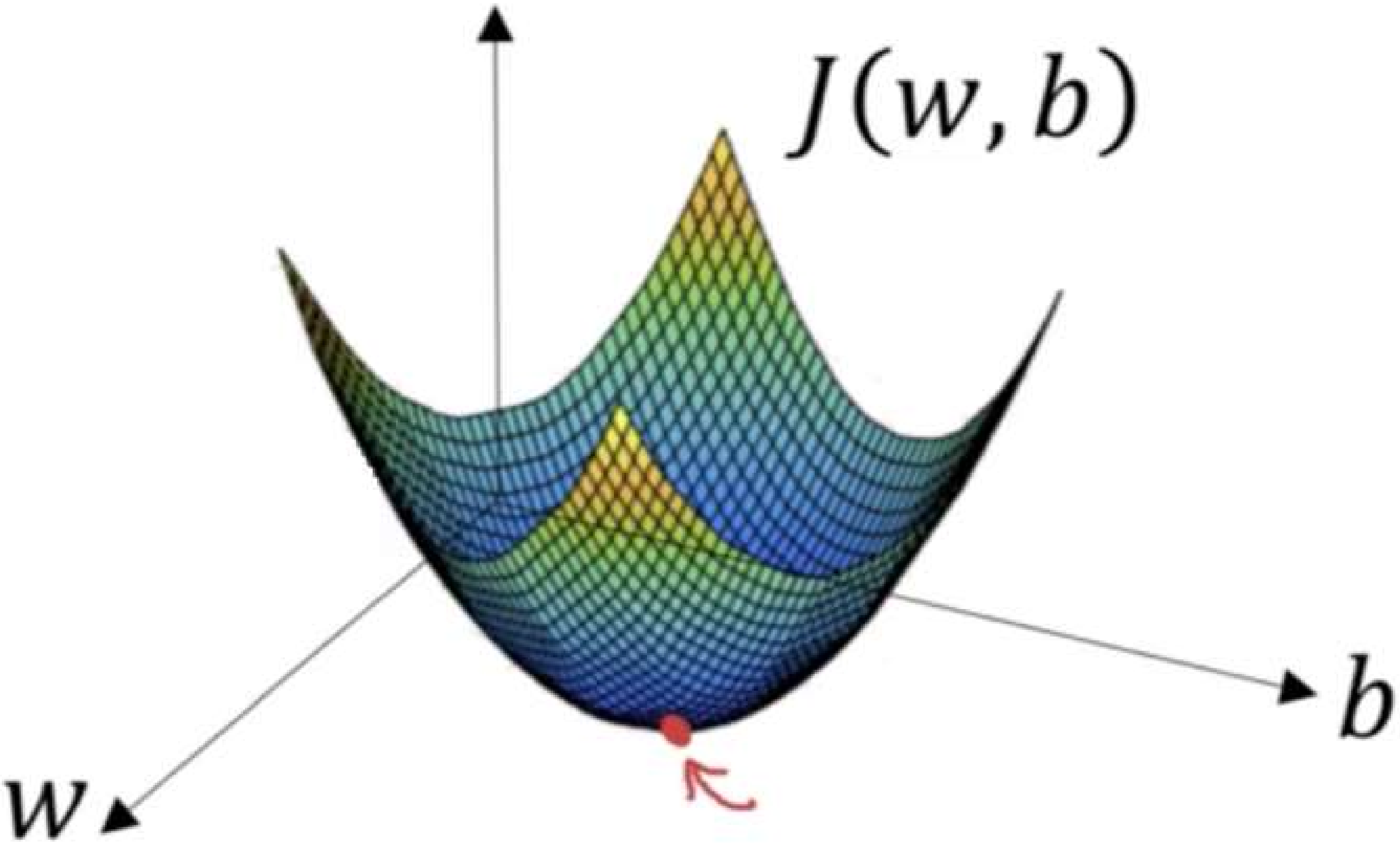
A chamamos função de custo, e podemos defini-la com a intenção de minimizar a distância média entre os pontos da nossa função linear e nossa medida final.

$$J_i(W) = \frac{1}{2}(\, h(x_i) - y \,)^2$$

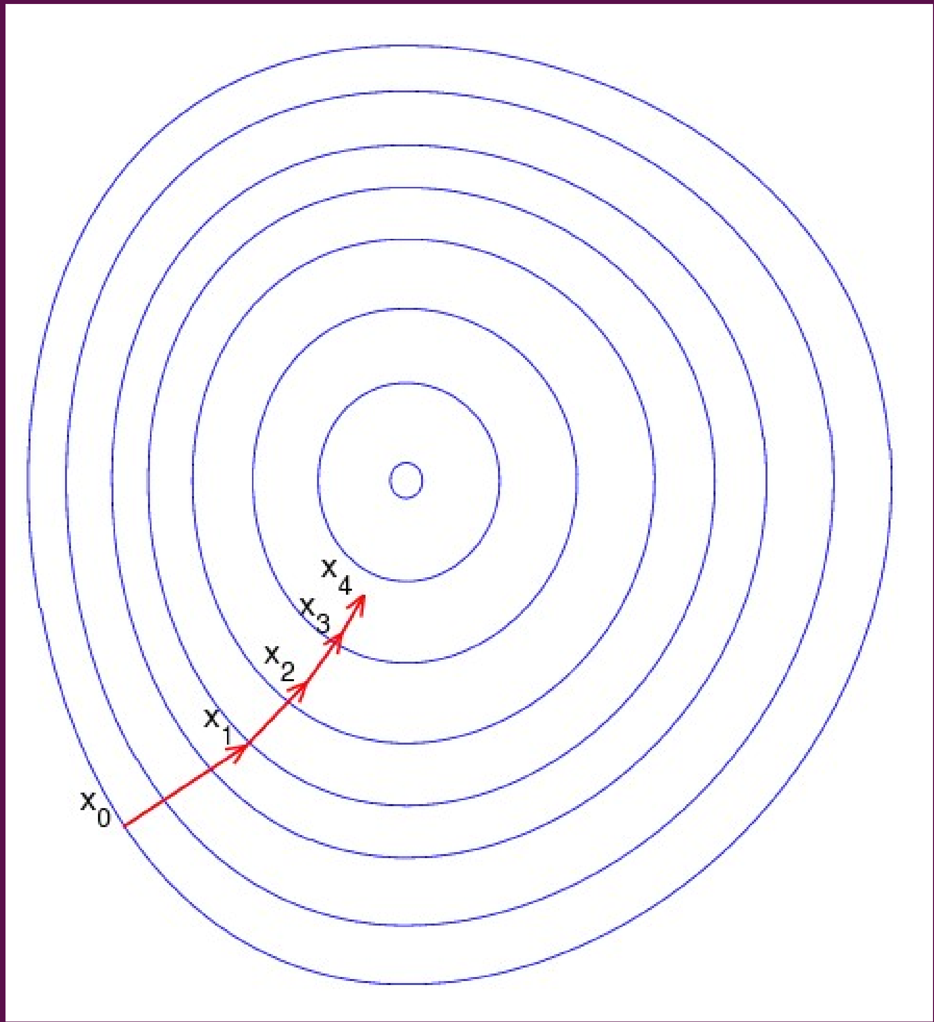
$$J(W) = \frac{1}{n}\sum_{i=0}^{n} J_i(x_i)$$

$$= \frac{1}{2n}(\,(W^T \circ X + b) - y\,)^2$$

Mas como posso encontrar a melhor função possível?



Gradiente Descendente



O que é o gradiente?

Como J é uma função quadrática e convexa, significa que ela possui um mínimo global.

Uma das formas de fazer isso é utilizando o gradiente da função J

O gradiente é um vetor com a derivada parcial da função J em relação a cada componente do vetor de pesos

$$\nabla_W J = \frac{\partial J}{\partial W} = \begin{bmatrix} \frac{\partial J}{\partial w_1} \\ \frac{\partial J}{\partial w_2} \\ \vdots \\ \frac{\partial J}{\partial w_n} \end{bmatrix}$$

Gradiente Descendente

Algoritmo do Gradiente Descente:
Repita até convergir:

$$\mathrm{W} \coloneqq W - \alpha \nabla_W J$$

sendo α a taxa de aprendizado

Tudo bem, mas e a Rede Neural?

Redes neurais são constituídas de vários elementos chamados perceptrons, que são otimizadores quase-lineares.

$$h(x_1, x_2, x_3, \dots x_n) = \sigma\left(\sum_{i=0}^{n} w_i x_i + b \right)$$

Onde σ é uma função não-linear.

Funções utilizadas em geral são:

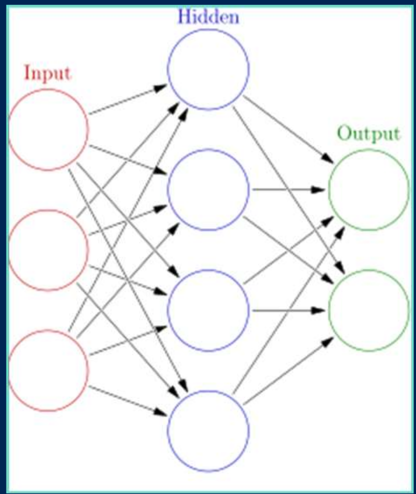
Função sigmoide : $\mathrm{sig}(x) = \frac{1}{1+e^{-x}}$

Função sigmoide : $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Função ReLU: $ReLU(x) = \max(0, x)$

Tudo bem, mas e a Rede Neural?

Quando vários perceptrons são unidos, eles conseguem separar o hiper-espaço em pedaços além dar pesos para as associações entre as diferentes entradas.



Podemos definir uma rede neural de 2 camadas como:

$$z_1 = W^{[1]} \circ X + b_1 \qquad a_1 = \sigma(z_1)$$

$$z_2 = W^{[2]} \circ a_1 + b_2 \qquad a_2 = \sigma(z_2)$$

Atualização dos Pesos

A atualização dos pesos da rede neural é chamada de Backpropagation. Em geral, é utilizado algoritmo do gradiente descendente ou alguma otimização dele

$$W^{[1]} \coloneqq W^{[1]} - \alpha \nabla_{w^{[1]}} J$$
$$b_1 \coloneqq b_1 - \alpha \nabla_{b_1} J$$

$$W^{[2]} \coloneqq W^{[2]} - \alpha \nabla_{w^{[2]}} J$$
$$b_2 \coloneqq b_2 - \alpha \nabla_{b_2} J$$

Atualização dos Pesos

Definiremos $\delta^{[1]}$ *e* $\delta^{[2]}$*:*

$$\delta^{[2]} \triangleq \frac{\partial J}{\partial a_2} = (a_2 - y)$$

$$\delta^{[1]} \triangleq \frac{\partial J}{\partial z_1} = (a_2 - y) \cdot {W^{[2]}}^T \odot \sigma'(z_1)$$

Os gradientes ficam como:

$$\nabla_{w^{[1]}} J = \delta^{[1]} x^T$$

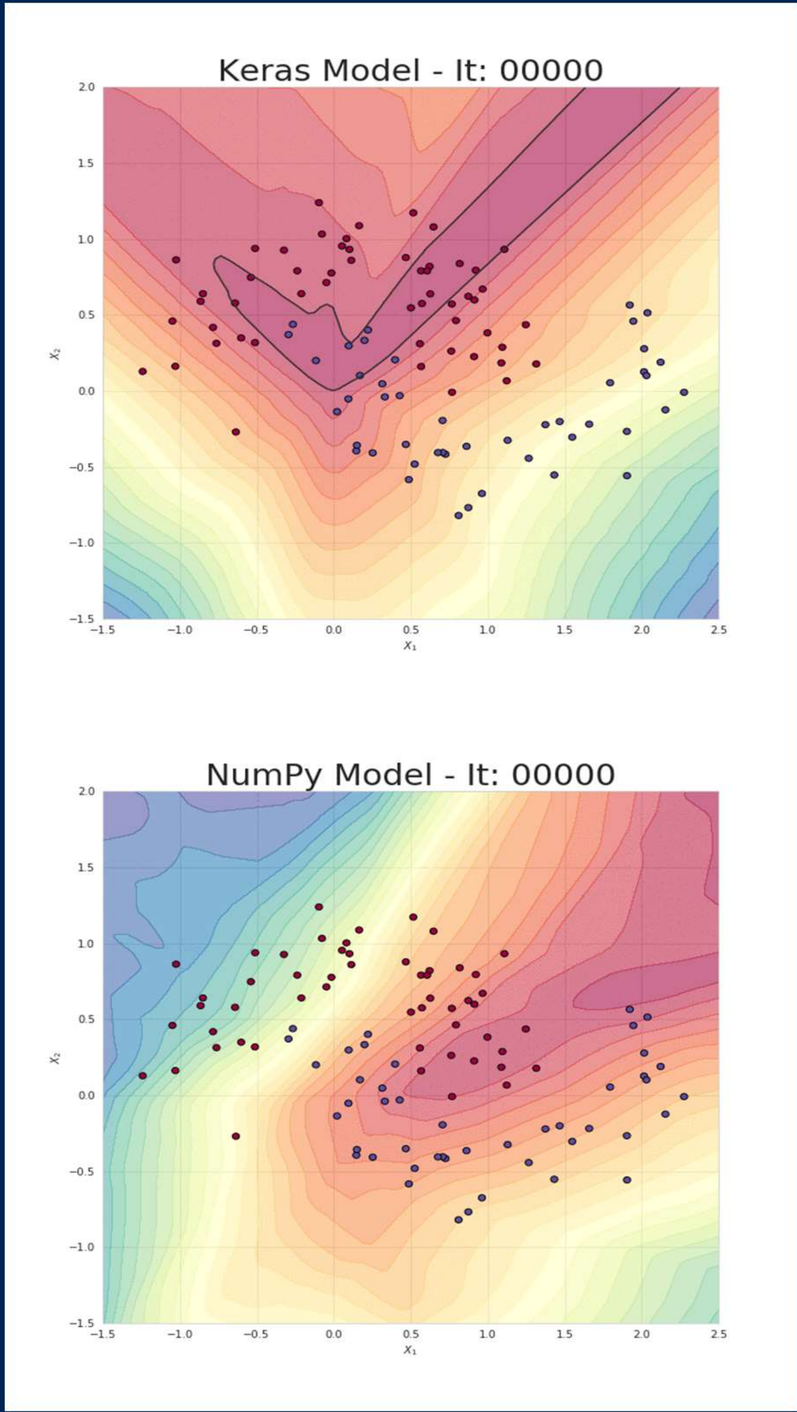
$$\nabla_{b_1} J = \delta^{[1]}$$

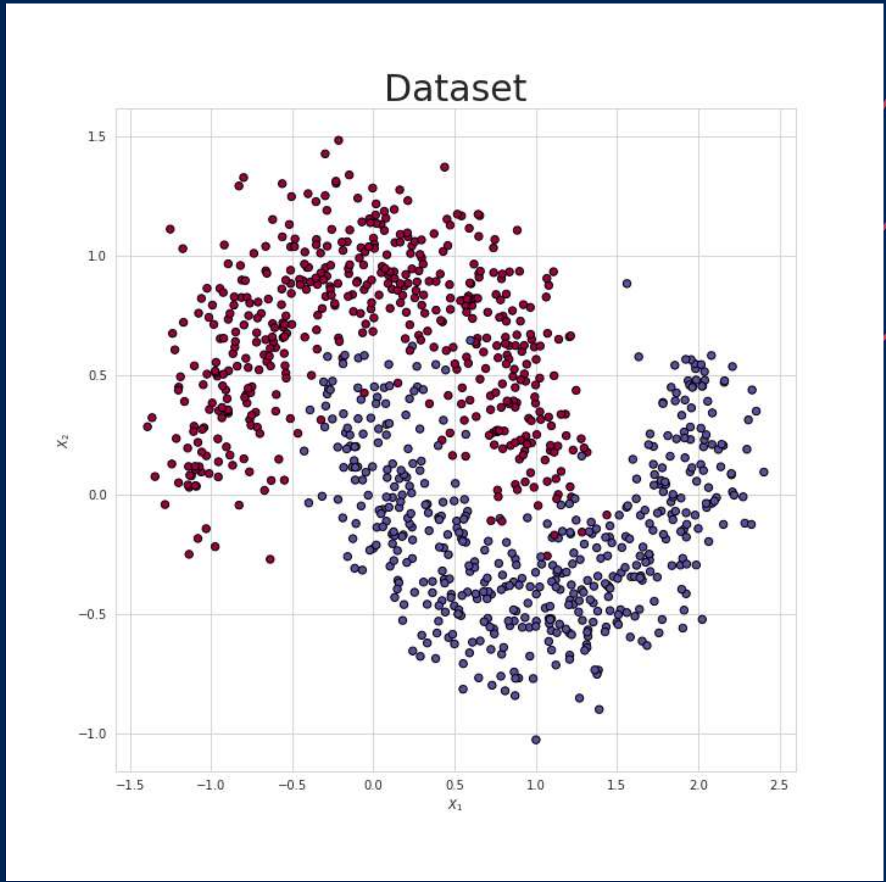
$$\nabla_{w^{[2]}} J = \delta^{[2]} a^T$$

$$\nabla_{b_2} J = \delta^{[2]}$$

Exemplo

Com múltiplas camadas, mesmo datasets não lineares podem ser resolvidos.





Agora vamos para o código

CONSTELLATION
ASSET MANAGEMENT

Referências

Wikipédia https://pt.wikipedia.org/wiki/Aprendizado_de_m%C3%A1quina

Builtin – Gradient Descent https://builtin.com/data-science/gradient-descent

Let's Code a Neural Network https://towardsdatascience.com/lets-code-a-neural-network-in-plain-numpy-ae7e74410795