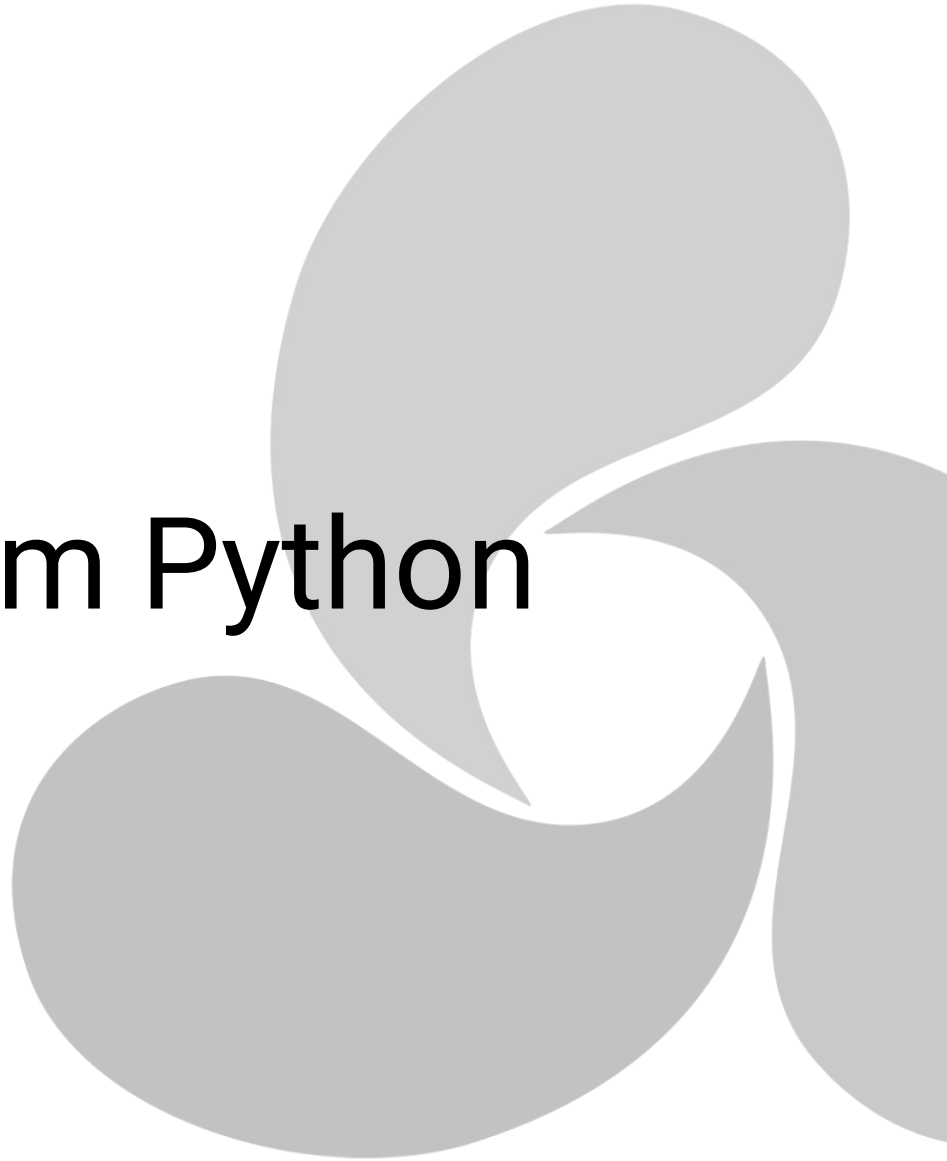
Machine Learning:
uma introdução prática em Python

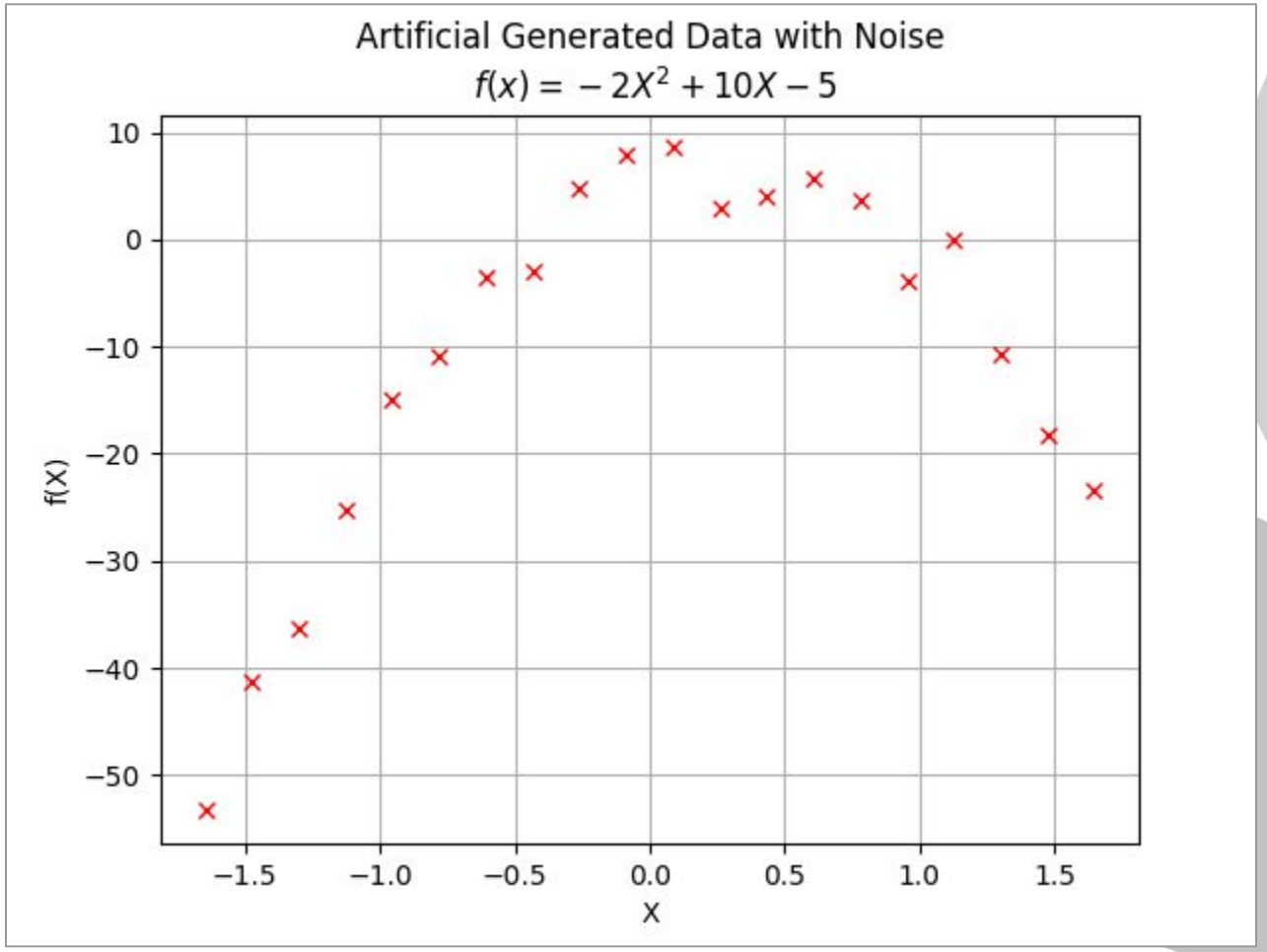
Otacílio “Minho” Neto, minhotmog@gmail.com



Regressão Polinomial

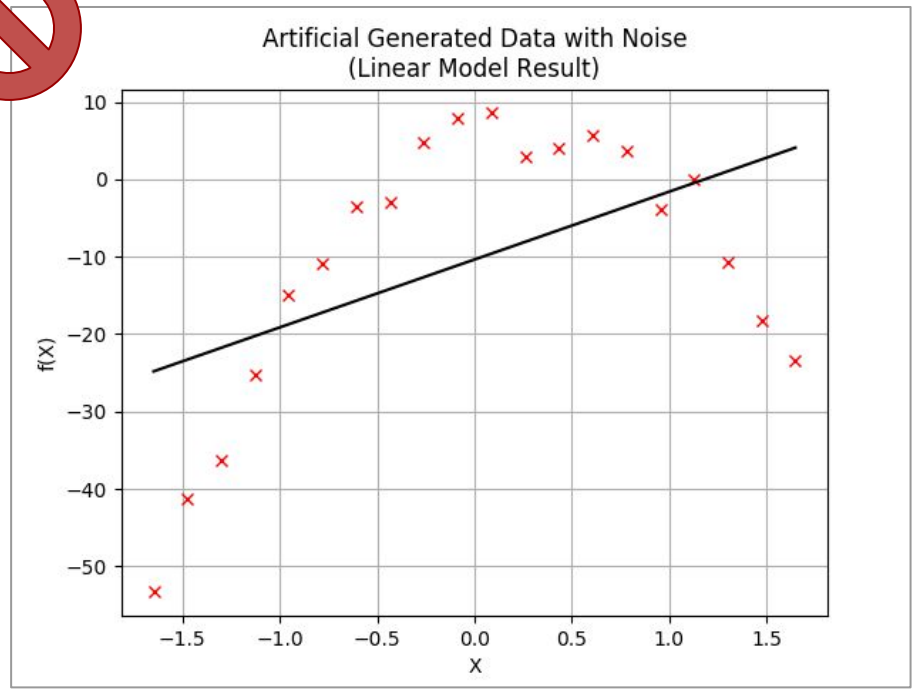
Criando modelos mais complexos!

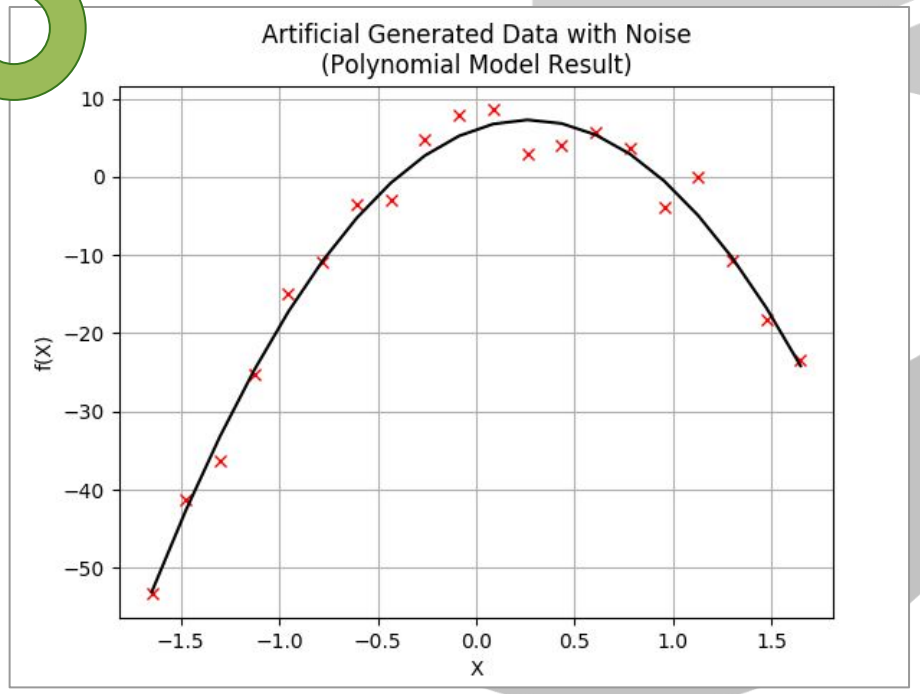
Como podemos aplicar uma Regressão Linear nesses dados?



Modelos Não-Lineares

É ingênuo considerar que todas as relações que podemos encontrar se comportam de forma linear!





Modelos Não-Lineares

- O problema está no modelo!
  Até agora estávamos usando modelos lineares:

$$h(\theta) = \theta_1 X + \theta_0$$

- Como a própria equação denunciava, os dados só podem ser modelados por um modelo polinomial:

$$h(\theta) = \theta_2 X^2 + \theta_1 X + \theta_0$$

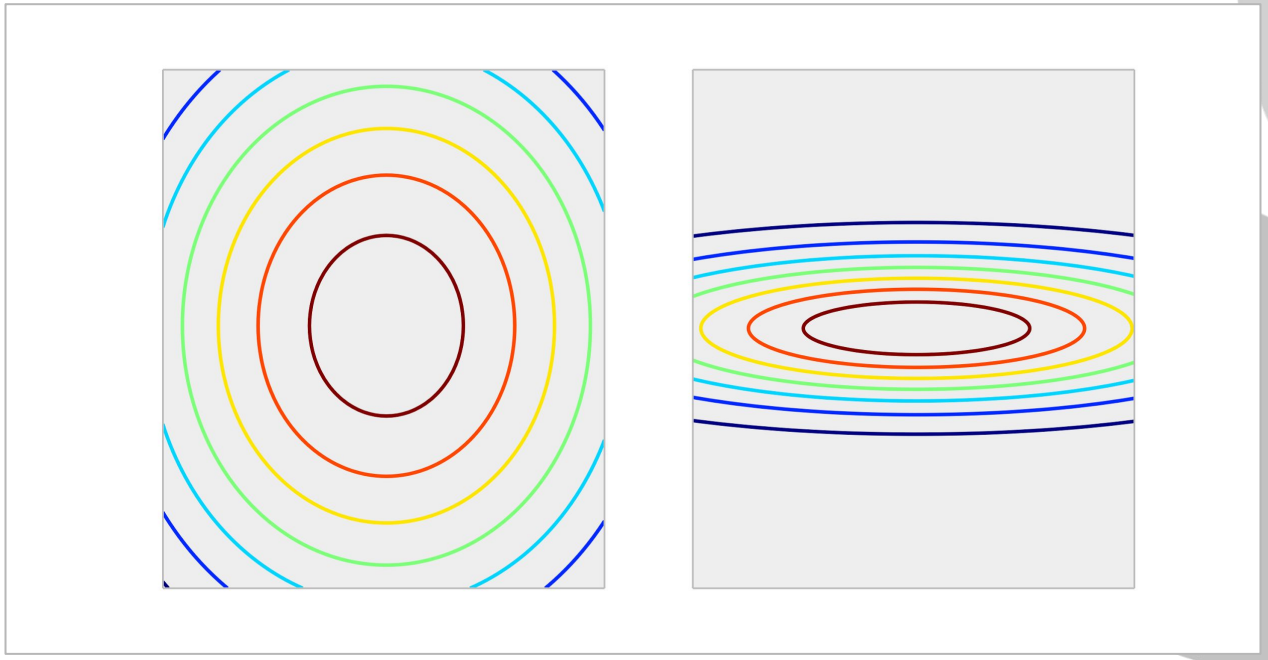
Algoritmo da Regressão Polinomial

- O treinamento é idêntico à da Regressão Linear, a diferença é o modelo. O objetivo, então, é incluir novos atributos: **transformações não lineares dos dados observados!**

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ X^{(1)} & X^{(2)} & \cdots & X^{(m)} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & \cdots & 1 \\ X^{(1)} & X^{(2)} & \cdots & X^{(m)} \\ (X^{(1)})^2 & (X^{(2)})^2 & \cdots & (X^{(m)})^2 \\ \vdots & \vdots & \ddots & \vdots \\ (X^{(1)})^n & (X^{(2)})^n & \cdots & (X^{(m)})^n \end{bmatrix}$$

O Problema da Escala

- É natural imaginar que, com essas transformações, vamos ter atributos com valores muito (e muito!) maiores que às suas versões lineares.
- **Problema:** Gradientes altíssimos!



Feature Scaling

- Para evitar problemas de divergência, treinos demorados e manter os parâmetros com valores “humildes”, devemos pôr os atributos em escalas semelhantes.

$$X = \frac{X - \overline{X}}{\sigma(X)}$$

- Não realize *feature scaling* antes de realizar as transformações!

Hands-On!!!

Regularização (Extra)

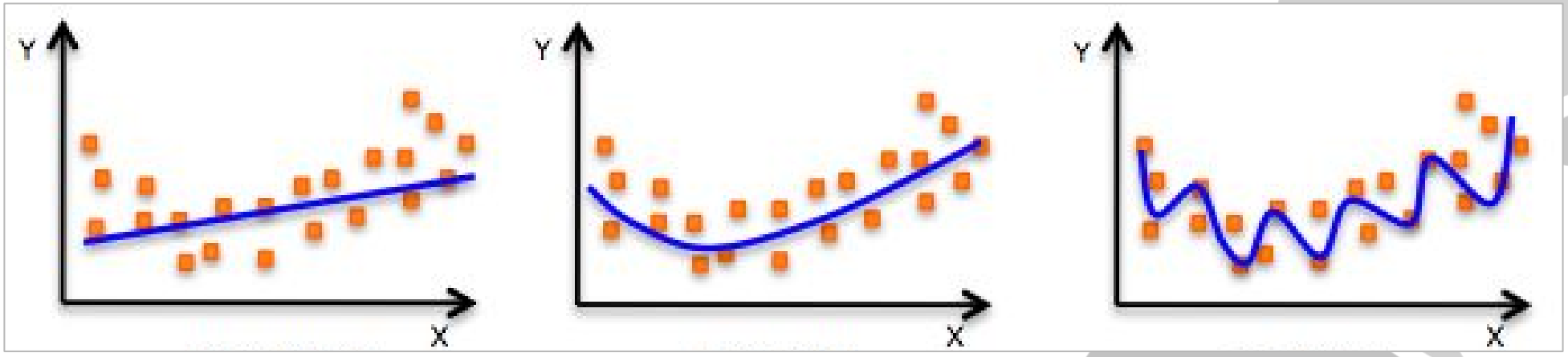
Dando estrutura aos nossos modelos.

O Caso da Generalização

Tolo! Tu fostes enganado!

- Na realidade, minimizar o erro é nosso objetivo; mas (em parte dos casos) **não queremos que o erro seja zero**!
- *Conjunto de Treino / Conjunto de Teste* → **Cross-validation**
- O Treinamento é apenas uma pequena parcela das complexas aplicações de *Machine Learning*!

Underfitting / Fitting / Overfitting



Underfitting Fitting Overfitting

Regularização!

- Na prática, um dos casos do Overfitting é a magnitude dos parâmetros. (**+complexidade** == **+sensibilidade**)

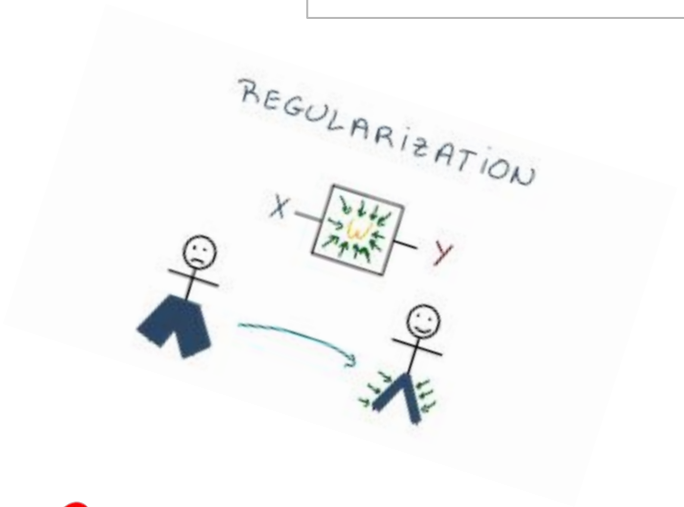
- Uma das maneiras mais simples de penalizar a magnitude dos parâmetros é simplesmente inserir seus valores (positivos) na Função de Custo!

Função de Custo Regularizada:

$$J(\theta) = \frac{1}{m}\sum(h(\theta) - y)^2 + \frac{1}{m}\sum(\theta)^2$$

Regularized Gradient Descent

$$\begin{cases} \theta_j = \theta_j - \alpha \ \frac{1}{m} \sum (h(\theta) - y) x_j & \text{if } j = 0 \\ \theta_j = \theta_j - \alpha \left[\frac{1}{m} \sum (h(\theta) - y) x_j \ + \ \frac{\gamma}{m} \theta_j \right] & \text{if } j >= 1 \end{cases}$$



Classificação

O segundo principal gênero de Aprendizagem Supervisionada.

O que é **Classificação**?

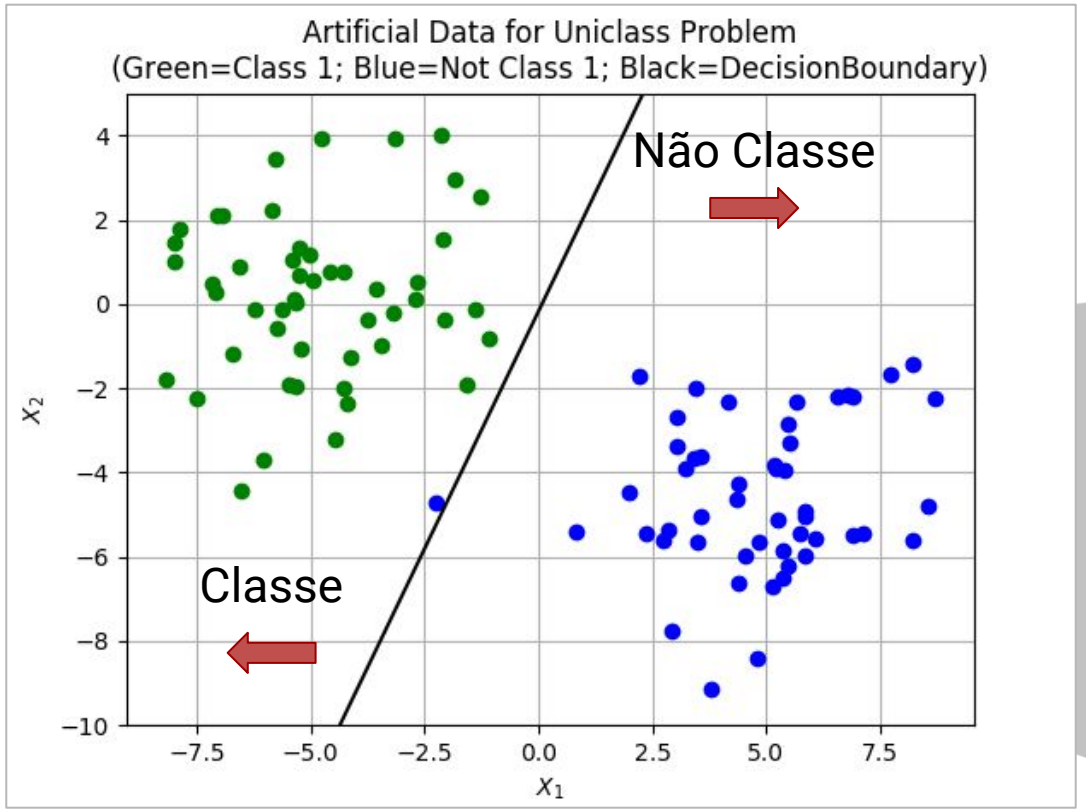
- Conjunto de técnicas destinadas a, através de **atributos**, indicar se um exemplar pertence (ou não) a determinada **classe** *(Uniclass Problem)* ou, dado um conjunto de **classes**, determinar à qual o exemplar pertence *(Multiclass Problem)*.

- **Regressão:** espaços contínuos.
  Classificação: espaços discretos.

- **Ex.:** diagnóstico de doenças; reconhecimento de objetos; processamento de fala; classificação pura; etc.

Decision Boundary (Linha de Decisão)

Para essa problemática, não precisamos de "*uma linha que acompanha os dados*", mas sim de "*um hiperplano que os separa*".



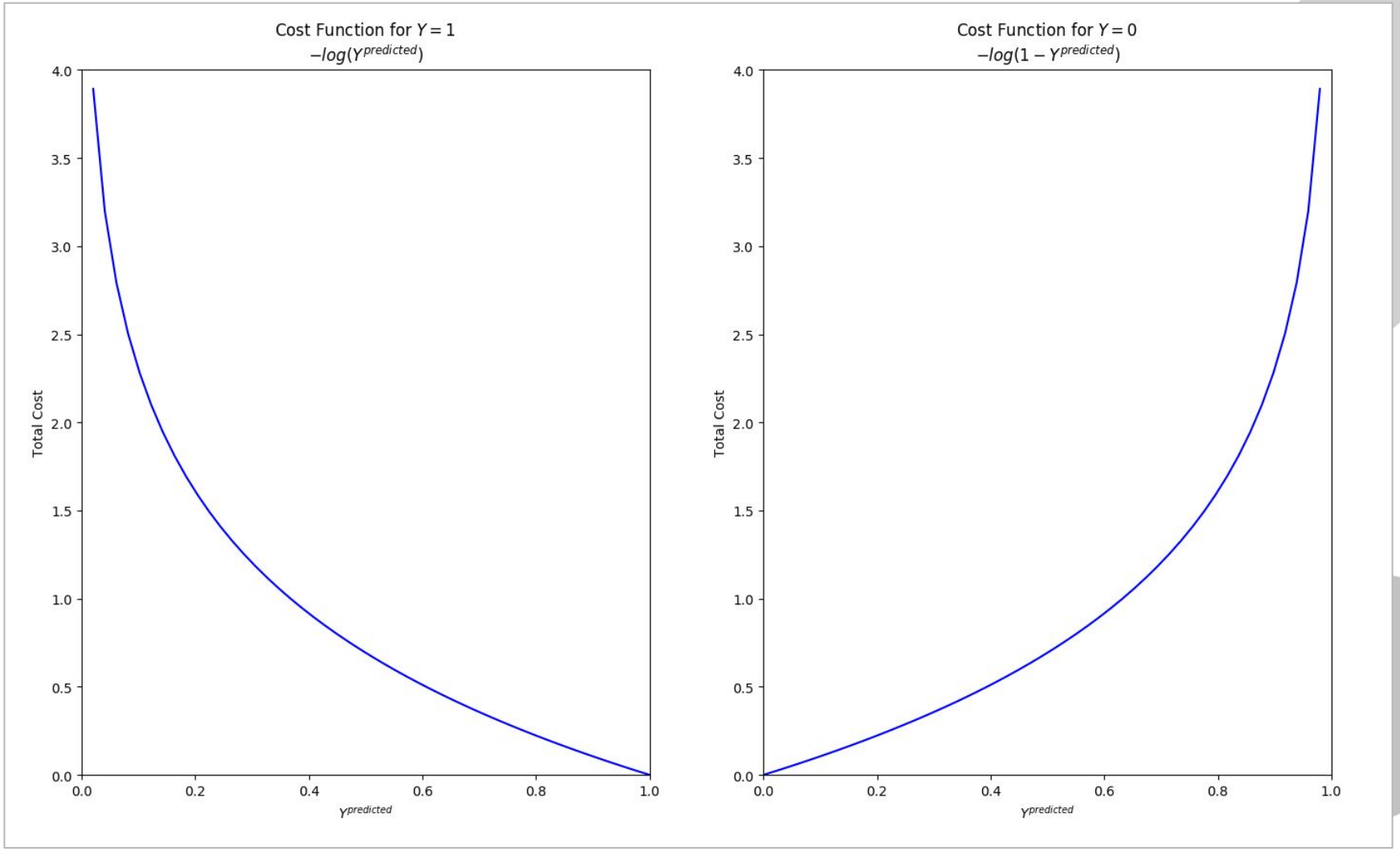
Cálculo de Erro

- Algoritmos de *Aprendizagem Supervisionada* possuem esta abordagem:
  - Definir função de custo sobre as predições;
  - Calcular gradientes com base na função de custo;
  - Utilizar gradientes para convergir a um mínimo;

- A questão, então, é a seguinte:

Como calculamos o custo total de uma Classificação?

Cálculo de Erro



Regressão Logística

Nosso primeiro algoritmo de Classificação!

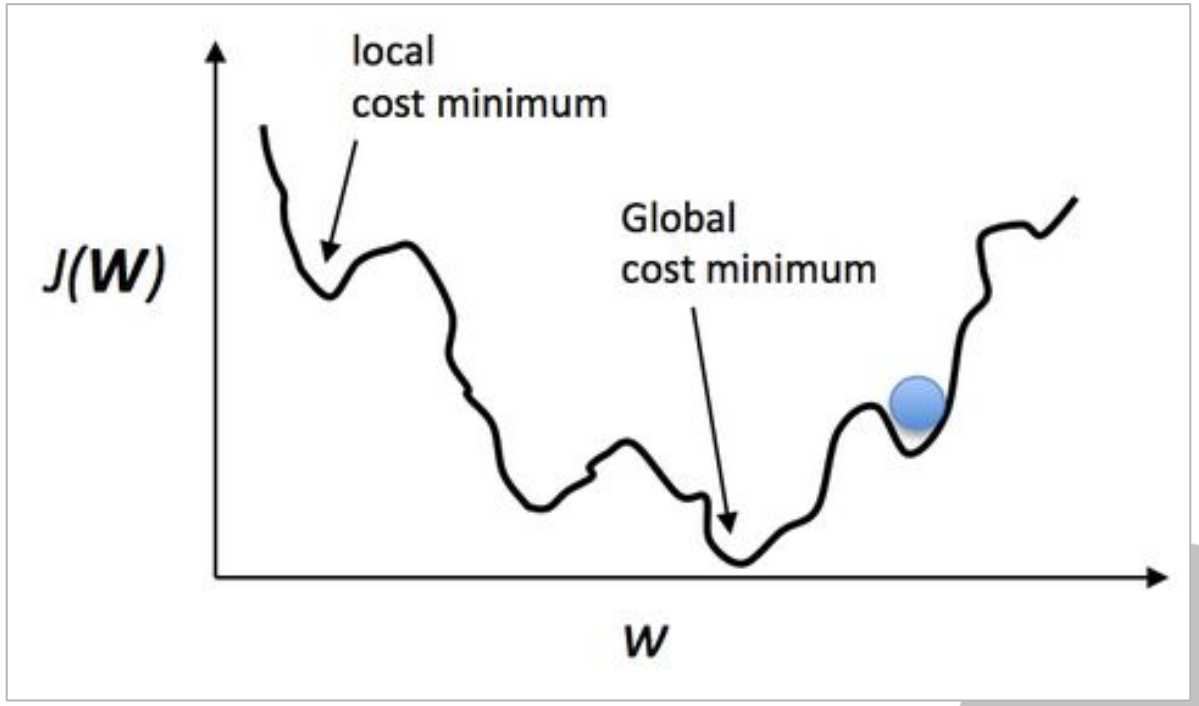
Utilizar Regressão?

- Podemos resumir àquelas duas funções anteriores em apenas uma só:

$$J(\theta) = \sum -(1-y)log(1-h(\theta))-(y)log(h(\theta))$$

- Isso significa que agora podemos utilizar aquele nosso modelo anterior e utilizar o Gradiente Descendente para achar os parâmetros, certo? **ERRADO!**

A função não é convexa!

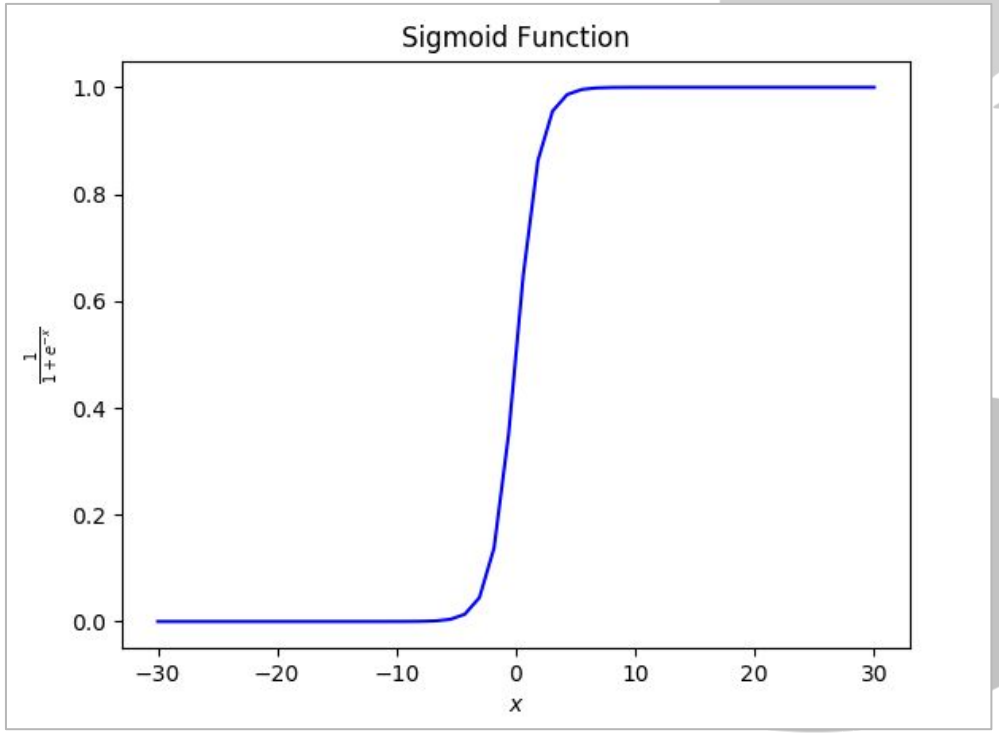


Nosso dever é definir um novo modelo que permite a Função de Custo ser convexa! (e então aplicamos o Gradiente Descendente)

Função Logística (Sigmoid)

Devemos “filtrar” nosso modelo, para que ele sempre retorne valores no intervalo **[0, 1]** independente dos atributos e parâmetros.

$$g(x) = \frac{1}{1 + e^{-x}}$$



Modelo Logístico

Nosso novo modelo, então, será uma aplicação da função logística no nosso modelo anterior

$$h^{new}(\theta) = g(h^{old}(\theta)) = \cdots$$

$$\cdots = \frac{1}{1 + e^{-(h^{old}(\theta))}} = \frac{1}{1 + e^{\theta^T X}}$$

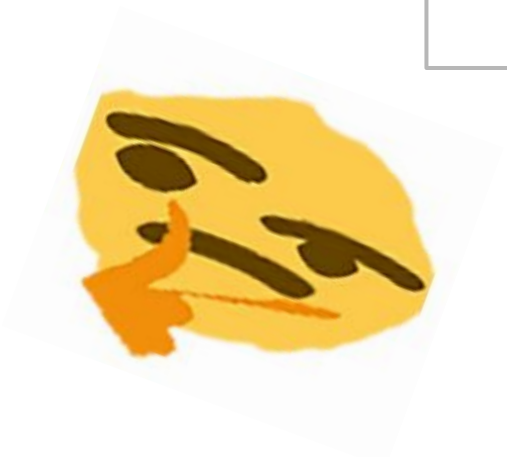
Cálculo dos Gradientes

Agora que nossa função foi remodelada, o Custo Total será convexo.

Vamos, então, calcular as Derivadas Parciais (Gradientes) da Função de Custo total?

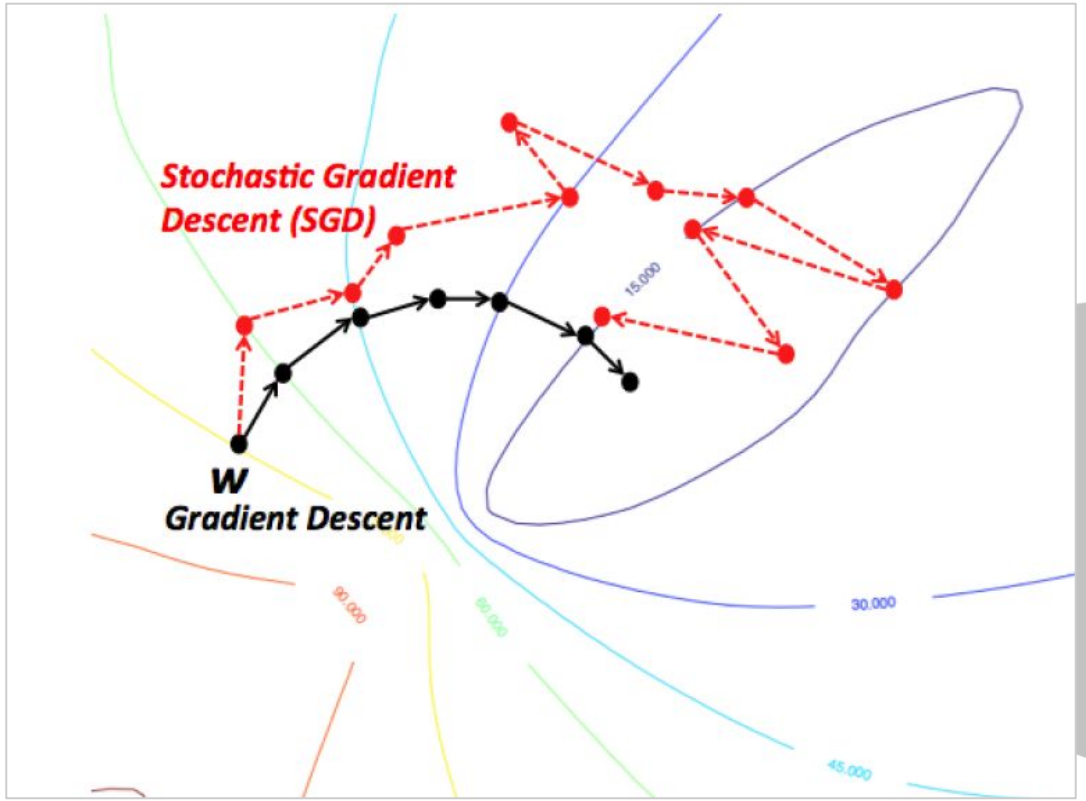
Cálculo dos Gradientes

$$\frac{\delta}{\delta\theta_i}(J(\theta)) = \frac{1}{m}\sum(h(\theta) - y)x_i$$



Stochastic Gradient Descent

$$\theta_j = \theta_j - \alpha\ (h(\theta) - y)x_j$$



Hands-On!!!

Obrigado!

minhotmog@gmail.com

www.petcomp.ufc.br