Um modelo de ML

- Dados
- Modelo
- Problema de otimização
- Algoritmo de otimização

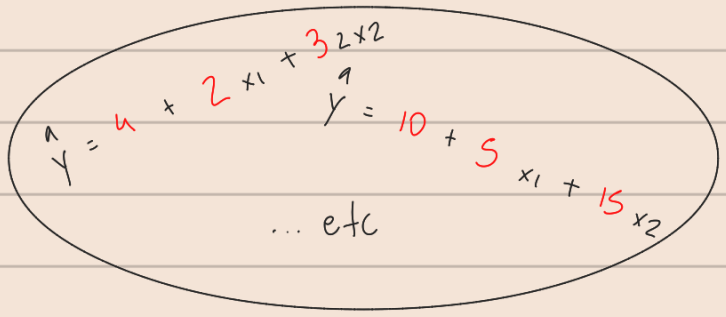
Um **modelo** é um **conjunto de funções preditoras** que **obedecem à mesma regra de construção.**

Exemplo: modelo linear para o problema de prever o **nível de açúcar** de uma banana baseado no seu **peso** e **comprimento**

y: nvl açúcar | x_1: peso | x_2: comprimento

Então o modelo linear é:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$



O **problema de otimização** define um critério de escolha da **função preditiva** que melhor se ajusta a os **dados**

Exemplo: para o modelo linear é usual definir a **função de perda** como sendo o MSE

$$MSE(\vec{\theta}; X_{train}, y_{train}) = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}^{(i)} - y^{(i)})$$

$\hookrightarrow \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)}$

Então, o **problema de otimização** é minimizar o MSE em relação aos **parâmetros** $\vec{\theta} = (\theta_0, \theta_1, \theta_2)$

$$\vec{\theta}_{opt} = \underset{\vec{\theta}}{\operatorname{argmin}}\ MSE(\vec{\theta}; X_{train}, y_{train})$$

O algoritmo de otimização é o procedimento usado para encontrar o ponto mínimo da função de perda

Exemplo: no modelo linear, temos alguns algoritmos de otimização:

1. **Equação normal**

1. Crie $\underset{\sim}{X} = \left[\begin{array}{c|c} \text{coluna de 1's} & X_{train} \end{array}\right]$

↑ feature virtual

2. Resolva a equação normal

$$X^T X \vec{\theta} = X^T y_{train}$$

Se m = nº de exemplos
n = nº de features original (sem a virtual)

$$\Rightarrow \quad X \in \mathbb{R}^{m \times (n+1)} \qquad \vec{\theta} \in \mathbb{R}^{(n+1) \times 1}$$
$$y_{train} \in \mathbb{R}^{m \times 1}$$

$$\Rightarrow \quad X X^T \in \mathbb{R}^{(n+1) \times (n+1)}$$
$$X^T y_{train} \in \mathbb{R}^{(n+1) \times 1}$$
$$X^T X \vec{\theta} \in \mathbb{R}^{(n+1) \times 1}$$

Custo de solução da equação normal?

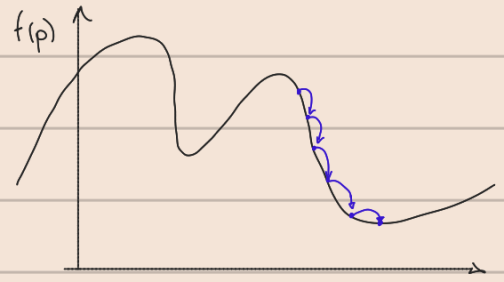
- custo construir $X^T X$ $O(n^2 m)$
- custo construir $X^T y_{train}$ $O(n m)$
- custo resolver sist linear $O(n^3)$

2. "Gradient descent"

Para X_{train} e y_{train} fixos, queremos minimizar

$$L(\vec{\theta}) = MSE(\vec{\theta}; X_{train}; y_{train})$$

gradient descent



1. Chuta valores para $\vec{\theta}$
   $\vec{\theta}$ = chute inicial
2. Enquanto "não convergiu"

$$\vec{\theta} \leftarrow \vec{\theta} - \eta \nabla L(\vec{\theta})$$

novo $\vec{\theta}$ · antigo $\vec{\theta}$ · antigo $\vec{\theta}$ · eta: tx de aprendizagem

Precisamos de $\nabla L(\vec{\theta}) = \nabla MSE(\vec{\theta}; X_{train}, y_{train})$

Felizmente esse gradiente é fácil de calcular no modelo linear:

$$\nabla L(\vec{\theta}) = \frac{2}{m}\left(X^T X \vec{\theta} - X^T y_{train}\right)$$

$X^T X \vec{\theta}$: $O(n m)$

Conclusão: se nº features muito alto → gradient descent
se nº features baixo → eq. normal

"O inimigo do overfitting é a regularização"