

# Introdução a *Machine Learning*

Nina S. T. Hirata  
(nina@ime.usp.br)

Departamento de Ciência da Computação  
Instituto de Matemática e Estatística  
(IME - USP)

Fevereiro / 2017

# Objetivos e roteiro

## Objetivos

- visão geral sobre *machine learning*
- entender o pipeline treinamento-teste
- estudar dois modelos de aprendizagem
- praticar alguns conceitos usando Python & Cia

## Roteiro

- manhã: aula expositiva
- tarde: parte prática

## Onde *machine learning* se encaixa ?

Considere situações nas quais temos

- observações (entrada  $\mathbf{x}$ )

e para as quais pode estar associado

- estado, identidade, ação (saída  $y$ )

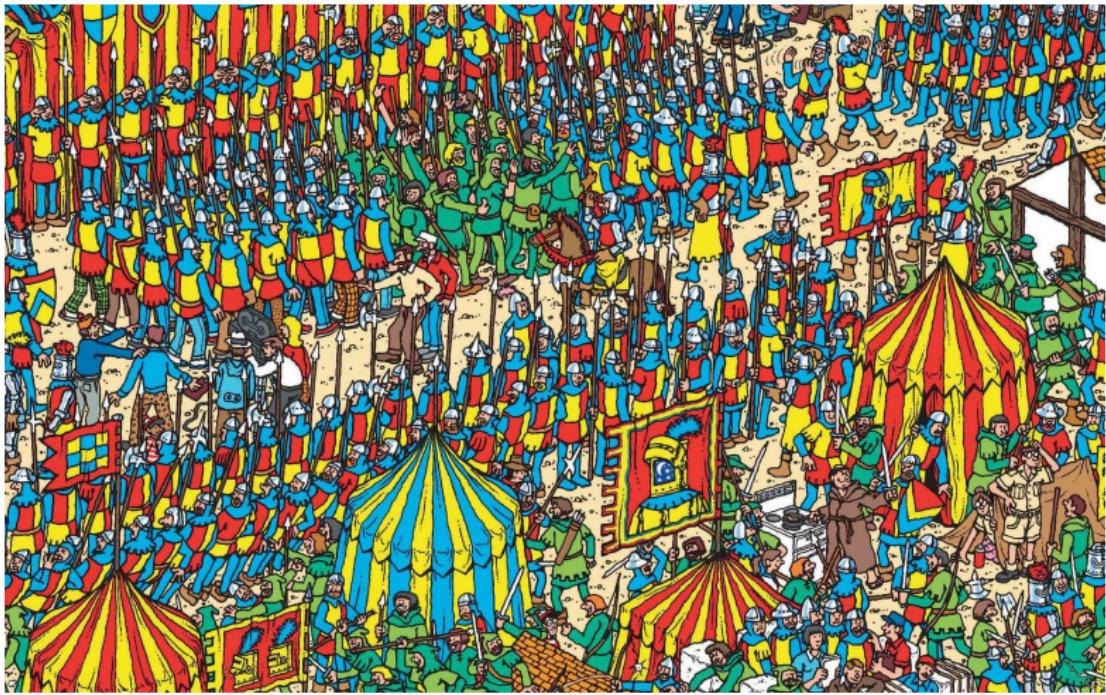
**Queremos:** dada uma observação  $\mathbf{x}$  qualquer, predizer a saída associada da melhor forma possível.

De certa forma, nós humanos, somos muito bons em fazer isso.

## Exemplos

- Reconhecer objetos, pessoas, lugares, etc
- Reconhecer e entender escrita
- Entender falas e reconhecer sons
- Identificar situações que requerem uma ação e tomar uma ação adequada
- Separar o joio do trigo
- Perceber quando alguém não está no melhor dos dias

# Where is Wally ?



## Mais exemplos

Esses envolvem conhecimentos mais específicos:

- diagnóstico médico
- decidir qual é o melhor investimento financeiro
- qual anúncio publicar no meu site ?
- o que deu errado no preparo desse prato ?
- qual candidato será eleito ?

## Para humanos

Algumas das atividades anteriores são *piece of cake*.

Mas muitas delas

- envolvem uma enorme quantidade de dados
- são repetitivas
- são difíceis
- etc

**Machine learning** pode facilitar/auxiliar o processo de análise de dados e tomada de decisões.

# **Big Data**

**Hoje:** Dados em todos os cantos e lados

**Desafios do big data:** os 5 V's

- Volume
- Velocity
- Variety
- Veracity / Variability
- Value

# Valor dos dados

Value of data  $\Rightarrow$  data analysis

**Estatística:** ênfase no ajuste de um modelo aos dados

## Computação:

- construir um modelo a partir dos dados
- Machine Learning and Data mining

Ambos relacionam-se com “**learning from data**”

## Different people doing data analysis ...

**“Machine Learning is AI people doing data analysis”**

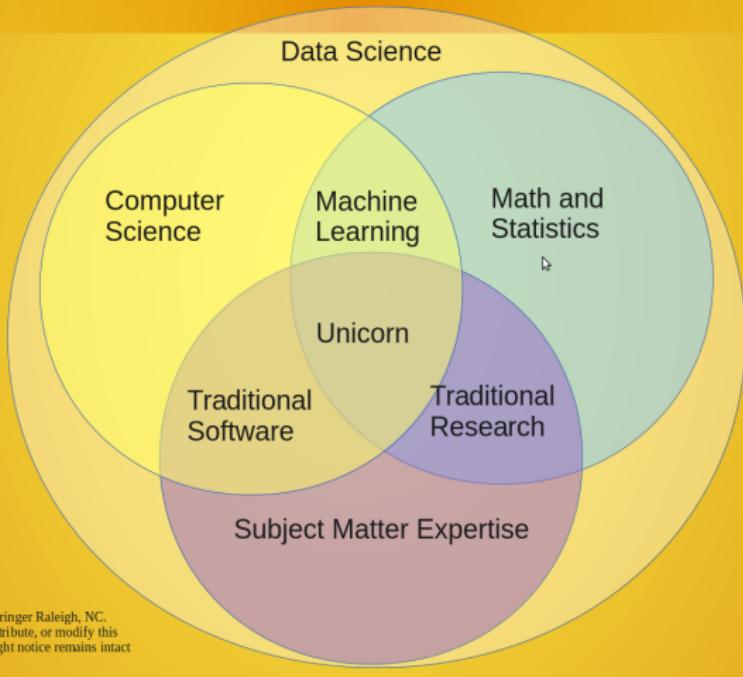
**“Data mining is database people doing data analysis”**

**“Applied Statistics is Statistics people doing data analysis”**

(quotes from Quora)

# Machine Learning = estatística + computação ?

## Data Science Venn Diagram v2.0



# Abordagens em machine learning

	<i>Supervised Learning</i>	<i>Unsupervised Learning</i>
<i>Discrete</i>	classification or categorization	clustering
<i>Continuous</i>	regression	dimensionality reduction

**Daqui em diante,  
apenas supervisionado**

**Entrada:** *features* (atributos)

$$\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$$

**Saída:**

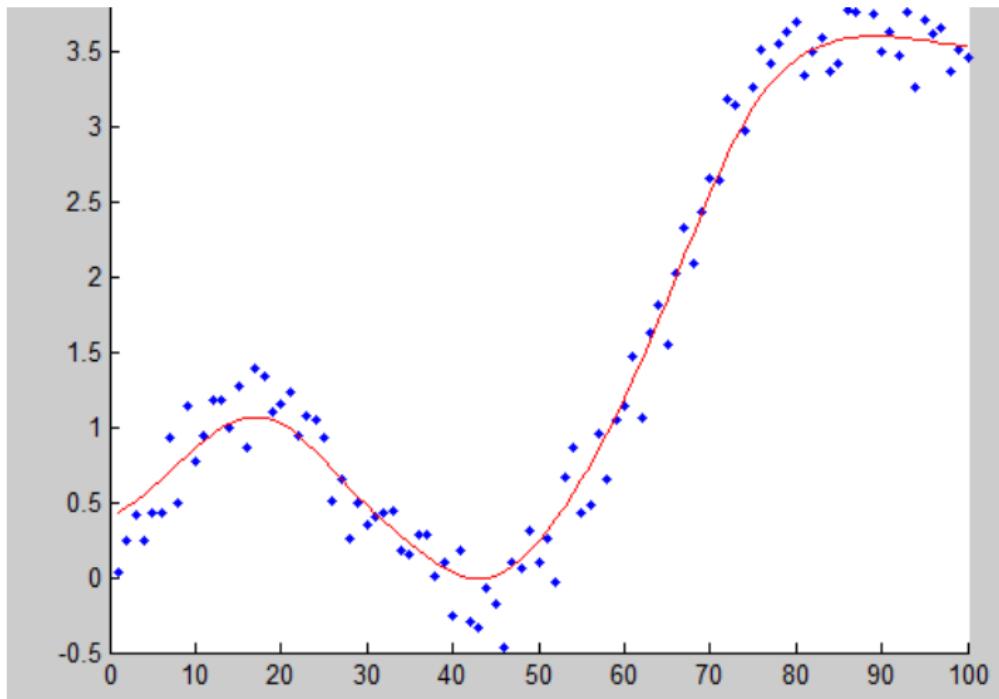
- Classificação: rótulo de classe  $y \in Y = \{1, 2, \dots, c\}$
- Regressão: um valor  $y \in Y = \mathbb{R}$

**Conjunto de exemplos de treinamento**

$$S = \{(\mathbf{x}_i, y_i) : i = 1, 2, \dots, N\}$$

# Regressão

$n = 1$

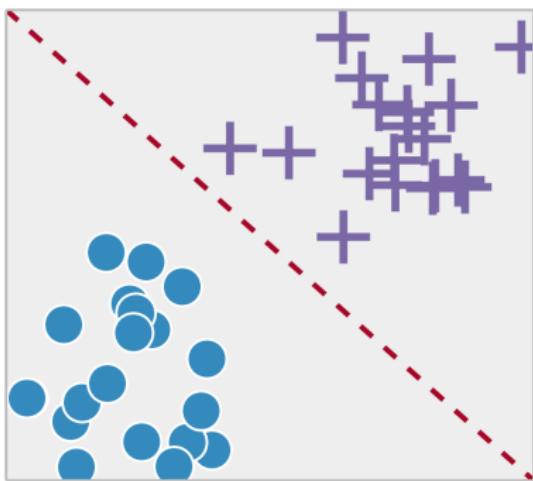


## Classificação × regressão

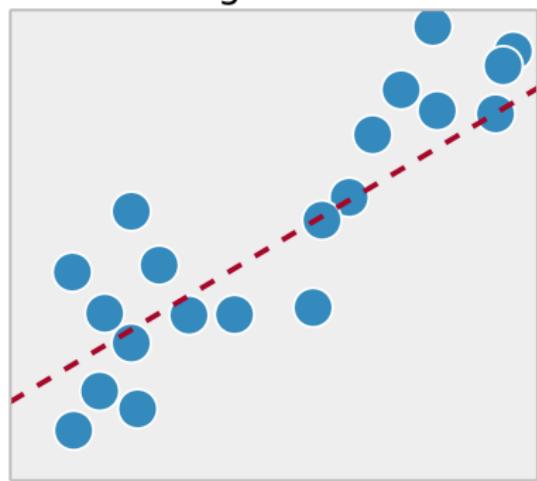
Classificação: encontrar uma fronteira de decisão

Regressão: ajustar uma curva/superfície aos dados

Classification

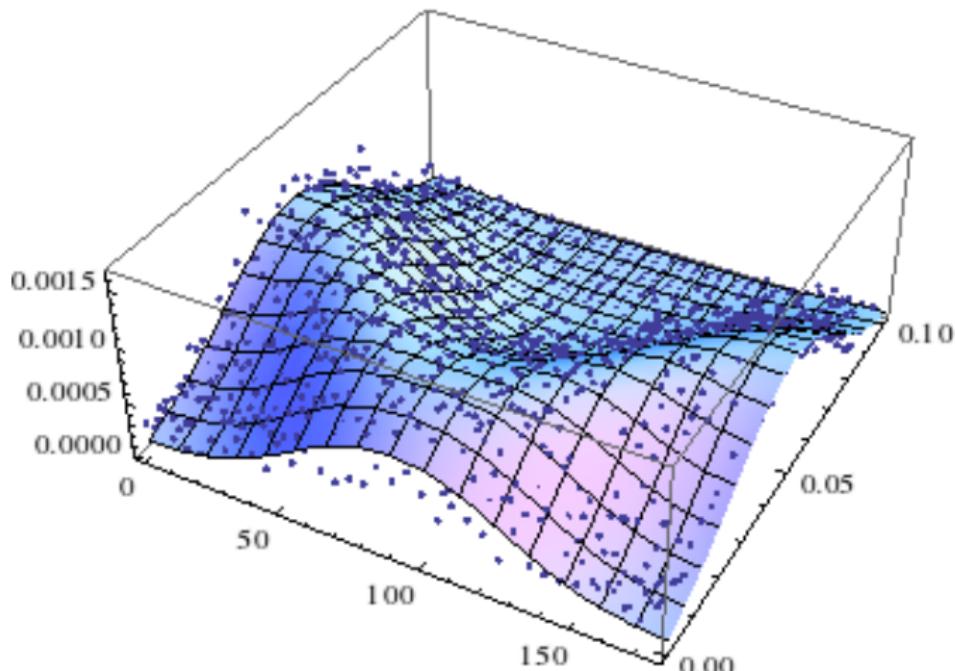


Regression



$n = 2$  e  $n = 1$ , resp

# Regressão

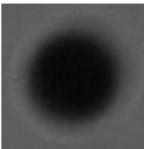


$$n = 2$$

# Exemplo concreto: classificação de plâncton



Appendicularia



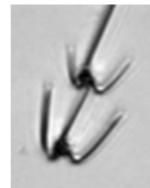
Bubble



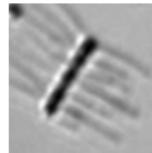
Calanoida



Dinoflagellate

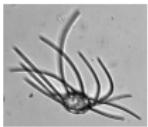


Dinoflagellate



Chaetoceros

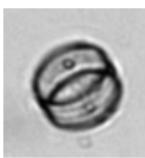
(multiple)



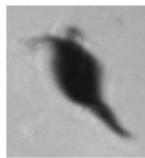
Cnidaria



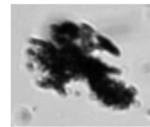
Copepoda  
(no antenna)



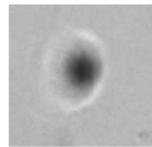
Coscinodiscus



Cyclopoida



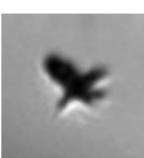
Detritus



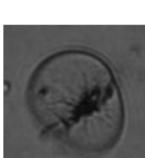
Detritus ball



Filaments



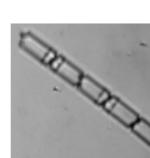
Nauplii



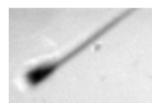
Noctiluca



Penilia



Phyto.

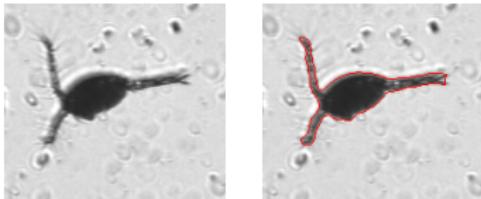


Stalked  
ciliate

Dactyliosolen

# Pipeline

**Segmentar plâncton:** separar na imagem o alvo do fundo (*background*)



**Extrair *features*:** atributos numéricos ou categóricos que descrevem/representam o alvo  $\rightarrow (x_1, x_2, \dots, x_n)$

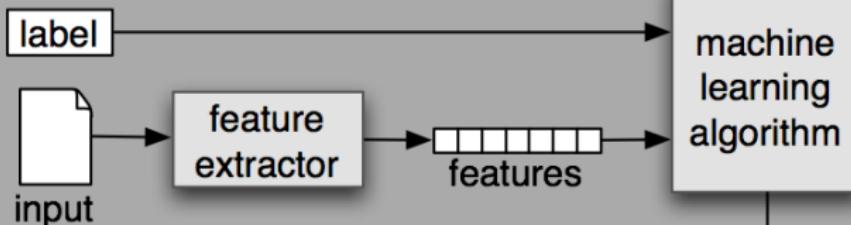
- informação de forma, cor, textura, ...

**Estimar um modelo:** classificador

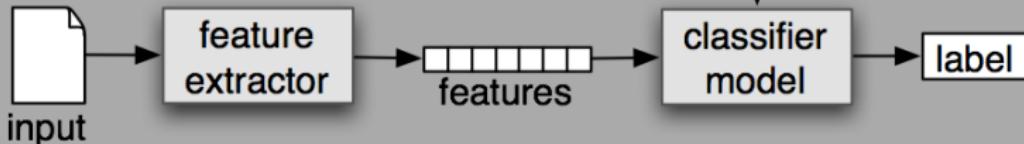
- (no caso supervisionado) processo conhecido como **treinamento**

# Supervised Classification

## (a) Training

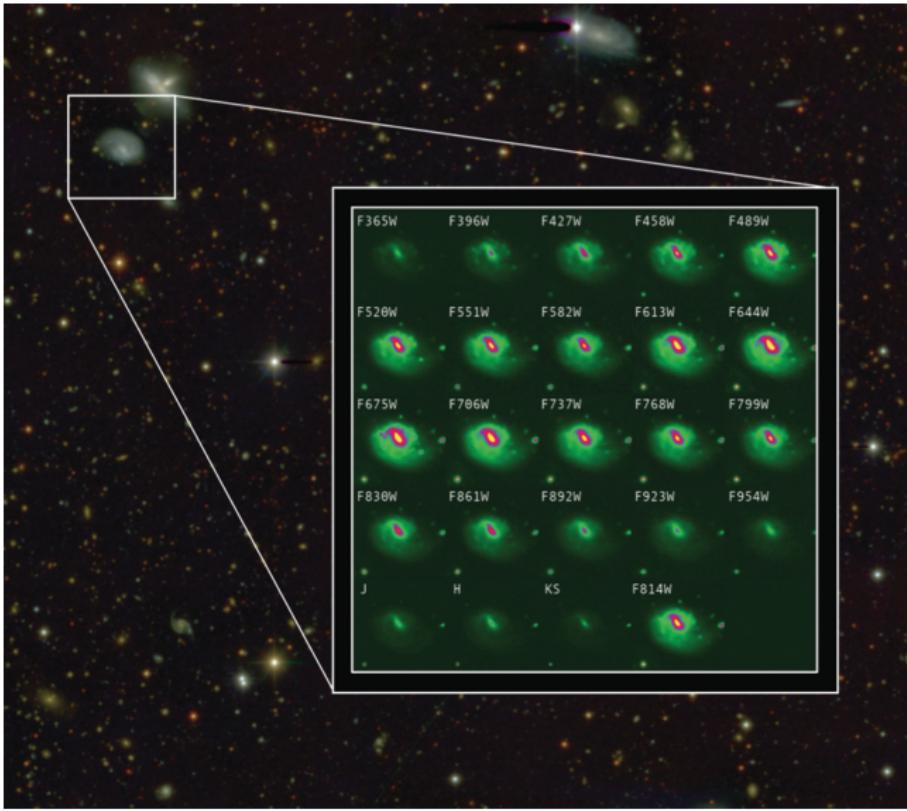


## (b) Prediction



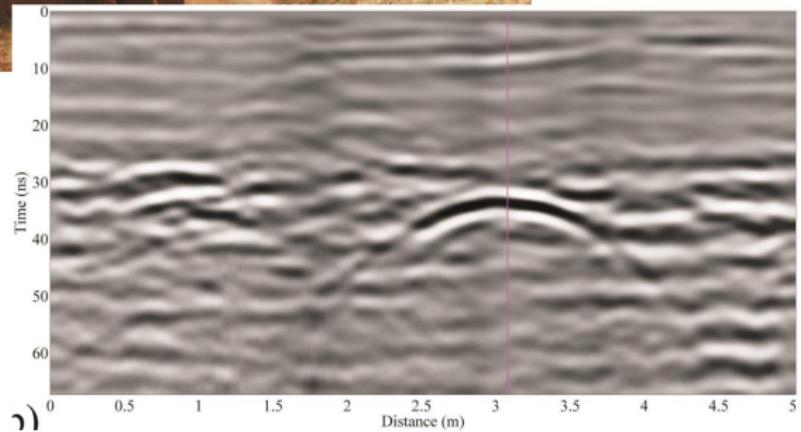
<http://www.nltk.org/book/ch06.html>

## Exemplo concreto: identificação de objetos no espaço



A. Molino *et.al.*, The ALHAMBRA Survey: Bayesian photometric redshifts with 23 bands for 3 deg,

## Exemplo concreto: identificação de objetos no subsolo



**Chega de blah, vamos  
aos fundamentos**

# Notações

$$D = \{(\mathbf{x}_i, y_i) : i = 1, 2, \dots, N\}$$

$\mathbf{x}_i$  é uma entrada e  $y_i$  é a respectiva saída

$$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})$$

## Função linear

Função linear geral no  $\mathbb{R}^n$ :

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n = \sum_{j=0}^n w_jx_j$$

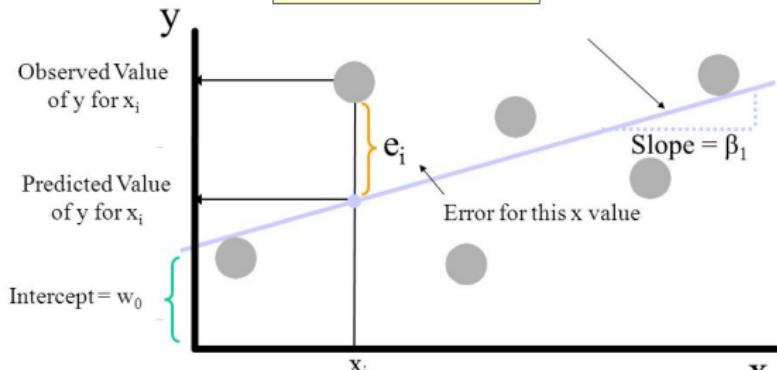
No  $\mathbb{R}^2$ , os pontos onde  $h(x_1, x_2) = 0$  formam uma reta.

$$h(x_1, x_2) = w_0 + w_1x_1 + w_2x_2$$

# Regressão linear

## Linear Regression

$$y = w_0 + w_1 x$$



# Regressão linear

$(x_1, x_2, \dots, x_n) \implies (1, x_1, x_2, \dots, x_n) \in \mathbb{R}^{n+1}$

$\mathbf{w} = (w_0, w_1, \dots, w_n) \in \mathbb{R}^{n+1}$  vetor de pesos

$$h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n = \sum_{j=0}^n w_j x_j$$

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$$

Queremos  $h_{\mathbf{w}}(\mathbf{x}_i) \approx y_i$

Uma ideia simples:

1. chutar um vetor de pesos qualquer  $\mathbf{w}$
2. verificar se a função linear se ajusta adequadamente
3. se sim, terminar
4. senão, alterar ligeriramente a função linear (i.e.,  $\mathbf{w}$ ) e voltar para o passo 2

Mas até quando repetir isso?

Se  $\mathbf{z}_i = h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{x}^T \mathbf{w}$

queremos que a diferença entre  $y_i$  e  $\mathbf{z}_i$  seja a menor possível.

Essa diferença pode ser expressa, por exemplo, por

$$\begin{aligned}SSR_D(\mathbf{w}) &= \sum_{\mathbf{x}_i \in D} (y_i - z_i)^2 \\&= \sum_{\mathbf{x}_i \in D} (y_i - \mathbf{x}^T \mathbf{w})^2\end{aligned}$$

**Problema:** queremos encontrar o ponto  $\mathbf{w}$  que minimiza  $SSR_D$

# Método do gradiente descendente

Função custo

$$J_D(\mathbf{w}) = \frac{1}{2} \sum_{\mathbf{x}_i \in D} (y_i - z_i)^2$$

É uma função quadrática positiva; logo tem ponto de mínimo

**Gradiente descendente:** Método de otimização de função, que consiste em:

- chutar um valor para  $\mathbf{w}$
- calcular o vetor gradiente de  $J_D$  em  $\mathbf{w}$
- alterar  $\mathbf{w}$  na direção oposta ao do vetor gradiente

## Método do gradiente descendente

Vetor gradiente de  $J$ :

$$\nabla J(\mathbf{w}) = \left[ \frac{\partial J}{\partial w_0}, \frac{\partial J}{\partial w_1}, \dots, \frac{\partial J}{\partial w_n} \right]$$

$$\begin{aligned}\frac{\partial J}{\partial w_j} &= \frac{\partial}{\partial w_j} \frac{1}{2} \sum_i (y_i - z_i)^2 \\ &= \frac{1}{2} \sum_i \frac{\partial}{\partial w_j} (y_i - z_i)^2 \\ &= \frac{1}{2} \sum_i 2(y_i - z_i) \frac{\partial}{\partial w_j} (y_i - z_i) \\ &= \sum_i (y_i - z_i) \frac{\partial}{\partial w_j} (y_i - (w_0 + w_1 x_{i1} + \dots + w_j x_{ij} + \dots + w_n x_{in})) \\ &= - \sum_i (y_i - z_i) x_{ij}\end{aligned}$$

# Método do gradiente descendente

Peso inicial:  $\mathbf{w}(0)$

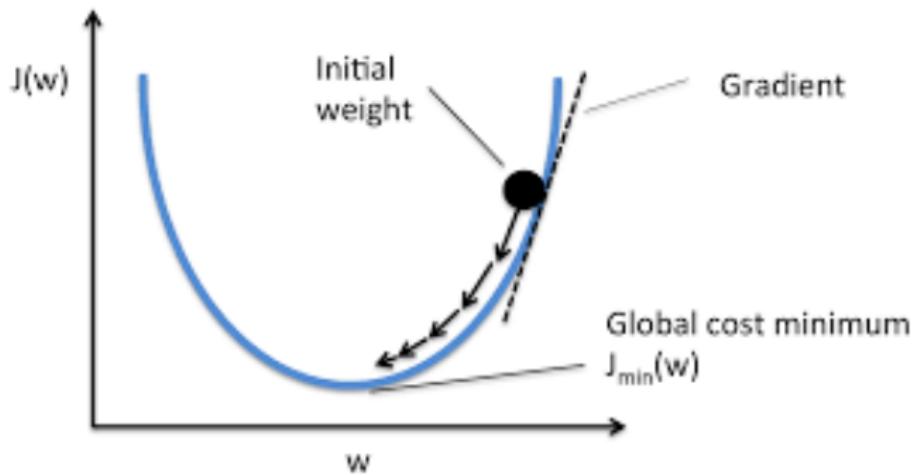
Regra de atualização:

$$\mathbf{w}(r+1) = \mathbf{w}(r) + \eta \Delta \mathbf{w}(r)$$

$$\Delta \mathbf{w}(r) = -\nabla J(\mathbf{w})$$

$$\Delta w_j(r) = \sum_i (y_i - z_i) x_{ij}$$

# Método do gradiente descendente



## Regressão linear: resumo

- Ajustar uma função linear aos dados pode ser modelado como um problema de minimizar uma função custo  $J_D$  expressa em termos da soma do quadrado dos resíduos (RSS).
- Para cada função linear, caracterizado por um vetor de pesos  $\mathbf{w}$  específico, há um custo  $J_D(\mathbf{w})$  associado
- O ponto de mínimo pode ser calculado usando-se o método do gradiente descendente

# Classificação

**Problema que queremos resolver:** dado um conjunto de amostras, encontrar uma fronteira de decisão que melhor separe as duas classes.

**Pontos a serem considerados:**

- que tipo de fronteira de decisão ?
- qual o significado de “melhor separação” ?
- quantas classes são ?
- etc

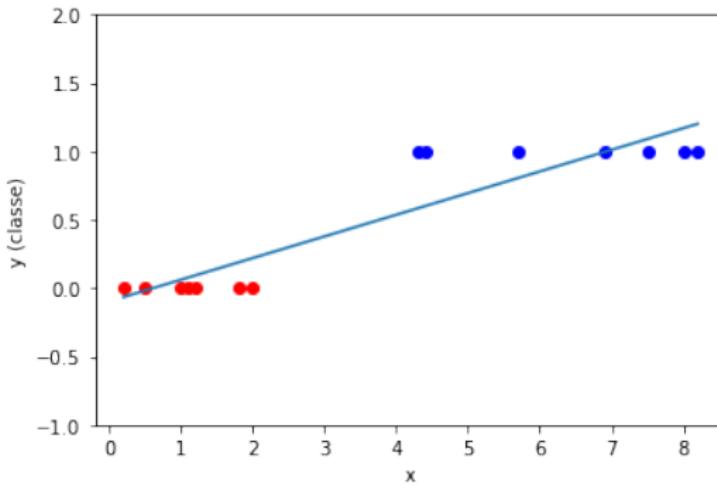
## Classificação

Podemos aplicar a regressão linear e decidir como segue ?

$$z = \begin{cases} 1, & \text{se } h_{\mathbf{w}}(\mathbf{x}) > 0, \\ 0, & \text{se } h_{\mathbf{w}}(\mathbf{x}) \leq 0. \end{cases}$$

**Pergunta:** é uma boa solução ?

Parece que não ...

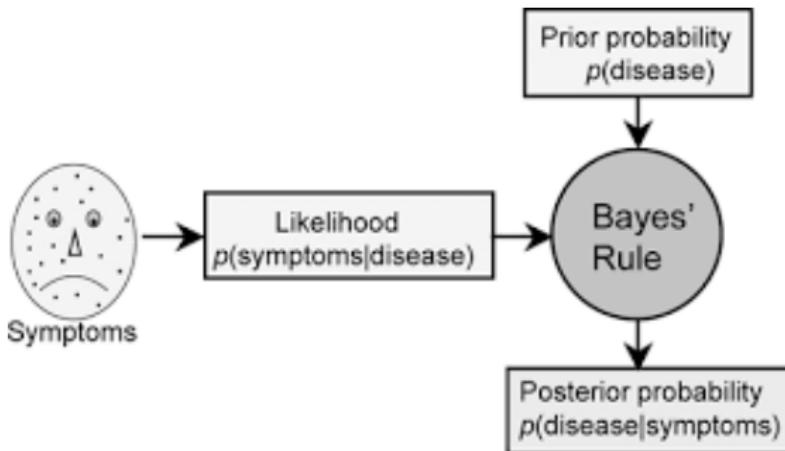


Em problemas de classificação, podemos usar a regressão logística

**Mas antes vamos ver a  
regra de Bayes**

## Regra de Bayes

$$P(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)P(y)}{p(\mathbf{x})}$$



$$P(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)P(y)}{p(\mathbf{x})}$$

**Se as distribuições fossem conhecidas**, seria possível escolher  $y$  com maior  $P(y|\mathbf{x})$

É sabido que, em termos probabilísticos, **o erro de predição é ótimo (mínimo) se tal escolha é feita.**

$$P(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)P(y)}{p(\mathbf{x})}$$

Em geral, **não conhecemos essas distribuições**.

### **Abordagens generativas:**

modelam  $p(\mathbf{x}|y)$  e estimam  $P(y)$ , a partir das quais pode calcular  $P(y|\mathbf{x})$

### **Abordagens discriminativas:**

estimam  $P(y|\mathbf{x})$  diretamente

## Abordagem discriminativa

No caso de duas classes

$y = 0$  : exemplo negativo

$y = 1$  : exemplo positivo

Queremos encontrar alguma função  $h$  tal que

$$P(y = 1 | \mathbf{x}) > P(y = 0 | \mathbf{x}) \implies h(\mathbf{x}) > 0$$

$$P(y = 1 | \mathbf{x}) < P(y = 0 | \mathbf{x}) \implies h(\mathbf{x}) \leq 0$$

# Regressão logística

Se tivermos

$$g_1(\mathbf{x}) \approx P(y = 1 | \mathbf{x})$$

$$g_0(\mathbf{x}) = 1 - g_1(\mathbf{x}) \approx P(y = 0 | \mathbf{x})$$

E fizermos

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_0(\mathbf{x})$$

temos

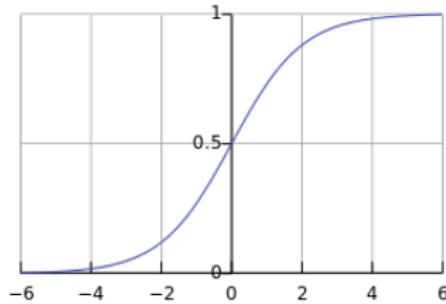
$$g(\mathbf{x}) > 0 \iff P(y = 1 | \mathbf{x}) > P(y = 0 | \mathbf{x})$$

$$g(\mathbf{x}) < 0 \iff P(y = 1 | \mathbf{x}) < P(y = 0 | \mathbf{x})$$

# Regressão logística

Usa a função sigmoide:

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



$$0 \leq s(x) \leq 1$$

# Regressão logística

$$h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T \mathbf{w} = \sum_{j=0}^n w_j x_j$$

$$g_1(\mathbf{x}) = s(h_{\mathbf{w}}(\mathbf{x})) = \frac{1}{1 + e^{-h_{\mathbf{w}}(\mathbf{x})}}$$

$$g_0(\mathbf{x}) = 1 - g_1(\mathbf{x})$$

$$h_{\mathbf{w}}(\mathbf{x}) = 0 \implies g_0(\mathbf{x}) = g_1(\mathbf{x}) = 0.5$$

$$h_{\mathbf{w}}(\mathbf{x}) < 0 \implies g_0(\mathbf{x}) > g_1(\mathbf{x})$$

$$h_{\mathbf{w}}(\mathbf{x}) > 0 \implies g_0(\mathbf{x}) < g_1(\mathbf{x})$$

Ou ainda

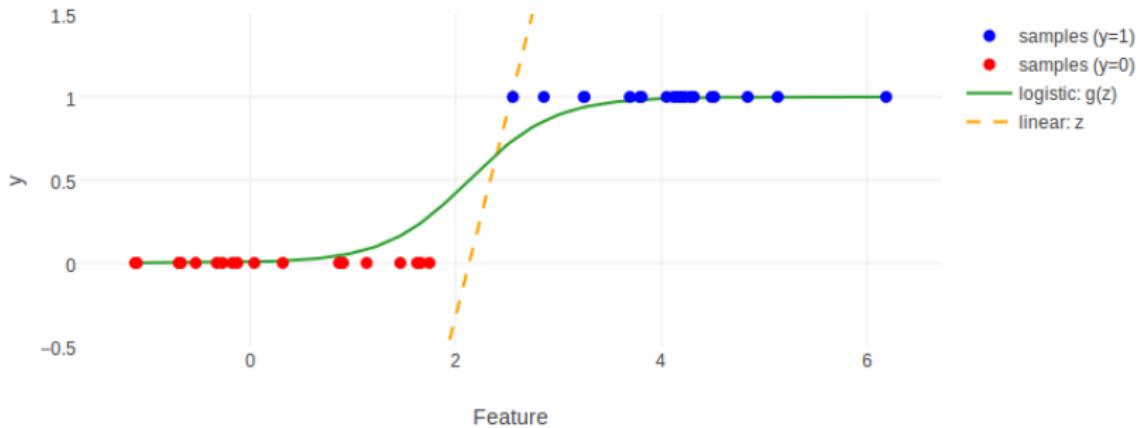
$$g_0(\mathbf{x}) = g_1(\mathbf{x}) = 0.5 \iff h_{\mathbf{w}}(\mathbf{x}) = 0$$

$$g_1(\mathbf{x}) < 0.5 \iff h_{\mathbf{w}}(\mathbf{x}) < 0$$

$$g_1(\mathbf{x}) > 0.5 \iff h_{\mathbf{w}}(\mathbf{x}) > 0$$

O vetor de pesos  $\mathbf{w}$  define uma fronteira de decisão linear.

## Logistic Regression: 1 Feature



## Como encontrar o vetor de pesos w?

Função de verossimilhança

$$L(\mathbf{w}) = \prod_{i=1}^N P(y_i | \mathbf{x}_i, \mathbf{w}) = \prod_{i=1}^N [g_1(\mathbf{x}_i)]^{y_i} [1 - g_1(\mathbf{x}_i)]^{(1-y_i)}$$

O ponto de máximo dessa função é o ponto de mínimo da função

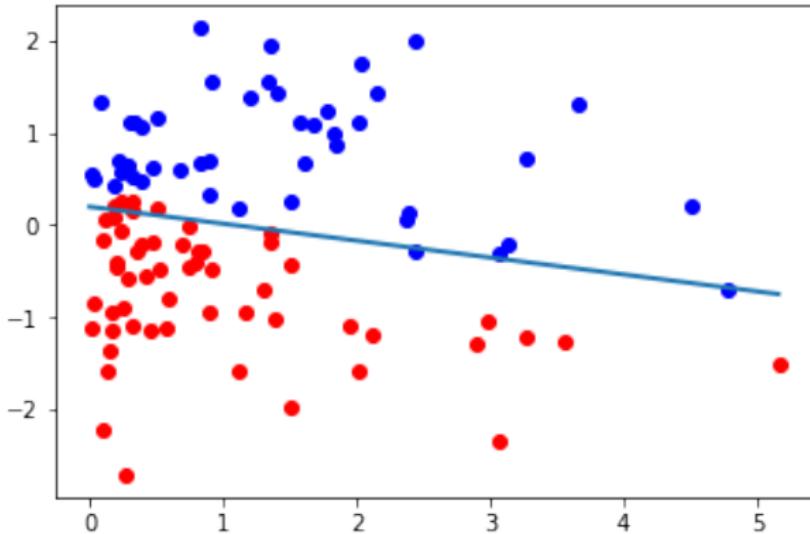
$$l(\mathbf{w}) = -\ln L(\mathbf{w})$$

Assim, pelo método do gradiente descendente obtém-se:

$$\Delta w_j(r) = \sum_{i=1}^N (y_i - g_1(\mathbf{x}_i)) \mathbf{x}_{ij}$$

## Regressão logística: resumo

- Usado para encontrar uma fronteira de decisão linear para problemas de classificação
- O valor  $h_{\mathbf{w}}(\mathbf{x})$  é modulado pela sigmoide  $s$  de forma que fique no intervalo  $[0, 1]$  antes do cálculo da distância até  $y$
- O vetor de peso  $\mathbf{w}$  pode ser obtido aplicando-se o método do gradiente descendente ao log da função de verossimilhança
- A fronteira de decisão resultante é linear



# Fronteiras de decisão não lineares

1. Criar novas *features*

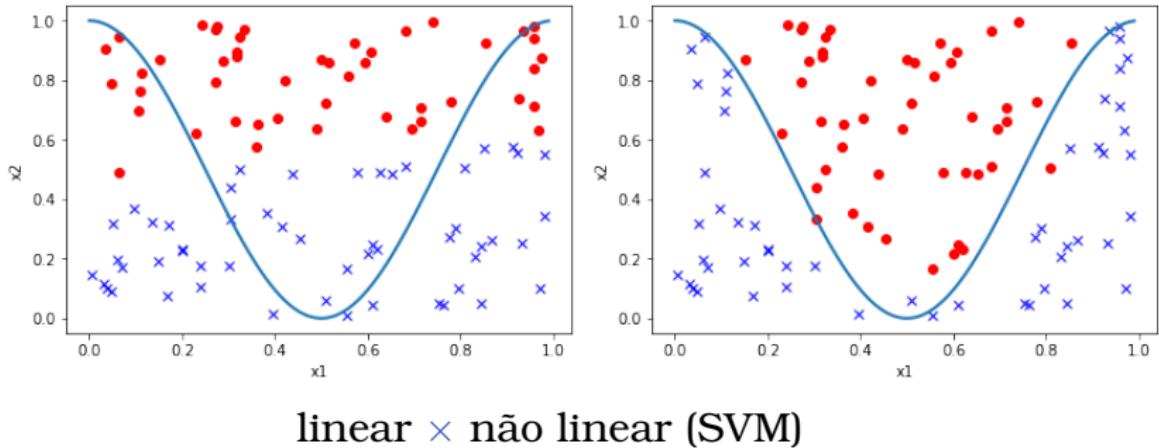
**Exemplo:**  $(1, x_1, x_2) \Rightarrow (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$

$n = 2$  passa para  $n' = 5$

e aplicar a regressão logística

2. Usar modelos não-lineares (ex.: rede neural)

# Fronteiras de decisão não lineares



# Outros tópicos

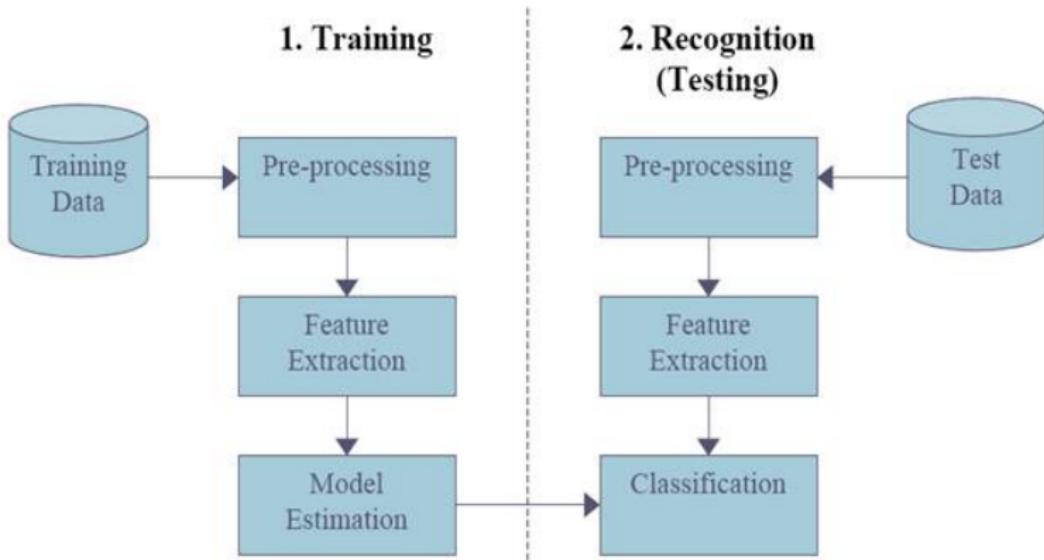
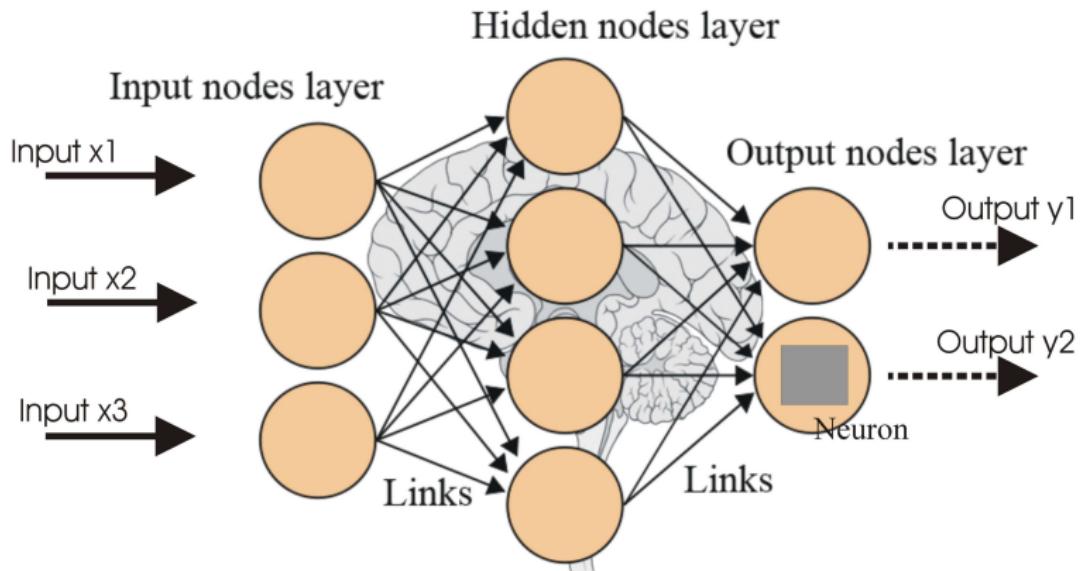


Figure 2: The pattern classification process

- Treinamento - Validação - Teste
- avaliação de desempenho
- Vários modelos: SVM, KNN, DT, NN, etc
- Seleção de modelos
- Quantos e quais *features* ?
- Seleção de features
- Múltiplas classes

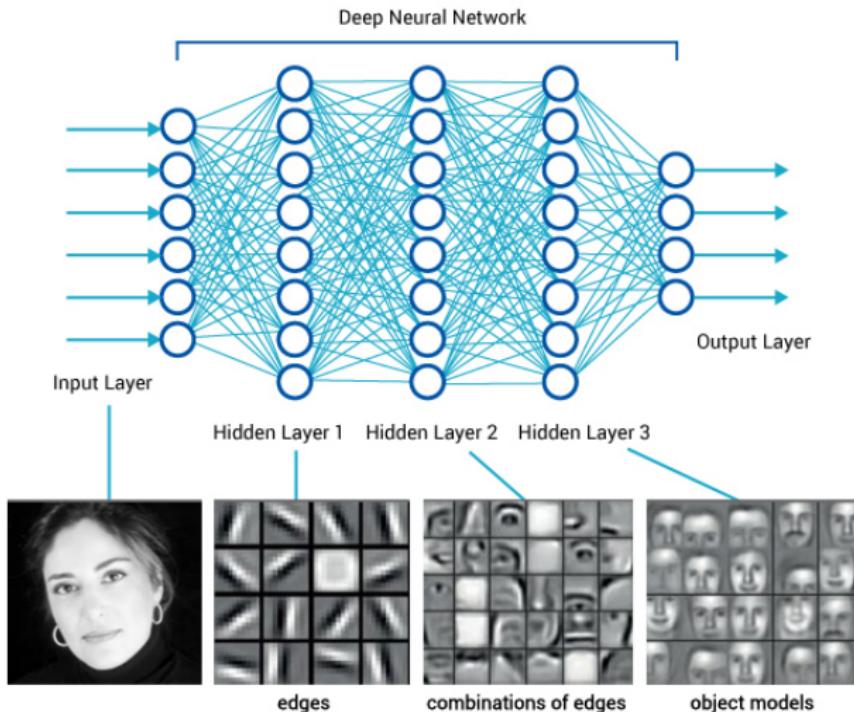
# Deep learning

Evolução de neural nets:



# Deep learning

Capaz de fazer *feature learning*



## Comentários

- só falamos do aprendizado supervisionado
- regressão linear e regressão logística
- parte da tarde: oportunidade para experimentar esses métodos na prática
- para aprender NN e CNN,  
[neuralnetworksanddeeplearning.com](http://neuralnetworksanddeeplearning.com)