

## UNIVARIATE DATA ANALYSIS

$$H = \frac{n}{\frac{1}{n_1} + \frac{1}{n_2} + \cdots + \frac{1}{n_n}}$$

Harmonic Mean

$$\sigma_{\text{population}} = \sqrt{\frac{1}{n} \sum_{i=1}^n (n_i - \bar{x})^2}$$

desvio padrão

$$r = \frac{\text{cov}(y_1, y_2)}{\sqrt{\text{var}(y_1)} \sqrt{\text{var}(y_2)}} = \frac{\sum y_1 y_2 - \frac{\sum y_1 \sum y_2}{n}}{\left( \sum y_1^2 - \frac{(\sum y_1)^2}{n} \right) \left( \sum y_2^2 - \frac{(\sum y_2)^2}{n} \right)}$$

correlação de Pearson

## DECISION TREES AND MODULE EVALUATION

1. calcular  $IG(y_k)$  para cada feature  $y_k$
2. Escolher  $y_k$  com maior  $IG$  para meter no nó da árvore
3. Repetir para as sub-árvore de baixo de cada ramo

$$H(\text{class}) = - \sum_{\substack{c=\text{class} \\ \uparrow \text{entropy}}} p(\text{class}=c) \cdot \log_2(p(\text{class}=c))$$

$$H(\text{class} | y_k) = \sum_{j \in y_k} \left( p(y_k=j) \times H(\text{class} | y_k=j) \right)$$

$$IG(y_k) = H(\text{class}) - H(\text{class} | y_k)$$

$$\text{Accuracy} = \frac{TP + TN}{\text{All}}$$

$$\text{Error rate} = 1 - \text{accuracy}$$

$$\text{Recall/sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Fallout/specificity} = \frac{TN}{TN + FP}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$F1\text{-score} = \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \times 2$$

## PROBABILITY THEORY

Gaussian distribution probability density function

$$pdf = \frac{1}{\sigma \sqrt{2\pi}} \times e^{-\frac{1}{2} \left( \frac{n-\mu}{\sigma} \right)^2}$$

Multivariate Gaussian distribution p.d.f.

$$pdf = \frac{1}{(2\pi)^{n/2} \times |\Sigma|^{1/2}} \times e^{-\frac{1}{2} (n-\mu)^T \Sigma^{-1} (n-\mu)}$$

$\downarrow$  determinante

Bayes Rule

$$P(\text{classe}=c | n) = \frac{P(n | \text{classe}=c) \times P(\text{classe}=c)}{P(n)}$$

↳ Naive Bayes: features do dataset são independentes

$$P(\text{classe}=c | n) = \frac{P(y_1=n_1 | \text{classe}=c) \times \cdots \times P(y_p=n_p | \text{classe}=c) \times P(\text{classe}=c)}{P(n)}$$

## KNN AND EVALUATION

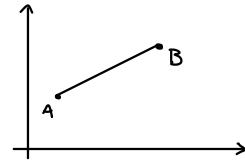
Descobrir o output de uma observação com base nas K observações mais próximas

1. Calcula a distância de  $x_i$  a todas as observações do dataset
2. Escolho as K mais próximas
3. O output de  $x_i$  vai ser
  - a média do output dos K vizinhos se o output é contínuo
  - a moda dos K vizinhos se o output for categórico

### DISTÂNCIAS

#### Euclidiana

$$d(a, b) = \sqrt{\sum_{i=1}^p (a_i - b_i)^2}$$

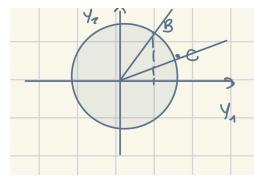


#### Manhattan

$$d(a, b) = \sum_{i=1}^p |a_i - b_i|$$

(em vez de andarmos para qualquer lado p.e. se só andarmos pra cima e pra baixo = soma das retas)

#### Similaridade do cosseno



$$\cos(a, b) = \frac{a \cdot b}{\|a\| \times \|b\|}$$

quanto maior o cosseno  
mais perto eles estão

#### Hamming

$$\sum_{i=1}^p a_i \neq b_i$$

ex:  $n_1 = \begin{bmatrix} A & A & B \end{bmatrix}$   
 $n_2 = \begin{bmatrix} B & A & B \end{bmatrix}$

hamming ( $n_2, n_1$ ) = 1

### FEATURES CATEGÓRICAS

**DISTÂNCIAS PESADAS** dão mais importância aos vizinhos mais próximos

$$w_i = \frac{1}{d(n_i, n)} \quad \text{OU } \underline{\underline{1}} \text{ se } n_i = n$$

↑ peso do vizinho  $i$

- PARA CLASSIFICAR  $\text{output}_n = \text{moda} (w_i \times \text{target}(n_i)) \quad \forall n_i \in \text{ENN}$
- PARA REGRESSÃO  $\text{output}_n = \frac{\sum w_i \times \text{target}(n_i)}{\sum w_i}, \quad \forall n_i \in \text{ENN}$   
↳ média ponderada

### AVALIAR MODELOS DE REGRESSÃO

MAE :

$$\sum_{i=1}^n |z_i - \hat{z}_i|$$

↑ output real      ↑ output previsto

MSE :

$$\frac{\sum_{i=1}^n (z_i - \hat{z}_i)^2}{N}$$

SSE :

$$\sum_{i=1}^n \sum_{j=1}^n (z_{ij} - \hat{z}_{ij})^2$$

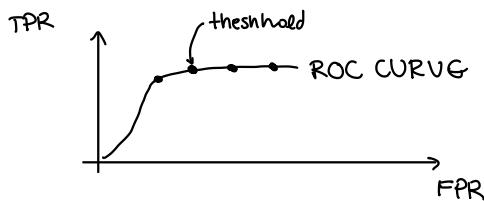
RMSSE :

$$\sqrt{\frac{\sum_{i=1}^n (z_i - \hat{z}_i)^2}{n}}$$

ROC Curve : true positive rate vs false positive rate

True positive rate = TPR = Recall =  $\frac{TP}{TP + FN}$

False positive rate = FPR =  $\frac{FP}{FP + TN}$

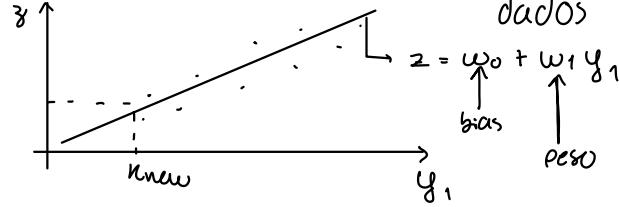


⇒ Queremos maximizar TPR e minimizar FPR

# LINEAR REGRESSION

algoritmo que prevê outputs contínuos

**Regressão linear**: encontram uma reta que melhor se ajuste aos dados



Em geral observação  $\begin{bmatrix} n_{i1} \\ n_{ip} \end{bmatrix}$

$$\hat{z}_i = \mathbf{w}^T \cdot \mathbf{n}_i = w_0 + w_1 n_{i1} + w_2 n_{i2} + \dots + w_p n_{ip} \leftarrow 1 \text{ observação } n_i$$

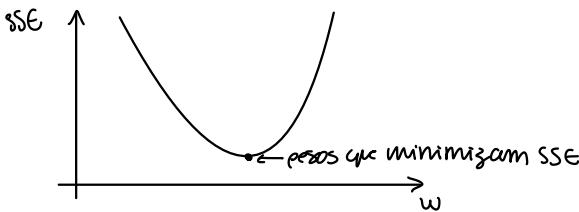
$$\hat{\mathbf{z}} = \mathbf{X} \mathbf{w} = \begin{bmatrix} 1 & n_{11} & n_{12} & \dots & n_{1p} \\ 1 & n_{21} & n_{22} & \dots & n_{2p} \\ \vdots & & & & \vdots \\ 1 & n_{N1} & n_{N2} & \dots & n_{Np} \end{bmatrix}_{N \times p} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{bmatrix}_{p \times 1} = \begin{bmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \vdots \\ \hat{z}_N \end{bmatrix}_{N \times 1}$$

cost function

$$\text{Sum squared errors} = \text{SSE} = \sum_{i=1}^n (\hat{z}_i - z_i)^2$$

$$\hat{z}_i = \mathbf{w}^T \cdot \mathbf{n}_i$$

objetivo: encontram os pesos que minimizam o erro

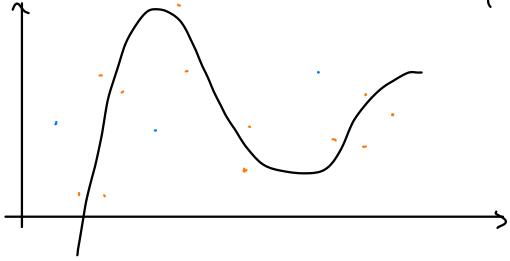


$$\frac{\partial \text{SSE}}{\partial \mathbf{w}} = 0 \quad (\Leftarrow)$$

$$\Leftarrow \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{z}$$

MATRIZ MOORE-PENROSE

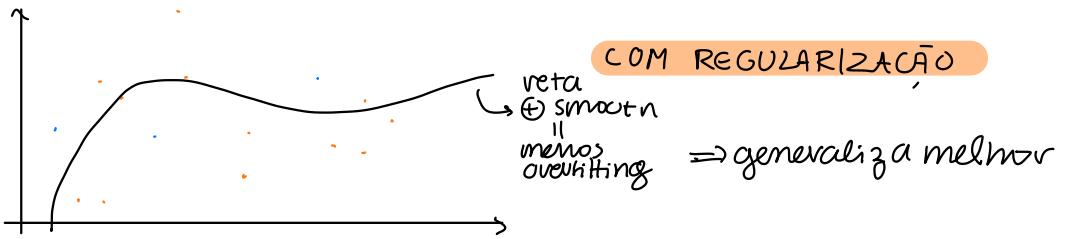
Regularização: método usado para reduzir overfitting (retirar importância a outliers)



■ - dados teste  
■ - dados treino

$\Rightarrow$  erros teste > erros treino

↳ para balancear,  $\uparrow$  erro treino e  $\downarrow$  erro teste



COM REGULARIZAÇÃO

reta + smooth  
menos overfitting  $\Rightarrow$  generalização melhor

↳ em retas, é objetivo diminuir o declive

### Regularização L<sub>2</sub> / Ridge

$$\text{cost function}_{\text{Ridge}} = \text{ERRO} + \frac{\lambda}{2} \sum_{j=1}^p w_j^2$$

shrinkage factor

$\uparrow$  shrinkage  $\downarrow$  os pesos  
 $\&$  shrinkage = 0

↳ regularização normal

matriz identidade

$$\frac{\partial \text{cost function}_{\text{Ridge}}}{\partial w} = 0 \Leftrightarrow w = (x^T x + \lambda I)^{-1} x^T z$$

### Regularização Lasso

$$\text{cost function}_{\text{Lasso}} = \text{ERRO} + \frac{\lambda}{2} \sum_{j=1}^p |w_j|$$

threshold : dividir classificação em acima do threshold e abaixo

Espaco features originais :

$$\phi(y_1) = y_1^2 \quad \phi(y_2) = y_2^2$$

$$\hat{z} = w_0 + w_1 y_1^2 + w_2 y_2^2$$

é uma curva

Espaco transformado

$$z = w_0 + w_1 y_1^2 + w_2 y_2^2$$

é um plano

$$y_1^2 \rightarrow y_1^2$$

isto é agora um plano  
pois temos novas  
features que não  
estão no quadrado

$$y_2^2 \rightarrow y_2^2$$

→ com features transformadas, conseguimos tornar as observações linearmente separadas, de forma a conseguirmos resolver problema de classificação

→ regressão : encontrou reta que segue pontos

→ classificação : encontrou uma decisão que separe classes

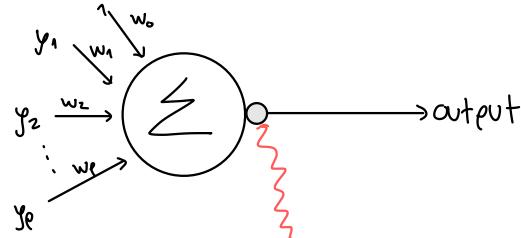
$$w_0 + w_1 y_1 + w_2 y_2 = \text{threshold}$$

$$\underline{\text{Espace original}} : w_0 + w_1 y_1^2 + w_2 y_2^2 = \text{threshold}$$

$$\underline{\text{Espace transformado}} : w_0 + w_1 y_1^2 + w_2 y_2^2 = \text{threshold}$$

# PERCEPTRON

## Perceptrão

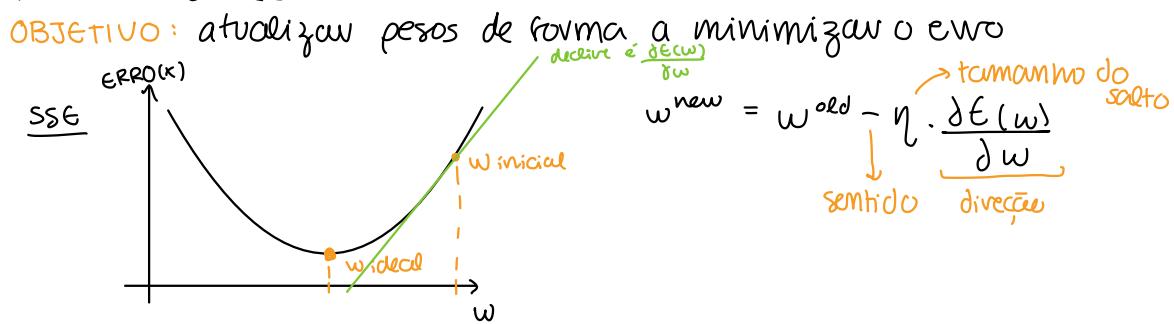


$$\delta(n) = w^T \cdot n = w_0 + w_1 y_1 + \dots + w_p y_p$$

ativacão ex:  $\text{sign}(f(x_i)) = \begin{cases} +1, & \text{se } \delta(n) \geq 0 \\ -1, & \text{se } \delta(n) < 0 \end{cases}$

1. Inicializar pesos e definir learning rate  $\eta$
2. Escolher uma observação  $n_i$
3. Calcular output de  $n_i \rightarrow z_i = \text{ativacão}(r(n_i)) = \text{ativacão}(w^T \cdot n_i)$
4. Se  $z_i \neq \hat{z}_i$  (previsão errada)  $\rightarrow$  ATUALIZAR PESOS  
 $w_{\text{new}} = w_{\text{old}} + \eta (z_i - \hat{z}_i) n_i$
5. Repetir passos 1-4 para todas as observações (corresponde à época)  
ou repete por todo o dataset
6. No fim de uma época  $\rightarrow$  se houve updates, repete mais 1 época  
 $\rightarrow$  se não houve updates, convergiu e paramos

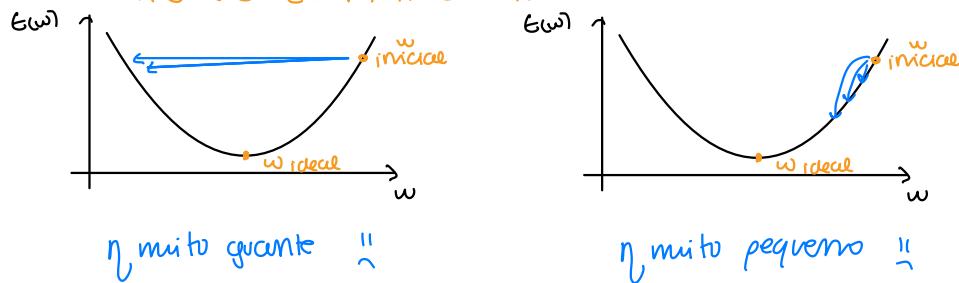
# GRADIENT DESCENT



## TIPOS:

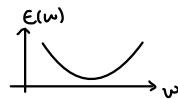
- Batch: usamos todas as observações para fazer update
- Mini-Batch: usamos um subset de observações
- Stochastic: usamos 1 observação

## EFEITO DO LEARNING RATE:

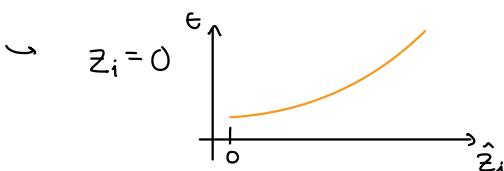
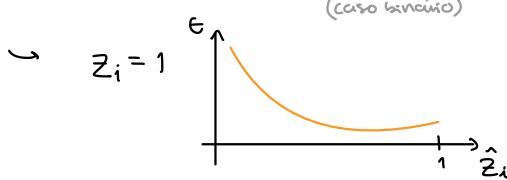


## FUNÇÕES DE ERRO:

Regressão  $\rightarrow$   $SSE = \sum_{i=1}^N (z_i - \hat{z}_i)^2$



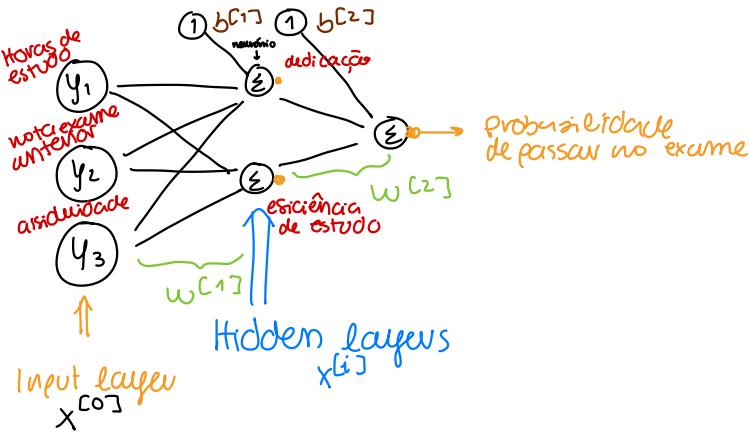
Classificação  $\rightarrow$  CROSS-ENTROPY =  $-\sum_{i=1}^N z_i \cdot \log_2(\hat{z}_i) + (1-z_i) \cdot \log_2(1-\hat{z}_i)$   
(caso binário)



$$\text{Ewo} \rightarrow \text{cross-entropy} \text{ (binária)} = - \sum_{i=1}^N z_i \ln(\hat{z}_i) + (1-z_i) \ln(1-\hat{z}_i)$$

# REDES NEURONAIAS

(aprendem novas features)



$$x^{[0]} \xrightarrow[w^{[1]} b^{[1]}] z^{[1]} \xrightarrow{\text{activation}(z^{[1]})} x^{[1]} \longrightarrow \dots \longrightarrow z^{[\text{out}]} \xrightarrow{} x^{[\text{out}]}$$

$$z^{[i]} = w^{[i]} \cdot x^{[i-1]} + b^{[i]}$$

$$x^{[i]} = \text{activation}(z^{[i]})$$

Porque usar ativações?

$$x^{[1]} = z^{[1]} = w^{[1]} \cdot x^{[0]} + b^{[1]}$$

$$x^{[2]} = z^{[2]} = w^{[2]} \cdot x^{[1]} + b^{[2]} =$$

$$= \underbrace{w^{[2]} w^{[1]} x^{[0]}}_{w} + \underbrace{w^{[2]} b^{[1]} b^{[2]}}_{b}$$

apenas aprendemos combinações lineares



Forward propagation

$$z^{[i]} = w^{[i]} \cdot x^{[i-1]} + b^{[i]}$$

$$x^{[i]} = \phi(z^{[i]})$$

## Back Propagation

Queremos minimizar o erro

(t é o target real)



- $\frac{1}{2} SSE = \frac{1}{2} \sum_{i=1}^N (x^{[out](i)} - t^{[i]})^2 = \frac{1}{2} \|x^{[out]} - t\|^2$

- $\text{cross-entropy} = \sum_{i=1}^N \sum_{c=1}^{C_i} t_c^{(i)} \cdot \log(x_c^{[out](i)})$

→ updates  $w^{[i]} = w^{[i]} - \eta \cdot \frac{\partial E}{\partial w^{[i]}}$

$$b^{[i]} = b^{[i]} - \eta \cdot \frac{\partial E}{\partial b^{[i]}}$$

→ derivadas  $\frac{\partial E}{\partial w^{[i]}} = \frac{\partial E}{\partial h^{[i]}} \circ \underbrace{\frac{\partial h^{[i]}}{\partial z^{[i]}}}_{g^{[i]}} \cdot \left( \frac{\partial z^{[i]}}{\partial w^{[i]}} \right)^\top$  hadamard

$$\frac{\partial E}{\partial b^{[i]}} = g^{[i]} \cdot 1 = g^{[i]}$$

↳ calcular  $\delta^{[i]}$

- última layer :

Se erro = SSE :

$$\delta^{[\text{out}]} = \frac{\partial E}{\partial x^{[\text{out}]}} \circ \frac{\partial x^{[\text{out}]} }{\partial z^{[\text{out}]}} = \\ = (x^{[\text{out}]} - t) \circ \frac{\partial x^{[\text{out}]} }{\partial z^{[\text{out}]}}$$

Se erro = CROSS ENTROPY e ativação softmax :

$$\delta^{[\text{out}]} = x^{[\text{out}]} - t$$

- layer intermediária

regra da cadeia  
↓

$$\delta^{[i]} = \frac{\partial E}{\partial x^{[i]}} \circ \frac{\partial x^{[i]} }{\partial z^{[i]}} = \left( \left( \frac{\partial z^{[i+1]}}{\partial x^{[i]}} \right)^T \cdot \delta^{[i+1]} \right) \circ \frac{\partial x^{[i]} }{\partial z^{[i]}}$$

||  
 $(w^{[i+1]})^T$

## Funções de ativação

- output layer

↪ sigmoid (classificação binária)

$$\frac{1}{1 + e^{z^{[\text{out}]}}}$$

$z_c^{[\text{out}]}$

↪ softmax (classe não binária)

$$P(\text{class} = c) = \frac{e^{z_c^{[\text{out}]}}}{\sum_{j=1}^{|\mathcal{C}|} e_j^{[\text{out}]}}$$

↪ linear / Relu (regressão)

$$\begin{cases} z^{[\text{out}]}, & z^{[\text{out}]} > 0 \\ 0, & \text{otherwise} \end{cases}$$

- Hidden layer

sigmoid, tanh, Relu

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \circ \begin{bmatrix} a^2 & b^2 \\ c^2 & d^2 \end{bmatrix} = \begin{bmatrix} a^3 & b^2 \\ c^2 & d^2 \end{bmatrix}$$

## Updates

$$w^{(2)} = w^{(2)} - \eta \cdot \delta^{(2)} \cdot (x^{(1)})^T = \dots$$

$$b^{(2)} = b^{(2)} - \eta \cdot \delta^{(2)} = \dots$$

$$w^{(1)} = w^{(1)} - \eta \cdot \delta^{(1)} \cdot (x^{(0)})^T = \dots$$

$$b^{(1)} = b^{(1)} - \eta \cdot \delta^{(1)} = \dots$$

o  $\text{softmax}$  na output layer :  $P(\text{class} = c) = \frac{e^{z_c^{(\text{out})}}}{\sum_{i=1}^{|C|} e^{z_i^{(\text{out})}}}$

$\xrightarrow{\text{Probabilidade de } x_i \text{ ser da classe } c}$

cross entropy multi-class :  $-\sum_{i=1}^N \sum_{l=1}^{|C|} t_e^{(i)} \cdot \log(x_e^{(\text{out})[c,i]})$

$\xrightarrow{\text{?}, \text{ se } x_i \text{ for da classe } l}$

$\xrightarrow{0, \text{ C.C.}}$

# CLUSTERING

## K-means clustering

1. inicializar os centroides
2. calcular distância de cada observação aos centroides
3. atribuir a cada observação ao cluster com o centroide mais próximo
4. Atualizar os centroides: média dos pontos que pertencem ao cluster do centroide
5. Repetir passos 2-4 até centroides não mudarem

**Silhouette**: avalia a qualidade dos clusters

- coesão: minimizar distâncias intraclusters
- separação: maximizar distâncias interclusters

$$S(n_i) = \begin{cases} 1 - \frac{a}{b}, & \text{se } a < b \\ \frac{b}{a} - 1, & \text{se } a > b \end{cases} \Rightarrow [-1, 1] \quad \begin{matrix} \text{(quanto mais perto)} \\ \text{de 1 melhor)} \end{matrix}$$

→ a: média das distâncias de  $n_i$  aos pontos do seu cluster ( $\downarrow a \uparrow$  coesão)

→ b:  $\min_c$  (média das distâncias de  $n_i$  aos pontos do outro cluster) ( $\uparrow b \uparrow$  separação)

## EM-Clustering

E-step: calcular probabilidade de cada ponto pertencer a cada cluster

M-step: Atualizar os parâmetros das distribuições

E-step:  $P(K=1 | n_i) = \frac{P(n_i | K=1) \cdot P(K=1)}{P(n_i)}$  ← posterior = likelihood  $\times$  prior

M-Step:  $\mu_c = \frac{\sum_{i=1}^N P(K=c | n_i) \cdot n_i}{\sum_{i=1}^N P(K=c | n_i)}$

$\sigma_c = \sqrt{\frac{\sum_{i=1}^N P(K=c | n_i) \cdot (n_i - \mu_c)^2}{\sum_{i=1}^N P(K=c | n_i)}}$  após atualização

$$P(K=c) = \frac{\sum_{i=1}^N P(K=c | n_i)}{N}$$

M-step :  $\sum_C^{(i,j)} = \frac{\sum_{m=1}^N P(K=c | n_m) \cdot ((n_{mi}) - (\bar{m}_{ci})) \cdot (n_{mj} - \bar{m}_j)}$

entradas de matriz concordâncias

valor da feature  $i$  na observação  $n_m$   
após atualização

# DIMENSIONALITY REDUCTION

## Reducción de dimensionalidad

Objetivo: Representarmos os pontos iniciais de  $p$  dimensões em novos pontos de  $K$  dimensões, com  $K < p$

→ Fazem-se transformações de dimensão ou redução de dimensão.

→ Feature selection ≠ reducción de dimensionalidad

→ K-L Transformation: mapear features possivelmente correlacionadas em features não correlacionadas

1. Calcular a matriz de covariâncias  $C$  (do espaço original)
  2. Calcular os valores próprios:  $\det(C - \lambda I) = 0$  **valores próprios**
  3. calcular os vetores próprios:  $C u_i = \lambda_i u_i \Rightarrow (C - \lambda_i I) u_i = 0$
  4. Normalizar os vetores próprios  $u_i := \frac{u_i}{\|u_i\|}$
  5. Transformar os dados:  $x' = \underset{\substack{\uparrow \\ \text{matriz de vetores próprios}}}{U^T} x$  **\*eig**

que mantém o espaço de dimensões

→ PCA (Principal Components Analysis)

PC1 → direcção do espaço original que contém  $\oplus$  informação =  $\oplus$  variância  
 ↳ vetor próprio com maior valor próprio

PC2 → vetor próprio com  $\lambda^2$  maior valor próprio