Toutes les Q sur Bayesien Naif:

2023-2024:

$\mathbb{P}(X=x) = \lambda e^{-\lambda x}$

$\mathbb{E}(X) = 1/\lambda$

$N = 21067$ images ($L \times L = 50$ pixel), $K = 3$ classes

1) Dimension D des données d'entrée: $L^2 = 50^2 = 2500$

Taille de X: (N, D) $(X \in \mathbb{R}_+^{N \times D})$

Taille de Y: N

$(Y \in \{1,2,3\}^n$ ou $\{$ rugbey, volley, foot$\}^N$

2) On s'intéresse au 1er pixel des images de classe K=1

a) modèle associé? param à prendre? $\forall$ pour tous les n tq $y_n = 1$. Chaque $x_{n,1}$ est une valeur de gris (continue, ≥ 0)

⇝ On va modéliser le pixel 1 $(x_{n,1})$ de la classe 1 comme issu d'une loi exponentielle de paramètre $\lambda_{1,1} = \mathbb{P}(X_1 = x_1) = \lambda e^{-\lambda x_1}$ (abus de notation)

$X_1 \mid (Y=1) \sim Exp(\lambda_{1,1})$

b) estimation du maximum de vraisemblance:

[≠ λ contrôle l'échelle: λ grand ⇒ valeurs typiques petites

λ petit ⇒ valeurs typiques grandes

et on sait que pour une exponentielle, $\mathbb{E}[X] = 1/\lambda$]

⇝ on peut deviner que $1/\lambda_{1,1}$ sera la valeur moyenne du pixel 1 pour les images de classe K=1

3) paramètres du modèle global? (θ = ?)

On fait du Naïf Bayes exponentiel:

- on a un modèle par classe $K \in \{1, \dots, K\}$
- dans chaque classe, chaque pixel $j \in \{1, \dots, L^2\}$ est modélisé par une exponentielle avec son propre λ

($\forall$ classe k et $\forall$ j pixel, $X_j \mid (Y=k) \sim Exp(\lambda_{k,j})$

Donc $\theta = \{\lambda_{kj}; k=1,\dots,K, j=1,\dots,L^2\}$ (il y en a KL^2: ici $3 \times 2500 = 7500$ paramètres)

⇝ Si on considère la fréquence d'apparition de chaque classe comme un paramètre à apprendre: on rajoute K=3 paramètres de plus.

4) Maximiser la vraisemblance des données:

[Notations: $X = (X_1, \dots, X_N)$ la v.a. associée à tout le dataset, $(\vec{x_n})_n$ l'ensemble des données effectivement vues, $(y_n)_n$ les labels effectivement vus]

⇝ Cela revient à apprendre les valeurs des paramètres θ telles que $\mathcal{L}(\theta) = \mathbb{P}(X = (\vec{x_n}) \mid y = K, \theta)$ soit maximale.

On cherche $\theta^* = \text{argmax}_\theta(\mathcal{L}(\theta))$

$\theta^* = \text{argmax}_\theta(\log(\mathcal{L}(\theta)))$ [log ↗]

$= \text{argmax}_\theta(\log(\mathbb{P}(X = (\vec{x})_n \mid y^{(n)}, \theta))$ [def vraisemblance: $\mathcal{L}(\theta) = \mathbb{P}(\text{données} \mid \theta)$]

$= \text{argmax}_\theta(\log \prod_{n=1}^{N} \mathbb{P}(X_n = \vec{x_n} \mid y_n = K, \theta))$ [par indépendance des $X_n, n \in [\![1, N]\!]$]

$= \text{argmax}_\theta(\sum_{n=1}^{N} \log(\mathbb{P}(X_n = \vec{x_n} \mid y_n = K, \theta)))$ [$\log(\prod_n a_n) = \sum_n \log a_n$]

or $\forall n, \forall K$ $\mathbb{P}(X_n = \vec{x_n} \mid y_n = K, \theta) = \prod_{d=1}^{D} \mathbb{P}(X_{n,d} = x_{n,d} \mid y_n = K, \theta)$ [hypothèse Naif Bayes: indépendance conditionnelle des pixels sachant $y_n = K$. Abus de notation: $X_{n,d}$ et la v.a. et $x_{n,d}$ la valeur prise dans la réalisation de cette v.a que l'on nomme $x_{n,d}$]

$$= \underbrace{\prod_{d=1}^{D} \lambda_{K,d} e^{-\lambda_{K,d} x_{n,d}} \mathbb{1}_{y_n = K}}_{\text{le modèle de la distribution des features d'entrée est Exponentiel. (les } \theta \text{ sont les } \lambda_{K,d})}$$

donc $\theta^* = \text{argmax}_\theta \left(\sum_{n=1}^{N} \log \left(\prod_{d=1}^{D} \lambda_{K,d} e^{-\lambda_{K,d} x_{n,d}} \right) \mathbb{1}_{y_n = K} \right)$

$= \text{argmax}_\theta \left(\sum_{n=1}^{N} \sum_{d=1}^{D} \log(\lambda_{K,d} e^{-\lambda_{K,d} x_{n,d}}) \mathbb{1}_{y_n = K} \right)$

$= \text{argmax}_\theta \left(\sum_{n=1}^{N} \sum_{d=1}^{D} [\log(\lambda_{K,d}) - \lambda_{K,d} x_{n,d}] \mathbb{1}_{y_n = K} \right)$

⟶ On trouve le maximum de cette fonction [on dérive par rapport à θ, ici $\lambda_{K,d}$ et on cherche la valeur de $\lambda_{K,d}$ telle que la dérivée soit nulle]

$$0 = \frac{\partial \log \mathcal{L}(\theta)}{\partial \lambda_{K,d}} = \frac{\partial}{\partial \lambda_{K,d}} \left(\sum_{n=1}^{N} \sum_{d'=1}^{D} [\log(\lambda_{K,d'}) - \lambda_{K,d'} x_{n,d'}] \mathbb{1}_{y_n = K} \right)$$

↳ pour $d \neq d'$, $\log(\lambda_{K,d'}) - \lambda_{K,d'} x_{n,d'}$ est de 0 % $\lambda_{K,d}$ et le terme dans la somme vaut 0 si $d \neq d'$

$$\Rightarrow 0 = \sum_{n=1}^{N} \left[\frac{1}{\lambda_{K,d}} - x_{n,d} \right] \mathbb{1}_{y_n = K} + \sum_{d' \neq d}^{D} (0)$$

$$\Rightarrow \sum_{n=1}^{N} \frac{1}{\lambda_{K,d}} \mathbb{1}_{y_n = K} = \sum_{n=1}^{N} x_{n,d} \mathbb{1}_{y_n = K}$$

$$\Rightarrow \frac{1}{\lambda_{K,d}} \sum_{n=1}^{N} \mathbb{1}_{y_n = K} = \sum_{n=1}^{N} x_{n,d} \mathbb{1}_{y_n = K}$$

$$\Rightarrow \frac{1}{\lambda_{K,d}} = \frac{\sum_{n=1}^{N} x_{n,d} \mathbb{1}_{y_n}}{\sum_{n=1}^{N} \mathbb{1}_{y_n = K}}$$

⇝ moyenne des pixels numéro d des images de classe K

↑ compte le nombre d'images dans la classe K

5) Comment visualiser ces données?

On a appris des param $\lambda_{K,d}$ (un par classe K et par pixel d). Comme chaque $\lambda_{K,d}$ correspond exactement à un pixel, on peut les remettre en forme L x L. Donc pour chaque classe K, je construis une "image des paramètres" $\Lambda_K(i,j) = \lambda_{K,d(i,j)}$ (mod L x L), je l'affiche en niveaux de gris (ou en heatmap)

Interprétation: $\mathbb{E}[X_{K,d}] = 1/\lambda_{K,d}$: afficher $1/\lambda_{K,d}$ donne une image qui ressemble à la "moyenne" des images de classe K.

6) Ce modèle est aussi un modèle génératif. Comment échantillonner de nouvelles images à partir du modèle?

On choisi un K. ↳ (définir une loi de probas sur les images conditionnellement à une classe K)

On génère $\vec{x} \in \mathbb{R}_+^D$ entiers, pour chaque pixel $d = 1, \dots, D$, indépendamment: $x_d \sim Exp(\lambda_{K,d})$, et on remet $\vec{x}$ en forme L x L.

⇝ Ces images seront facilement identifiables comme fausses car le modèle impose l'indépendance des pixels. Il n'y a donc pas de corrélations spatiales (pixels voisins). On obtient donc du "bruit" et des variations brutales en gris dans des zones censées être uniformes.

7) [Bonus] fonction de décision

Pour un nouveau point de donnée: on connaît $\vec{x}$ et on souhaite prédire y, connaissant les paramètres θ.
On choisi la classe k comme celle qui correspond le plus probablement à l'image observée $\vec{x}$,

$$k^* = \underset{x_k}{\operatorname{argmax}} \left(P(y=k \mid \vec{x}, \theta) \right)$$

$$= \underset{x_k}{\operatorname{argmax}} \left(\frac{P(y=k \cap \vec{x} \mid \theta)}{P(\vec{x} \mid \theta)} \right) \quad \text{Bayes: } P(A|B,C)\,P(B|C) = P(A,B|C)$$

$$= \underset{x_k}{\operatorname{argmax}} \left(\frac{P(\vec{x} \mid y=k, \theta) P(y=k \mid \theta)}{P(\vec{x} \mid \theta)} \right)$$

$$= \underset{x_k}{\operatorname{argmax}} \left(P(\vec{x} \mid y=k, \theta) P(y=k \mid \theta) \right)$$

↑ ne dépend pas de k ; ↑ dépend du modèle choisi ; ↑ probas d'observer une image de classe k, connaissant θ mais pas $\vec{x}$

$$\forall k,\ \pi_k = P(y=k \mid \theta) = \frac{\sum_n^N \mathbb{1}_{y_n=k}}{\sum_n^N 1}$$

↑ fréquence d'apparition de la classe k dans le training set

donc $k^* = \underset{x_k}{\operatorname{argmax}} P(\vec{x} \mid y=k, \theta) P(y=k \mid \theta) = \underset{x_k}{\operatorname{argmax}} \left(\prod_{d=1}^{D} \lambda_{k,d} e^{-\lambda_{kd} x_d} \frac{\sum_n^N \mathbb{1}_{y_n=k}}{N} \right)$

$$= \underset{x_k}{\operatorname{argmax}} \left(\log \left(\prod_{d=1}^{D} (\lambda_{k,d} e^{-\lambda_{k,d} x_d} \right) - \log \sum_n^N \frac{\mathbb{1}_{y_n=k}}{N} \right)$$

$$= \underset{x_k}{\operatorname{argmax}} \left(\sum_{d=1}^{D} [\log(\lambda_{k,d}) - \lambda_{k,d} x_d] + \log \pi_k \right)$$

2021: Bayésien Naïf Gaussien

$P(X_n \in [x, x+dx]) = \rho(x)\,dx$

$$\rho(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)$$

Abus de notation: $P(X_n = x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)$

1) $\theta = \{???\}$

Ici, $X_d \mid (Y_k = d) \sim \mathcal{N}(\mu_{k,d}, \sigma^2_{k,d})$

donc $\theta = \{ \mu_{k,d}, \sigma^2_{k,d} ;\ k = 1, \dots, K,\ d = 1 \dots D \} \cup \{ \pi_k : k = 1, \dots, K \}$

2) $|\theta| = 2KD + 2$

3) Écrire l'algo d'apprentissage (estimation du maximum de vraisemblance) accolades vertes

Mais: $P(X_n = \vec{x_n} \mid y_n = K, \theta) = \prod_d^D \frac{1}{\sqrt{2\pi\sigma^2_{k,d}}} \exp\left(-\frac{(x_{n,d} - \mu_{k,d})^2}{2\sigma^2_{k,d}} \right)$

Les θ étant les $\mu_{k,d}, \sigma_{k,d}$

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \left(\sum_n \log \left(\prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma^2_{k,d}}} \exp\left(-\frac{(x_{nd} - \mu_{kd})^2}{2\sigma^2_{kd}} \right) \right) \mathbb{1}_{y_n=k} \right)$$

$$= \underset{\theta}{\operatorname{argmax}} \left(\sum_n^N \sum_d^D \log \left(\frac{1}{\sqrt{2\pi\sigma^2_{kd}}} \exp\left(\frac{-(x_{nd} - \mu_{kd})^2}{2\sigma^2_{k,d}} \right) \right) \mathbb{1}_{y_n=k} \right)$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_n^N \sum_d^D \left(-\frac{1}{2} \log(2\pi\sigma^2_{k,d}) + \left[\frac{-(x_{nd} - \mu_{kd})^2}{2\sigma^2_{kd}} \right] \right) \mathbb{1}_{y_n=k}$$

↳ $0 = \frac{\partial \log(\mathcal{L}(\theta))}{\partial \mu_{k,d}} = 0 + \left(\sum_n^N \sum_{d'=1}^D \left[\frac{-1}{2\sigma^2_{kd'}} \frac{\partial}{\partial \mu_{k,d}} (x_{nd'} - \mu_{kd'})^2 \right] \mathbb{1}_{y_n=k} \right)$

$$\Rightarrow 0 = \left(\sum_n^N \left[\frac{1}{\sigma^2_{kd}} (x_{nd} - \mu_{kd}) \right] \mathbb{1}_{y_n=k} \right) + \sum_{d \neq d'}^{D} (0) = \sum_n^N [x_{nd} - \mu_{kd}] \mathbb{1}_{y_n=k}$$

$$\Rightarrow \sum_n^N \mu_{kd}\, \mathbb{1}_{y_n=k} = \sum_n^N x_{nd}\, \mathbb{1}_{y_n=k} \Rightarrow \mu_{kd} = \frac{\sum_n^N x_{n,d}\, \mathbb{1}_{y_n=k}}{\sum_n^N \mathbb{1}_{y_n=k}}$$

ce qui correspond à faire la moyenne de la feature n°d des points de classe k

4) Représentation visuelle des paramètres appris par le modèle

$\mu_{k,d}$: pour chaque classe k, remettre $(\mu_{k,d})_{d=1,\dots D}$ en matrice $L \times L$ et l'afficher en niveau de gris (⇝ image "moyenne" de la classe)

$\sigma_{k,d}$ (ou $\sigma^2_{k,d}$) : pour chaque classe k, afficher $(\sigma_{k,d})_d$ en image $L \times L$ (carte d'écart-type/variance par pixel)

π_k: afficher le vecteur $(\pi_1, \dots, \pi_K)$ en diagramme de barres (une barre par classe pour $K > 2$ avec $\sum_k \pi_k = 1$)

5) 🤔 ? échantillonner de nouvelles images

Il suffit de générer des images en tirant au hasard la valeur des pixels (valeur du niveau de gris) selon la loi (gaussienne) de chaque pixel, pour chaque indépendamment des autres pixels.

6) Fonction de décision ?

puis pour la Gaussienne:

$$k^*_{(x)} = \underset{k}{\operatorname{argmax}}\ \pi_k \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma^2_{kd}}} \exp\left(-\frac{(x_d - \mu_{k,d})^2}{2\sigma^2_{k,d}} \right) = \underset{k}{\operatorname{argmax}} \left[\cancel{\log \pi_k} + \sum_{d=1}^{D} \left(\frac{-1}{2} \log(2\pi\sigma^2_{k,d}) - \frac{(x_d - \mu_{k,d})^2}{2\sigma^2_{k,d}} \right) \right]$$

(si on ignore)

2024-2025 : Gaussienne (+ qqs questions)

$x_n \in \mathbb{R}$, $X = (x_1, x_2, \dots, x_N)$

a) Expliciter la vraisemblance / log vraisemblance

↳ probabilité d'avoir obtenu le tirage $X_{(N,1)}$ de puis la v.a. X (ici $D=1$)

$$P(X = X_{(N,1)}) = P(X_1 = x_1, X_2 = x_2, \dots, X_N = x_n)$$

$$\overset{\text{indep}}{=} \prod_n^N P(X_n = n) = \prod_n^N \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_n - \mu)^2}{2\sigma^2} \right)$$

log-vraisemblance:

$$\mathcal{L}(\theta) = N \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) + \sum_n \log\left(\exp\left(-\frac{(x_n - \mu)^2}{2\sigma^2} \right) \right) = -N\frac{1}{2}\log(2\pi\sigma^2) + \sum_n \left(-\frac{(x_n - \mu)^2}{2\sigma^2} \right)$$

NOTE (me): pour $X_{nd} \sim \text{Exp}(\lambda_d)$

$\mathcal{L}(\theta) = P(X = X_{(1,D)}) = P(\forall n \leq N, \forall d \leq D ;\ X_{n,d} = x_{n,d}) \overset{\text{abus de notation}}{=} \prod_{n=1}^{N} \prod_{d=1}^{D} P(X_{n,d} = x_{n,d})$ [indépendance en n et en d]

$P(X_{n,d} = x) = \lambda_d e^{-\lambda_d x}$ d'où $\mathcal{L}(\theta) = \prod_{n=1}^{N} \prod_{d=1}^{D} \lambda_d e^{-\lambda_d x_{n,d}}$ ($= \prod_n \lambda e^{-\lambda x_n}$ en dim 1)

$\log(\mathcal{L}(\theta)) = \sum_n \sum_d (\log \lambda_d - \lambda_d x_{n,d})$

($= N \log \lambda - \lambda \sum_n x_n$ en dim 1)

b) En déduire μ_{MLE}

$$\mu_{MLE} = \underset{\mu}{\operatorname{argmax}}(\mathcal{L}) : \quad 0 = \frac{\partial}{\partial \mu} \mathcal{L} \Rightarrow 0 = 0 + \frac{\partial}{\partial \mu} \sum_n \left(-\frac{(x_n - \mu)^2}{2\sigma^2} \right)$$

$$\Rightarrow 0 = -\sum_n \frac{2(x_n - \mu)}{2\sigma^2} (-1) \Rightarrow 0 = \sum_n (x_n - \mu)$$

$$\Rightarrow \mu = \frac{1}{N} \sum_n x_n$$

avec D dimensions: a) $\theta = \{ \dots \}$

b) $\theta^* = \dots$ (avant)

c) Quelle est la probabilité pour un point $\vec{x}_{test}$ d'appartenir à une certaine classe k?

(⇝ on estime la vraisemblance de la donnée)

Bayes: $P(y=k \mid \vec{x}, \theta) = \frac{P(\vec{x} \mid y=k, \theta)\, \pi_k}{P(\vec{x} \mid \theta)}$ ⇝ $P(y=k \mid \vec{x}, \theta) \propto P(\vec{x} \mid y=k, \theta)\, \pi_k$

$$P(y=k \mid \vec{x}, \theta) = \frac{\pi_k P(\vec{x} \mid y=k, \theta)}{\sum_{j=1}^{K} \pi_j P(\vec{x} \mid y=j, \theta)}$$

↓ version normalisée

Ici: $P(y=k \mid \vec{x}_{test}, \theta) \propto \pi_d \frac{1}{\sqrt{2\pi\sigma^2_{kd}}} \exp\left(-\frac{(x_{test,d} - \mu_{k,d})^2}{2\sigma^2_{kd}} \right)$

d) i) Un modèle Gaussien Naïf n'est pas approprié car les différents attributs (features) sont à priori corrélés entre eux. Le choix du Naïf est problématique. Le choix gaussien, pas spécialement.

ii) Comment contourner l'inconvénient? ⇝ on peut passer au modèle gaussien non-Naïf, c.-à-d. avec une matrice de covariance pleine. Mais on a alors KD paramètres pour les μ et $O(KD^2)$ pour la covariance ($\frac{KD(D+1)}{2}$)

(*) Lien entre Bayésien et K-moyennes

$\vec{\mu_k}$ K moyennes = les $\vec{\mu_k}$ du mélange de Gaussiennes

a_{nk} := proba pour le point n d'être issu du cluster k ⇝⇔ probas ds Q.c)

Bayesien plus général: (1) affectations molles (on donne une probabilité pour chaque cluster). 2) K-means suppose des clusters de même taille/dimension alors qu'avec un mélange gaussien chaque cluster peut avoir son "rayon" σ_k

Bayesien Naïf 2023 (pour le TAL)

1) Nb types, nb tokens
↓ nb mots différents ↓ nb mots

2) On s'intéresse à 2 types (i.e. $V=2$)

a) $x_{n,1}$ = "le mot gratuit apparait" $x_1=(1,0)$ $x_3=(0,0)$ $x_5=(1,1)$

$x_{n,2}$ = "le mot vite apparait" $x_2=(1,1)$ $x_4=(0,1)$ $x_6=(0,1)$

b) $\theta = \{??\}$ $\theta = \{p_{k,v} ; k\in\{1,2\}, v\in\{1,2\}\} \cup \{\pi_k ; k\in\{1,2\}\}$ (↑ 2 classes)

On a donc

$2\times 2 = 4$ paramètres $+2$ si on compte la fréquence des classes

-1 si on utilise le fait que la somme des fréquences vaut 1

au cas général : $K.V + K$ (pour les fréquences)

3) {} (vraisemblance des données)

dans ce cas (Bernoulli) $\mathbb{P}(X_n=\vec{x_n} \mid y_n=k,\theta) = \prod_d^D \mathbb{P}(X_{n,d}=x_{n,d} \mid y_n=k,\theta)$

$\Rightarrow \theta^* = \text{argmax}_\theta \left(\sum_n \log\left(\prod_d p_{k,d}^{x_{n,d}}(1-p_{k,d})^{(1-x_{n,d})}\right)\mathbb{1}_{y_n=k}\right)$

$= \text{argmax}_\theta \left(\sum_n\sum_d \log\left(p_{k,d}^{x_{n,d}}(1-p_{k,a})^{(1-x_{n,d})}\right)\mathbb{1}_{y_n=k}\right)$

$= \text{argmax}_\theta \left(\sum_n\sum_d [x_{n,d}\log(p_{k,d}) + (1-x_{n,d})\log(1-p_{k,d})]\mathbb{1}_{y_n=k}\right)$

Pour trouver le max, $0 = \frac{\partial \log \mathcal{L}(\theta)}{\partial p_{k,d}} = \frac{\partial}{\partial p_{k,d}} \sum_n \sum_{d'=1}^{D} [x_{n,d'}\log(p_{k,d'}) + (1-x_{n,d'})\log(1-p_{k,d'})]\mathbb{1}_{y_n=k}$

$\Rightarrow 0 = \sum_n \left[\frac{x_{n,d}}{p_{k,d}} - \frac{1-x_{n,d}}{1-p_{k,d}}\right]\mathbb{1}_{y_n=k} + \sum_{d'\neq d}^{D}(0)$

$\Rightarrow 0 = \sum_n [x_{n,d}(1-p_{k,d}) - (1-x_{n,d})p_{k,d}]\mathbb{1}_{y_n=k} = \sum_n [x_{n,d} - p_{k,d}]\mathbb{1}_{y_n=k}$

$\Rightarrow \sum_n p_{k,d}\mathbb{1}_{y_n=k} = \sum_n x_{n,d}\mathbb{1}_{y_n=k} \Rightarrow p_{k,d} = \frac{\sum_n x_{n,d}\mathbb{1}_{y_n=k}}{\sum_n \mathbb{1}_{y_n=k}}$

4) Calculer la valeur des 3@

spam: $x_1=(1,0)$, $x_4=(0,1)$, $x_5=(1,1)$ — $p_{1,1}=2/3$, $p_{1,2}=2/3$ ($k=1$ spam)

ham: $x_2=(1,1)$, $x_3=(0,0)$, $x_6=(0,1)$ — $k=2$ = ham, $p_{2,1}=1/3$, $p_{2,2}=2/3$

5) Fonction de décision

{} + $k^* = \text{argmax}_{x_k} \mathbb{P}(\vec{x} \mid y=k,\theta)\,\mathbb{P}(y=k\mid\theta)$

$\mathbb{P}(\vec{x} \mid y=\text{spam},\theta) = \left(p_{1,1}^{x_1}(1-p_{1,1})^{1-x_1}\right)\left(p_{1,2}^{x_2}(1-p_{1,2})^{1-x_2}\right)$

$\rightsquigarrow \mathbb{P}(\vec{x} \mid y=k,\theta)\,\mathbb{P}(y=k\mid\theta) = \left(p_{1,1}^{x_1}(1-p_{1,1})^{1-x_1}\right)\left(p_{1,2}^{x_2}(1-p_{1,2})^{1-x_2}\right)\mathbb{P}(y=k\mid\theta)$

Prédiction issue : $\vec{x}=(1,0)$

On calcule la vraisemblance : $\mathbb{P}(\vec{x} \mid y=\text{spam},\theta) = \frac{2}{3}(1/3)^0 \cdot (2/3)^0 (1/3) = 2/9$. Pour l'autre classe, $\mathbb{P}(\vec{x} \mid y=\text{ham},\theta) = 1/3\,(2/3)^0(2/3)^0(1/3) = 1/9$. La fréquence des classes est 1/2 (équiréparties), donc le terme $\mathbb{P}(y=k\mid\theta)$ ne change pas la probabilité relative des 2 classes.

La proba est donc plus forte pour le spam (⇝ prédiction fausse)

si en réalité seulement 2% des messages sont des spam. on prend en compte en changeant le terme $\mathbb{P}(y=k\mid\theta)$ et dans ce cas il sera beaucoup moins fréquent de prédire un spam (⇝ élargir le vocab)

[questions cours sur TF IDF]

Algorithmes à connaître :

KNN : chercher les K plus proches voisins parmi les N points de données, puis la prédiction correspond à la moyenne du y correspondant, pour ces K voisins.

N calculs de distance $O(D)$, trier la liste $O(N\log N)$

temps d'entrainement $=0$, $O(1)$, stocker les données dans la mem : $O(ND)$, temps d'inférence : $O(N\log(N) + N.D) \approx O(ND)$

Le calcul de la moyenne est négligeable car K est petit.

pb : le choix de K est crucial mais c'est un hyper@, on ne sait pas comment l'ajuster.

K-Moyennes : algo non supervisé qui effectue un clustering

Entrées : $x_i \in \mathbb{R}^D$, K, init, itérations T

principe : alternance : (1) affecter chaque point au centroïde le plus proche, (2) mettre à jour chaque centroïde = moyenne de so cluster

Sorties : centroïdes μ_K + affectations a_i

$O(NKD)$ par itération (⇝ $O(TNKD)$)

$\mu_K = \frac{1}{|C_k|}\sum_{n\in C_k} x_n$ centroïde repr du cluster K

a_{nk} : la var d'affectation du pt x_n au cluster K ($a_{nk}\in\{0,1\}$ $\sum_k a_{nk}=1$) $= 1$ si $n \in$ cluster K, 0 sinon

$a_{nk}=1 \iff k = \text{argmin}_j \|x_n - \mu_j\|^2$

Perceptron :

minimisation $\mathcal{J}(\theta,X,T) = \frac{1}{N}\sum \max(0, -(\vec{\omega}.\vec{x_n})t_n)$ (distance signée du plan P(ω) à $\vec{x_n}$), $t\in\{-1,1\}$

Descente de gradient : $\vec{\theta} \to \vec{\theta} - \eta \vec{\nabla}_\theta \mathcal{J}(\theta,X)$. si $\omega x_n>0$ et $y_n=+1$ max $=0$ (bonne prédiction); si $\omega x_n<0$ et $y_n=-1$ max $=0$ (bonne prédiction)

$\max(0, -f_\theta(\vec{x_n})t_n) = 0$ si bien classé, >0 si mal classé (si $\omega x_n>0$ et $y_n=-1$ ⇒ $-\omega x_n y_n > 0$ ⇒ c'est la distance (absolue) (divisée par N))

$\vec{\nabla}_\theta \mathcal{J}(\theta,X,T) = \frac{-1}{N}\sum_{n\in \text{mal classés}} \vec{\nabla}_\theta(\vec{\omega}.\vec{x_n})t_n$

$\vec{\theta} \to \vec{\theta} + \eta\frac{1}{N}\sum_{n\in\text{mal classés}} \vec{x_n}t_n$

def overfitting : correspond à une performance train très bonne mais performance test/val mauvaise

Cas extrême : mémoriser le train (err train ≈ 0) mais pas de structure ⇝ test mauvais.

mes qt sur overfitting : différence entre le score mesuré (taux d'err, accuracy ou autres) sur l'ensemble d'entrainement et l'ensemble de validation

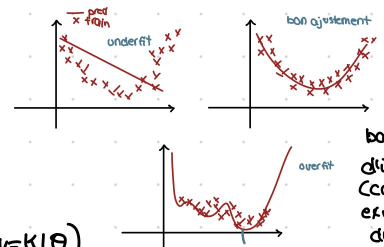
bien généraliser : perf sur données nouvelles ≈ perf sur train

cross validation : intérêt : réduit le bruit statistique lié au tirage des données

Découper en K-folds, entrainer sur K-1 et valider sur 1.

moyenner scores. utile pour choisir hyperparamètres

pq ∇ div ? pas trop grand ⇝ on saute le min



choix param : meilleur score validation

barres verticales : int d'incertitude (cross validation) ex : écart-type (ssu des diff échantillonnages)

η trop petit - descente lente

η bien choisi - convergence stable

η legèrement trop grand - oscillations

η trop grand - divergence

ITERMAX trop petit : arrêt prématuré

overfitting : train ↓ valid ↑

PCA : technique de réduction de dimension, linéaire.

Les composantes principales sont orthogonales les unes aux autres, et chacune capture une part ↘ de la variance totale des données. La var des données issues d'une projection sur une composante = vp associée à la direction propre choisie pour la projection

feature map polyn deg 2 $D=4$

$\phi(\vec{x})$: $1, x_1, x_2, x_3, x_4$ (5)

4 termes $x_1^2, x_2^2, x_3^2, x_4^2$ +4

+6 termes croisés (x_1x_2, x_1x_3, etc)

$\mathcal{L} = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\frac{1}{2}\|\vec{\theta_k} - \vec{x_n}\|\,\mathbb{1}_{y_n=k}$

$L_1 = \frac{1}{N}\sum_{n/y_n=1}\frac{1}{2}\|\vec{\theta_1} - \vec{x_n}\|^2$

$\frac{\partial}{\partial\theta_{1,1}}\|\vec{\theta_1} - \vec{x_n}\|^2 = \frac{\partial}{\partial\theta_{1,1}}\left((\theta_{1,1}-x_{n,1})^2 + (\theta_{1,2}-x_{n,2})^2\right)$

$+ \cdots + (\theta_{1,D} - x_{n,D})^2) = 2(\theta_{1,1}-x_{n,1}) + 0$

de m̂ pour $\frac{\partial}{\partial\theta_{1,2}}\|\vec{\theta_1} - \vec{x_n}\|^2 = 2(\theta_{1,2} - x_{n,2})$

Ainsi on a $\vec{\nabla}_{\vec{\theta_1}} L_1 = \frac{1}{N}\frac{2}{2}\sum_n(\vec{\theta_1} - \vec{x_n}) = \frac{1}{N}\sum_{n/y_n=1}(\vec{\theta_1} - \vec{x_n})$

Pour $\vec{\nabla}_{\theta_k} L$: il faut rajouter $\delta_{k,1}$ ou res, car $\vec{\nabla}_{\vec{\theta_2}} L = \vec{0}$

On a donc $\vec{\nabla}_{\theta_k} L = \frac{1}{N}\sum_{n/y_n=1}(\vec{\theta_1} - \vec{x_n})\,\delta_{k,1}$

b) $\vec{\nabla}_{\theta_k} L_2 = \frac{1}{N}\sum_{n/y_n=2}(\vec{\theta_2} - \vec{x_n})\,\delta_{k,2}$

donc $\vec{\nabla}_{\vec{\theta_k}} = \delta_{k,1}\frac{1}{N}\sum_{n/y_n=1}(\vec{\theta_1} - \vec{x_n}) + \delta_{k,2}\frac{1}{N}\sum_{n/y_n=2}(\vec{\theta_2} - \vec{x_n})$

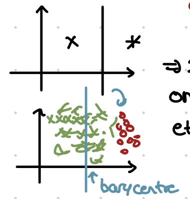
formule descente de grad :

$\theta_k \mapsto \theta_k - \eta\frac{1}{N}\sum_{n/y_n=k}(\vec{\theta_k} - \vec{x_n})$

En posant $\frac{1}{N}\sum_{n/y_n=k}(\vec{\theta_k} - \vec{x_n}) = 0$

$\Rightarrow \sum_{n/y_n=k}\vec{\theta_k} = \sum_{n/y_n=k}\vec{x_n} \Rightarrow N_k\vec{\theta_k} = \sum_{n/y_n=k}\vec{x_n}$

$\vec{\theta_k} = \frac{1}{N_k}\sum_{n/y_n=k}\vec{x_n}$ (barycentre)



eq droite :

$\|\vec{\theta_1} - \vec{x}\|^2 = \|\vec{\theta_2} - \vec{x}\|^2$

$\Rightarrow 2(\vec{\theta_2} - \vec{\theta_1}).\vec{x} + \|\vec{\theta_1}\|^2 - \|\vec{\theta_2}\|^2 = 0$

on identifie $\vec{\omega} = 2(\vec{\theta_2} - \vec{\theta_1})$

et $b = \|\vec{\theta_1}\|^2 - \|\vec{\theta_2}\|^2$

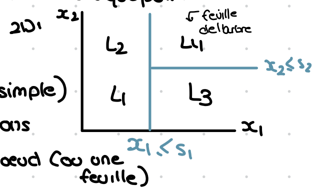
Dernière partie du cours : TF-IDF, K-moyennes, arbres de décision binaire.

I. Notions de base

1) Arbres de décision binaire

Idée générale : On se donne (x_i, y_i) avec $x_i \in \mathbb{R}^D$ (features) et y_i une classe

un arbre de décision binaire fait une suite de questions binaires "feature $x_j \leq t$?" et envoie chaque point dans une feuille.



Chaque feuille contient des exemples (échantillons) d'une ou plusieurs classes.

La prédiction de la feuille peut être : soit la classe majoritaire (vote à la majorité simple)

Soit un vecteur de probabilités de classe. L'espace des features est ainsi découpé en régions rectangulaires alignées sur les axes. On note D l'ensemble des exemples arrivant dans un noeud (ou une feuille)

Mesures d'impureté : Soit p_K la proportion d'exemples de classe K dans D

Taux d'erreur de classification : $I_{err}(D) = 1 - \max_K p_K$

Indice de Gini : $I_{Gini}(D) = \sum_K p_K(1-p_K) = 1 - \sum_K p_K^2$

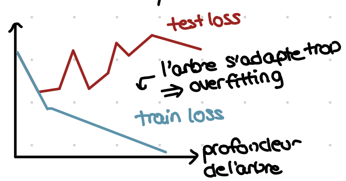
Entropie : $I_{ent} = -\sum_K p_K \log p_K$

Pour une subdivision $D \to (D_{gauche}, D_{droite})$ par ($x_j \leq t$?) on définit $\Delta I(j,t) = I_D - \frac{|D_{gauche}|}{|D|} I(D_{gauche}) - \frac{|D_{droite}|}{D} I(D_{droite})$

On choisi (j,t) qui maximise ΔI (i.e. minimise l'impureté des deux feuilles)

Algo : (choix de la coupure)
- Pour chaque features
  - Pour chaque seuil possible
    - Calculer la pureté de la division possible
    - prendre la meilleur combinaison (meilleure feature avec son meilleur seuil)
- Diviser les dataset selon cette coupure

options d'arrêts : Vote majoritaire (si un noeud contient plusieurs classes, on peut en faire une feuille et prédire la classe majoritaire / descente jusqu'à pureté (on continue à diviser tant que la feuille contient plusieurs classes)



⇝ d'où l'idée de choisir une profondeur intermédiaire (via validation)

II. TF-IDF

Définitions de base :

morphème : la plus petite unité de sens dans une langue. Il peut être :
lexical (racine) → port- dans porteur, portif... / grammatical → ex. -s du pluriel, -ait de l'imparfait...

token : (occurence)
- un token est une occurence d'une unité linguistique dans un texte

type : (forme unique)
- un type est la forme unique du token

vocabulaire V (dictionnaire des types)
- liste des types distincts rencontrés

taille d'un texte = nombre de tokens du texte

richesse de vocabulaire d'un texte : nombre de types apparaissant (taille de l'ensemble V des types présents)

- On va compter 1 lemme (par exemple : verbe aimer) comme ~70 types (↳ 70 formes flexionnelles en français)

2) Problèmes avec différentes langues / où couper les mots

- Exemples :
  - l'allemand est composé de mots extrêmement longs car souvent composé d'autres mots + c'est une langue avec beaucoup de déclinaisons
  - pomme de terre pourra être pris comme trois mots séparés, changeant sa signification s'il était pris en un mot
  - le chinois et le japonais n'ont pas d'alphabets de 'mots' mais plutôt de caractères
  - ponctuation, apostrophes, mots composés (choix de découpage)
  - langues avec morphologie complexe (un seul mot peut contenir plusieurs morphèmes)
    pb : explosion du nombre de types si pas de lemmatisation
    ex : ليكتب ⇝ un seul mot pour dire 'il l'a écrite'
  - langues avec agglutination (coréen, hongrois) : les mots s'enchaînent en blocs ⇝ segmentation indispensable

problèmes des homographes :
2 mots ayant la même écriture mais dont la signification est différente

mots ambigus : mots s'écrivant de la même manière mais dont la signification est totalement différente (ex avocats)

polysémies : " " sign différentes mais se rejoignent (ex : bureau)

idée générale :

idée : comparer les fréquences des types entre les documents

⇝ classer des documents par thème devrait être possible car certains types sont spécifiques à un thème

L'idée générale est de faire en sorte que les termes les plus courants impactent moins le résultat du calcul, aient un poids faible dans le calcul du vecteur représentant un texte.

term Frequency : TF(n,d) = # fois où le type d apparaît dans le doc n (⇝ c'est en fait un comptage)

Document Frequency : DF(d) = # docs où le type de d apparait au moins une fois

Inverse document Frequency : IDF

$= \log\left(\frac{N_{docs}}{DF(d)}\right)$

version régularisée

$IDF(d) = \log\left(\frac{1+N_{docs}}{1+DF(d)}\right)+1$

(ajout d'un doc fictif avec tous les types)

types très courants :
$DF(d) \sim N_{docs}$ donc $IDF(t) \approx 1$

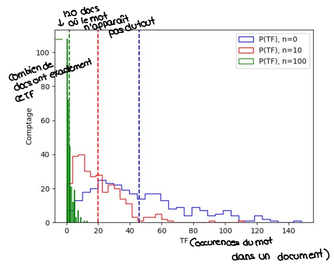
types très rares :
$DF(d) \sim O(1)$ donc $IDF(t) \approx \log(N_{docs})$

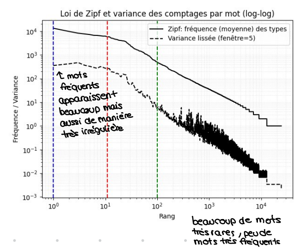
types intermédiaires :
$1 \ll DF(d) \ll N_{docs}$
donc
$IDF(d) \approx \log\left(\frac{N_{docs}}{DF(d)}\right)$

La formule TF.IDF est le produit TF(t).IDF(t)

nb de @ après classificateur linéaire : V pour une classification binaire
KV pour multiclasse si Bayésien avec un seul @ par exemple / one vs rest
one vs one : $V \frac{K(K-1)}{2}$ paramètres.

Matrice de correlation :
[comment lire :]
→ chaque case (i,j) = similitude entre l'exemple i et l'exemple j (valeur entre -1 et 1)
→ couleur claire ⇝ correlation forte → exemples semblables
→ couleurs foncée ⇝ correlation faible ⇝ exemples différents
la diagonale est toujours très claire ($corr(i,i)=1$) → on l'ignore





i.e. la loi de Zipf dit que, dans un grand corpus, la fréquence d'un mot est inversement proportionnelle à son rang : si un mot est le r-ième plus fréquent, $f(r) = C/r$

Quelques mots apparaissent énormément, et la plupart des mots apparaissent très rarement.

2024

1) [un problème est supervisé lorsqu'on a des labels
Y = catégories (= des valeurs qu'on cherche à prédire)]

[exemples de non supervisé:
- on veut regrouper les parties en clusters selon leur style
- faire une PCA pour visualisation
- faire du clustering des types des héros, etc] K-means, PCA, etc

2) [Δ! bien lire les détails du problème]

⇒ L'objectif est d'avoir un modèle qui, à partir des données relatives aux 1 0 premières minutes du jeu, prédit l'équipe gagnante.
~~[le pb serait une régression si $Y \in \mathbb{R}$]~~ régression linéaire, SVM, etc

3) Algorithmes adaptés à la classification:
Perceptron - SVM, SVC, regression logistique, arbres de décision, Bayésien Naïf, KNN...

4) Colonnes à supprimer des données
La colonne gameID ⇒ autant de valeurs possibles que la taille des données
⇒ trop de valeurs uniques, a priori pas d'info sur le jeu
La colonne rwins : [duplication de bwins] [/ combinaison linéaire d'autres colonnes]
La colonne bwins: le label est à supprimer.

5) [Quand mettre en one-hot:
- La variable est catégorielle (pas numérique)
- Peu de valeurs possibles mais sans relation d'ordre

- on retire les colonnes supprimées
- pour chaque one-hot: - retirer 1 colonne originale
- ajouter (nombre de catégories) colonnes.

6) [pour compter la nouvelle dimension D: attention à bien prendre en compte tout l'ensemble des données, et pas uniquement les colonnes montrées]

7) Le jeu de données n'est peut-être pas équilibré [les bleus / rouge gagnent plus?] On peut faire de l'oversampling après le train / validation / test split pour corriger cela.
↳ pour éviter data leakage.

8) [on utilise la symétrisation quand une permutation naturelle des rôles ne change pas le sens du problème..]
On peut copier les N lignes dont on dispose en les symmétrisant, c.à.d. rajouter N lignes au tableau de données, en échangeant les données de l'équipe bleue / rouge pour ces N lignes, et en changeant le vainqueur dans les entrées du tableau Y correspondant. Mathématiquement, $y_{N+i} = 1 - y_i$.
On obtient 2N lignes au lieu de N, pour X et Y. Le jeu de données devient alors exactement équilibré.

9) [P(feature = x_d | victoire) ⇒ si on a gagné, quelles valeurs cette feature prend-elle souvent?]
si intéressant: les distributions ne se recouvrent presque pas ⇒ bonne feature pour prédire la victoire
Le recouvrement entre les deux histogrammes est moindre pour bTtGld, donc cette feature est plus informative
Les 2 histogrammes sont quasiment superposés pour bwardsDstr, donc cette feature est peu utile à priori pour prédire le gagnant

⇒ Il se peut que bTtGld, bien que plus informative seule, soit très corrélée à d'autres features, et donc redondante, et donc inutile, parmi les autres features. Par contre, un classificateur qui n'utiliserait que bTtGld ou bien que BwardsDstr comme feature, aurait de meilleurs résultats avec bTtGld

2023

Id	Age	Sex	PainType	BP	Chol	Fast	ECG	HR	ExAng	Opk	STS	HeartDisease
0	40	M	ATA	140	289	0	Nml	172	No	0.0	Up	0
1	49	F	NAP	160	180	0	Nml	156	No	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	No	0.0	Up	0
3	48	F	ASY	138	214	0	Nml	108	Yes	1.5	Flat	1
4	54	M	NAP	150	195	0	Nml	122	No	0.0	Up	0
5	39	M	NAP	120	339	0	Nml	170	No	0.0	Up	0
6	45	F	ATA	130	237	0	Nml	170	No	0.0	Up	0
7	54	M	ATA	110	208	0	Nml	142	No	0.0	Up	0
8	37	M	ASY	140	207	0	Nml	130	Yes	1.5	Flat	1
9	48	F	ATA	120	284	0	Nml	120	No	0.0	Up	0

918 entrées, 13 colonnes
Age: 28-77
Sex: 'M', 'F'
PainType ['ATA' 'NAP' 'ASY' 'TA']
BP: 0~200
Chol 0~603
Fast 01
ECG ['Normal' 'ST' 'LVH']
HR 60~202
ExAng Yes, No
Opk ~2.6 ~6.2
STS ['Up' 'Flat' 'Down']
HD 01

1) Pb: classification binaire (personne malade ou pas)
2) Algo adapté: SVM, arbre de décision...
3) Suppression données: Id: que des val uniques + ne contient aucune information
4) BP ≠ 0, Chol ≠ 0 : 8 entrées qui présentent pb, sur 918 entrées, on peut simplement les supprimer.
5) encodage / formattage des données: Sex, Fast, ExAng, HD: binaire (True/False) ou one-hot
STS, ECG, PainType: one-hot
6) étapes préprocessing: il faut penser à standardiser, a priori: l'amplitude des valeurs diffère grandement d'une colonne à une autre. (i.e. PCA mais en gardant beaucoup de composantes)
7) répartition des données: le jeu de données n'est p.e. pas équilibré. On peut faire de l'oversampling après le train / validation / test split pour corriger cela.

cours: On note TP = true positives, le nb de cas où on prédit positif et c'est correct
FP = False positives = # prédit positif et c'est incorrect. Resp FN. precision: $\frac{TP}{TP+FP}$ recall = $\frac{TP}{TP+FN}$

Ici, on veut surtout éviter les faux négatifs, donc il faudra privilégier le recall, mais si avec des données non équilibrées, il faut aussi utiliser la précision.

2022

CT/T — 12 val. ↓ possibles — 83~500 — 1~5 — 12 possibilités

	gagnant du round	temps écoulé	carte	bombe posée	Pts vie CT	Pts vie Terro	CTs en vie	Terros en vie	arme CT J1	arme Te J1
1	T	20	dust2	False	500.0	500.0	5	5	famas	mp5sd
2	T	40	dust2	False	391.0	400.0	4	4	famas	mp5sd
3	T	60	dust2	False	391.0	400.0	4	4	famas	mp5sd
4	CT	20	vertigo	False	500.0	500.0	5	5	glock	p90
5	CT	40	vertigo	False	500.0	500.0	5	5	glock	p90
6	CT	60	vertigo	False	295.0	111.0	3	2	mag7	p90
7	CT	80	vertigo	True	295.0	111.0	3	2	mag7	deagl

N = 122410 points de données
Photographies toute les 20s
2000 parties distinctes

1) Pb: classification (binaire)
2) vigilence (20 s): les points de données sont des snapshots temporels d'une même partie ⇒ ils ne sont pas i.i.d. (fortement corrélés quand ils sont proche dans le temps) Donc le train / test split doit se faire par parties (ou par blocs temporels) sans mélanger des points d'une même partie entre train et test / validation (sinon fuite d'info et évaluation trop optimiste). Le nb d'exemples effectivement indépendants est < N (N_{eff}) donc quand on compare N et D il faut raisonner avec N_{eff} plutôt que N.
3) Encodage: Gagnant du round, Bombe Posée: à projeter sur {0,1}
Carte, arme CT J1, arme Te J1: à encoder en one-hot
(eventuellement standardiser PTs vie) (le reste on laisse telles quelles)
4) les joueurs d'une équipe sont interchangeables ⇒ on ne code pas "par joueur" mais par équipe
Pour chaque arme a et chaque équipe $t \in \{CT, T\}$, créer une feature $c_{t,a}$ = #{joueurs de l'équipe t équipés de a} $\in \{0, \_\_\_, 5\}$ donc 12 x 2 = 24 colonnes au lieu de 12 x 10 = 120 en one-hot par joueur
5) vigilence concernant la mesure des performances: le jeu de données n'est p.e pas éq. Il faut penser à utiliser une "balanced accuracy" pour mesurer la qualité du modèle.

2022

id	nom	popu	durée	exp.	artistes	date-sortie	dans.	énerg.	clef	tempo
0B4m..	On The..	39	351960	non	Donna..	1979-10-01	0.735	0.490	4	127.361
277Q..	Medley:..	2	384573	non	Bally..	2005-01-02	0.267	0.832	11	128.321
05ZV..	Please ..	51	234267	non	James..	2008-01-15	0.500	0.819	0	160.125
7cRN..	Masser ..	13	215293	non	Gasol..	1980	0.549	0.356	2	102.282
5f8D..	Noas A..	13	117627	non	Ivar ..	1992-11-23	0.773	0.800	4	149.857
77V0..	Sexy Bi..	37	195853	oui	David..	2009	0.814	0.626	11	130.028
54gA..	Still R..	40	289733	non	Jonny..	1998-01-01	0.649	0.809	10	98.732
1HW..	Intro..	45	319169	oui	Lijpe..	2015-11-09	0.773	0.693	6	93.764

On souhaite prédire la popularité
586672 pistes
nom: 446474 val uniques
popu: 0~100
durée en ms
exp: oui / non
clef: entiers de 0~11
artistes: 315062 v. ≠
dans: 0-1 energ: 01
tempo 0~250

1) apprentissage supervisé, car ici on connait les labels. Régression (car note est un nb réel) (moindres carrés)
2) Supression colonnes? Id, Nom, Artistes car à chaque fois il y a une trop grande proportion de valeurs uniques, donc un encodage one-hot produirait une information non pertinente.
(ou eventuellement du TAL, + TF-IDF pour exclure 'the', 'a' et d'autres stopwords, pe trouver une corrélation
3) Pb: parfois mois + jour, parfois non. Il faut supprimer le mois et jour et ne garder que l'année
4) encodage des données: exp: encodage en binaire 0/1 (one-hot non nécessaire car seulement 2 valeurs possibles)
clef: encodage one-hot à 12 catégories.
5) preprossessing: standardisation car les plages de valeurs couvertes par chaque feature sont très ≠.
On peut enchaîner avec une PCA mais vu la dispertion des valeurs, il faut surtout commencer par standardiser
6) minorité / ...: Oui : la variable cible ("popularité") est probablement très déséquilibrée (beaucoup de valeurs "typiques", peu de valeurs extrêmes) → risque d'un bon score moyen mais mauvais sur les extrêmes.
Remède : pondérer / ré-échantillonner les exemples, par ex. poids (w(y) \propto 1/P(y)) (inverse de la densité de popularité) pour mieux couvrir les rares valeurs.
Évaluation : ne pas se contenter de (MSE) ou (R^2) globaux ; regarder les erreurs par tranches / quantiles de popularité (et/ou scores pondérés), sinon on se trompe sur la perf.

En équilibrant cette répartition (au moins dans la mesure de perf val/test), on obtient une estimation plus fiable du modèle.

en regression

2021

On souhaite prédire la note IMDB d'un film

Titre Film	Année Sortie	Limite age	Durée (min)	Genre	Réal.	Acteur Star	Revenu Généré	Note IMDB
The Sha..	1994	A	142 min	Drama ..	Frank..	Tim Ro..	28,341 k	9.3
The God..	1972	A	175 min	Crime, Dra..	Franc..	Marlon..	134,966 k	9.2
The Dar..	2008	UA	152 min	Action, Cr..	Chris..	Christ..	534,858 k	9.0
The God..	1974	A	202 min	Crime, Dra..	Franc..	Al Pac..	57,300 k	9.0
12 Angr..	1957	U	96 min	Crime, Dra..	Sidne..	Henry ..	4,360 k	9.0
The Ret..	2003	U	201 min	Action, Ad..	Peter..	Elijah..	377,845 k	8.9
Pulp Fi..	1994	A	154 min	Crime, Dra..	Quent..	John T..	107,928 k	8.9

1) app supervisé, regression (note IMDB réparties de façon progressive)
2) Supression: Titre film, Réal, Acteur Star (n'apparaissent en moyenne que dans 12 films, on peut se dire que ce sont quasiment des tokens uniques. Cependant, certains acteurs stars ou réalisateur peuvent être à l'origine de leur succès)
3) encodage des données: Titre x
Année Sortie à mettre en nb entier / flottant, à centrer en 0 ou même standardiser
Revenu Généré, Note IMD ~ passer en float
Genre: several-hot (comme du one-hot parmi tous les sous genres mais il peut y avoir plusieurs bits mis à 1
Limite age: one-hot (en regroupant si possible les cat équivalentes, par tranches d'age)
Durée: enlever le token 'min' et convertir en int / float
4) préprocessing: standardiser les variables continues
5) Algos: SVR, Regression Trees, Regression linéaire (plutôt sans noyau)
6) Minorité: Oui : les notes (IMDB) ne sont pas uniformément réparties, donc un modèle peut "tricher" en prédisant surtout les notes les plus fréquentes. Pour évaluer correctement, il faut regarder les performances par zones de notes (et pas un MSE ou un (R^2) global), et idéalement ré-équilibrer / pondérer la distribution des notes dans la validation/test pour éviter une mesure plus biaisée.

Calcul du gradient :

TD:

optionnel : $J(\theta) = \|\vec\theta\|_2 \to a\vec\theta.\vec x$; $\vec\nabla = a x$

$$\frac{\partial\|\theta\|}{\partial\theta_i} = \frac{\partial(\theta_1^2+\theta_i^2+\dots+\theta_D^2)^{1/2}}{\partial\theta_i} = \frac{1}{2\|\theta\|_2} 2\theta_i = \frac{\theta_i}{\|\theta\|_2}$$

$$\boxed{\Rightarrow \vec\nabla J_0 = \frac{\vec\theta}{\|\vec\theta\|_2} - a.\vec x}$$ ← vecteur en dimension D

1) $J_1(\vec\theta) = \frac{1}{2}\|\vec\theta\|^2 = \frac{1}{2}\vec\theta.\vec\theta = \frac{1}{2}(\theta_1^2 + \_\_\_ + \theta_D^2)$

$\frac{\partial}{\partial\theta_i} J_1 = \theta_i$

$$\Rightarrow \boxed{\vec\nabla_{\vec\theta} J_1 = \vec\theta}$$

2) $J_2(\vec\theta) = \frac{1}{4}\|\vec\theta\|_4^4 = \frac{1}{4}(\theta_1^4 + \_\_\_ + \theta_D^4)$

$\frac{\partial J_2}{\partial\theta_i} = \theta_i^3$

$$\boxed{\Rightarrow \vec\nabla J_2(\vec\theta) = (\theta_1^3, \_\_\_, \theta_D^3)}$$

optionnel : $J_5(\vec\theta) = \|\vec\theta_1\| = |\theta_1| + \_\_\_ + |\theta_D|$

$\frac{\partial J_5}{\partial\theta_i} = \begin{cases} 1 & \text{si } \theta_i > 0 \\ -1 & \text{si } \theta_i < 0 \end{cases} = \text{sign}(\theta_i)$

$\Rightarrow \vec\nabla_{\vec\theta} J_5 = \text{sign}(\vec\theta)$

3) $J_3(\vec\theta) = \frac{1}{2}\frac{1}{N}\sum_n^N (\vec\theta.\vec{x_n} - y_n)^2$ ← fonction de coût pour la regression linéaire

$= \frac{1}{2N}\sum_n^N \left(\sum_j^D \theta_j x_{nj}) - y_n\right)^2$; $\sum_j^D \frac{\partial}{\partial\theta_i}(\theta_j x_{nj}) - y_n = x_i$

$$\boxed{\vec\nabla_{\theta} J_3 = \frac{1}{N}\sum_n^N (\vec\theta.\vec{x_n} - \vec{y_n})\vec{x_n}}$$

$X \in M_{N,D}(\mathbb{R}) = \begin{pmatrix} x_{1,1} & \_\_\_ & x_{1,D} \\ x_{n,1} & & | \\ x_N & \_\_\_ & x_{ND}\end{pmatrix}$; $Y = \begin{pmatrix} y_1 \\ | \\ y_N\end{pmatrix}$

4) $J_4(\vec\theta) = \frac{1}{4}\cdot\frac{1}{N}\sum_n^N (\vec\theta.\vec{x_n} - y_n)^4$

$= \frac{1}{4}\cdot\frac{1}{N}\sum_{n=1}^N \frac{\partial}{\partial\theta_i}\left[\left(\left(\sum_{j=1}^D \theta_j x_{nj}\right) - y_n\right)^2\right]$

$\vec\nabla_{\theta} J_4 = \frac{1}{4N}\sum_n^N 4(\vec\theta.\vec{x_n} - y_n)^3 \vec{x_n}$

$$\boxed{= \frac{1}{N}\sum_n^N (\vec\theta.\vec{x_n} - y_n)^3 \vec{x_n}}$$

$J(\vec\omega, X, Y) = \frac{1}{N}\sum_{n=1}^N \max(0, -(\vec\omega.\vec x^{(n)})y_n)$.

⇝ si sign $(\vec\omega.\vec x^{(n)}) = 1$ et si $y_n = 1$
alors $\max(0, -(\vec\omega.x^{(n)}y_n)) = 0$ (bien classé ⇝ ne s'ajoute pas à la fonction de coût.

⇝ si sign $(\vec\omega.\vec x^{(n)}) = -1$ et si $y_n = -1$
alors $\max(0, -(\vec\omega.x^{(n)}y_n)) = 0$

⇝ si sign $(\vec\omega.\vec x^{(n)}) \neq y_n$
alors $\max(0, -(\vec\omega.x^{(n)}y_n)) = -(\vec\omega.\vec x^{(n)})y_n$

I : l'ensemble des indices des exemples mal classés

$J(\vec\omega, X, Y) = \frac{1}{N}\sum_{n=1}^N \max(0, -(\vec\omega.\vec x^{(n)})y_n) = \frac{-1}{N}\sum_{n\in I}(\vec\omega.\vec x^{(n)})y_n$

⇝ Mise à jour des paramètres par descente de gradient
η taux d'apprentissage
$\vec\omega \leftarrow \vec\omega - \eta\vec\nabla_{\vec\omega} J(\vec\omega, X, Y)$

Ici : $\vec\omega \leftarrow \vec\omega + \eta\frac{1}{N}\sum_{n\in I} y_n \vec x^{(n)}$

$[\, i \in [\![1, D]\!]$, $\frac{\partial J}{\partial\omega_i} = \frac{-1}{N}\sum \frac{\partial}{\partial\omega_i}[(\vec\omega.\vec x^{(n)})y_n]$; $x_i^{(n)} y_n$

$$\boxed{\vec\nabla_{\vec\omega} J = \frac{-1}{N}\sum y\vec x^{(n)}}$$

3) a) Full Batch sur D_1 : $J = \sum_{n=1}^N \max(0, (-\vec\omega.\vec{x_n}) t_n)$
$\omega = (1\ 0\ 0)^t$ $\eta = 1$ ↳ ici, pour simplifier les calculs, on omet le 1/N

$w_0 = [0,0]$ $\vec\omega = [1\ 0\ 0]$
$x^{(n)} = [3,4]$ $\vec{x}^{(n)} = [1\ 3\ 4]$

$$\begin{bmatrix} 1&0&0\\1&0&1\\1&1&0\\1&1&1\\1&4&0\\1&4&1\end{bmatrix} \times \begin{bmatrix}-1\\0\\0\end{bmatrix} \cdot \begin{bmatrix}-1\\-1\\-1\\+1\\+1\\+1\end{bmatrix} = \begin{bmatrix}+1\\+1\\-1\\-1\\-1\\-1\end{bmatrix}$$ $\max(0,i)$

X : $N \times D$ (N=6, D=3)

$\vec{\omega_1} = \vec{\omega_0} - \eta\vec\nabla_{\vec\omega} J$
$\vec{\omega_1} = \vec{\omega_0} + \eta\sum_{n\in I} y_n\vec{x_n}$ | $\begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}-1\\0\\0\end{bmatrix} + \begin{bmatrix}-1\\0\\-1\end{bmatrix} = \begin{bmatrix}-1\\0\\-1\end{bmatrix}$ ($y_1 x^{(1)}$, $y_2 x^{(2)}$)

$\vec\omega^t\vec x + b = \begin{pmatrix}b\\(\vec\omega)\end{pmatrix}\cdot\begin{pmatrix}1\\(\vec x)\end{pmatrix}$

$-\eta\vec\nabla_{\vec\omega} J = +y\sum_{\text{mal classé}} \vec{x_n}.\vec{y_n}$ (⚠! à savoir)

mal classé ⇔ $-(\vec\omega.\vec{x_n})y_n \geq 0$

$-(\vec\omega.\vec{x_1})y_1 = -\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}1\\0\\0\end{pmatrix}\cdot(-1) = +1$

$-(\omega.\vec{x_2})y_2 = -\begin{pmatrix}1\\0\\0\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix}\cdot(-1) = +1$

$X\omega$ ⇝ $-(X.\omega) * Y$

2ᵉ itération :

$$\begin{bmatrix} 1&0&0\\1&0&1\\1&1&0\\1&1&1\\1&4&0\\1&4&1\end{bmatrix} \times \begin{bmatrix}1\\0\\1\end{bmatrix} \cdot \begin{bmatrix}-1\\-1\\+1\\+1\\+1\\+1\end{bmatrix} = \begin{bmatrix}-1\\-2\\1\\2\\1\\2\end{bmatrix}$$ $\max(0,i)$; $-\omega_1$

$\vec{\omega_2} = \vec{\omega_1} + \sum_{n\in I} y_n\vec{x_n}$ | $\begin{bmatrix}-1\\0\\-1\end{bmatrix} + \begin{bmatrix}1\\1\\0\end{bmatrix} + \begin{bmatrix}1\\1\\1\end{bmatrix} + \begin{bmatrix}1\\4\\0\end{bmatrix} + \begin{bmatrix}1\\4\\1\end{bmatrix} = \begin{bmatrix}3\\10\\1\end{bmatrix}$

bien classé

2023

Regression linéaire, descente de gradient : $J = \frac{1}{2}\frac{1}{N}\sum_n^N(\vec\omega.\vec{x_n} + b - y_n)^2$ $\theta = (\vec\omega, b)$

1) $\vec{x_1} = (0,1)$ $\vec{x_2} = (2,4)$ $\vec{x_3} = (3,0)$, $\vec{x_4}(2,1)$, $\vec{x_5} = (1,2)$ $Y = (3, 11, 4, 5, 6)$

Faire une itération

$b^0 = 2$, $\omega^{(0)} = \binom{2}{1}$, $\eta = 0.5$:

$X = \begin{bmatrix}0&1\\2&4\\3&0\\2&1\\1&2\end{bmatrix}$, $Y = \begin{bmatrix}3\\11\\4\\5\\6\end{bmatrix}$ $\hat y = X\omega + b.1$ $b^{(1)} = b^{(0)} - \eta\frac{1}{N}(\vec\omega^{(0)}.\vec{x_n} + b^{(0)} - y_n)$

$$\vec\omega^{(1)} = \vec\omega^{(0)} - \eta\frac{1}{N}\sum_n^N(\vec\omega^{(0)}.\vec{x_n} + b^{(0)} - y_n)\vec{x_n}$$

Calcul des $e_n = (\omega^{(0)}.x_n + b^{(0)}) - y_n$

$\omega.x_1 = 1$ donc $1+2=3 \Rightarrow e_1 = 3-3 = 0$

$\omega.x_2 = 8$ donc $8+2=10 \Rightarrow e_2 = 10-11 = -1$

$\omega.x_3 = 6$ donc $6+2 = 8 \Rightarrow e_3 = 4$ $(e_4 = 2, e_5 = 0)$

$\begin{bmatrix}0\\-1\\4\\2\\0\end{bmatrix}$ $\sum^N e_n = 0 - 1 + 4 + 2 + 0 = 5$

$\sum e_n x_n = 0(0,1) + (-1)(2,4) + 4(3,0) + 2(2,1) + 0(1,2) = (14,-2)$

une itération : $b^{(1)} = b^{(0)} - \eta\frac{1}{N}\sum_n e_n = 2 - 0.5.\frac{1}{5}.5 = 1.5$, $\omega^{(1)} = \omega^{(0)} - \eta\frac{1}{N}\sum_n e_n x_n = \binom{2}{1} - 0.5.\frac{1}{5}\binom{14}{-2} = \binom{0.6}{1.2}$

2) $f(\omega_1, \omega_2, \omega_3) = \omega_1 + 2\omega_2^2 + \omega_2\omega_3^3$ $\nabla f = (1, 4\omega_2 + \omega_3^3, 3\omega_2\omega_3^2)$

3) On part de $J(\omega, b) = \frac{1}{2N}\sum_n(\omega.x_n + b - y_n)^2$

Pour n fixé, on pose $g_n(\omega, b) = \omega.x_n + b - y_n$

Alors $J(\omega, b) = \frac{1}{2N}\sum_n g_n(\omega, b)^2$.

On dérive par rapport à b : $\frac{\partial J}{\partial b}(\omega b) = \frac{1}{2N}\sum_n\frac{\partial}{\partial b}(\omega, b)^2) = 2g_n(\omega, b)\frac{\partial g_n}{\partial b}(\omega, b)$

or $g_n(\omega, b) = \omega.x_n + b$ donc $\frac{\partial g_n}{\partial b} = 1$

Ainsi, $\frac{\partial J}{\partial b}(\omega, b) = \frac{1}{2N}\sum_n 2g_n(\omega, b) = \frac{1}{N}\sum_n(\omega.x_n + b - y_n)$ ⇝ descente de grad sur b $b^{(1)} = b^{(0)} - \eta\frac{\partial J}{\partial b}(\omega^{(0)}, b^{(0)})$
$= b^{(0)} - \eta\frac{1}{N}\sum_{n=1}^N(\omega^{(0)}.x_n + b^{(0)} - y_n)$

On fixe $i \in \{1, \_\_\_, D\}$. $\frac{\partial J}{\partial\omega_i}(\omega, b) = \frac{1}{2N}\sum_{n=1}^N\frac{\partial}{\partial\omega_i}\left(g_n(\omega, b)^2\right)$; or $g_n(\omega, b) = \sum_{j=1}^D \omega_j x_{n,j} + b - y_n$

↳ $= 2g_n(\omega, b)\frac{\partial g_n}{\partial\omega_i}(\omega, b)$

donc $\frac{\partial g_n}{\partial\omega_i}(\omega, b) = x_{n,i}$. Ainsi, $\frac{\partial J}{\partial\omega_i}(\omega, b) = \frac{1}{2N}\sum_{n=1}^N 2g_n(\omega, b)x_{n,i} = \frac{1}{N}\sum_{n=1}^N(\omega.x_n + b - y_n)x_{n,i}$

puis on fait la descente de gradient sur ω $\omega^{(1)} = \omega^{(0)} - \eta\nabla_\omega J(\omega^{(0)}, b^{(0)}) = \dots$ ⇝ $\nabla_\omega J(\omega, b) = \frac{1}{N}\sum_{n=1}^N(\omega.x_n + b - y_n)x_n$

Partie IV : exemples d'erreurs dans le code.

Annale 2024-2025 :

```
clf = sklearn.svm.SVC(C=1, Kernel='poly', degree=4)
clf.fit(X_train_transformed, Ytrain)
trainscore = clf.score(X_train_Transformed, Ytrain)
clf.fit(X_valid_Transformed, Yvalid)
```
✗ il faut pas faire fit deux fois
+ il faut pas fit avec les données de validation

Annale 2022-2023.

```
preProc = sklearn.decomposition.PCA(n_components = 10)
preProc.fit(X_train)
X_transformed = preProc.transform(X_train)
X_valid_Transformed = preProc.transform(X_valid)
clf = sklearn.svm.SVC(C=1, Kernel='poly', degree=3)
clf.fit(X_transformed, Y)
```
il faut entraîner le modèle sur les données transformées
⇒ clf.fit(X_train_Transformed, Y_train)

⇝ l'erreur était que on entraînait sur toutes les données (train + val) donc il n'y avait pas en fait de validation set.
Donc la performance de "validation" était la m̄ qu'en train. Avec un split, comme attendu, la perf en validation est légèrement moins bonne.

NOTES. bien transformer le valid comme le train, only fit train, ne pas fit sur les données valides.

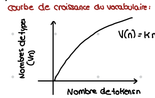
autres exemples :

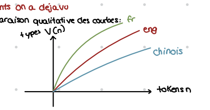
type d'erreur	code erroné	correction	conséquence / explication
1. fit du modèle sur tout le jeu	clf.fit(X, Y)	clf.fit(X_train, Y_train)	Data LeaKage : le modèle "voit" les données de test ⇝ score irréaliste (trop bon)
2. fit du transformeur sur tout X	pca.fit(X)	pca.fit(X_train) puis pca.transform (X_valid) / X_test	fuite d'information : les composantes principales sont influencées par le test ⇝ biais dans la validation
3. Mauvais ordre fit / transform	X_train = pca.transform (X_train) pca.fit(X_train)	pca.fit(X_train) X_train = pca.transform (X_train)	erreur logique : le transform précède le calcul des composantes ⇝ résultats incohérents
4. Double fit du modèle	clf.fit(X_trainval) puis clf.fit(X_trainval)	faire un seul fit	le second fit écrase le 1ᵉʳ ⇝ incohérence, ou apprentissage sur le mauvais split
5. Evaluation biaisée	clf.score(X_trainval, Y_trainval)	clf.score(X_test, Y_test)	test non indépendant ⇝ on mesure la performance sur des données déjà vues
6. Transform après fit global	pca.fit_transform(X) puis split train/test	Split avant, puis fit_transform sur train	Les infos du test influencent la base du modèle ⇝ fuite de données
7. Transform seulement sur le train	pca.fit(X_train) mais X_test inchangé	X_test = pca.transform(X_test)	Le modèle apprend dans un espace transformé. ⚠! les données de validation / test sont dans un espace différent ⇝ prédictions incohérentes

*train dans cross_validate

Exercices sur TF-IDF : (2021)

1) On note à chaque mot, combien de mots différents on a déjà vu
courbe de croissance du vocabulaire : comparaison qualitative des courbes :





3) 1000 pages plantes potagères, les autres traitent des plantes médicales. Étapes d'entraînement du classificateur :
1) prétraitement : nettoyage, tokenization, lemmatisation..
2) vectorisation : construire la représentation TF-IDF de chaque page ⇒ vecteurs x_n
3) split : train / validation / test
4) Apprentissage : entraîner un classifieur (ex Naive Bayes, régression logistique...)
5) Évaluation : accuracy, matrice de confusion, ajuster hyperparamètres si besoin.

2) 2000 page
"et" "concombre" ⇝ 20 fois
page "concombre"
49 x "a" et 49 x "et" Score Tf-Idf ??

on prend la def :
$tfidf(\omega, d) = tf(\omega, d).idf(\omega)$ $idf(\omega) = \frac{\log N}{df(\omega)}$
données : $N = 2000$
$tf(\text{concombre}) = tf(et) = 49$
$df(et) = 2000$ donc $idf(et) = \log\left(\frac{2000}{2000}\right) = 0 \Rightarrow tfidf(et) = 0.49 = 0$
$df(\text{concombre}) = 20$ donc $idf(\text{concombre}) = \log\left(\frac{2000}{20}\right) = \log 100$ ⇝ $tfidf(cc) = 49 \cdot \log(100)$

mini ex : 1) "Le chat noir mange une souris"
2) "Une souris court vite vers le chat"
3) "Le chien obs un chat noir"
19 token, 12 types donc $|V| = 12$
$DF(\text{chat}) = 3$, $DF(\text{souris}) = 2$
$TF(\text{chat}) = 1$, $TF(\text{souris}) = 1$ pour (1)
$IDF(\text{type}) = \log\left(\frac{N_{doc}}{DF(\text{type})}\right)$
donc $IDF(\text{chat}) = \log(3/3) = 0$
$TDIFF(\text{chat}, 1) = 0$
$IDF(\text{souris}) = \log(3/2)$
$IDF.TF(\text{souris}, 1)$
$= 1.\log(3/2) = \log(3/2)$

1) Comment mesurer la richesse de vocabulaire d'un auteur ?
On peut tokeniser un utilisant une expression régulière comme par exemple \w+, ce qui permet de compter le nombre de tokens (chaque mot du texte) et le nombre des types (nombre de mots ≠) de chaque texte. Le ratio type / token est un indice de la richesse du vocabulaire.

2) Comparer la richesse : si on calcule simplement le taux type/token des oeuvres entières, on ne peut rien dire sur la richesse de vocabulaire des auteurs car la taille des oeuvres risque d'être très différente. On pourrait par contre prendre un échantillon randomisé, par exemple 1000 phrases de chacun des auteurs et calculer le taux type/token. Dans ce cas, on pourrait dire que l'auteur avec le plus haut type/token a un vocabulaire plus riche.