Misc

Cross-entropy

Ici on utilise la fonction de loss d'entropie croisée ! En fait, elle vient de ce qu'on a vu en probabilités et statistiques avec la méthode du likelihood :

? d'où vient la cross entropy loss ?

On a donc deux coins, deux distributions :

- le vrai coin **1**, avec comme distribution $\frac{1}{2}$, $\frac{1}{2}$
- notre coin modélisé **2**, qui cherche à se rapprocher de la distribution du coin 1 (parce qu'on ne connaît pas la distribution du coin **1** bien sûr, on cherche à s'en rapprocher à partir de ce qu'on observe). On définit au début nos poids à $0.55, 0.45$ par exemple.

Avec des données d'entraînement, comme l'observation `H H T H T`, on peut calculer :

$$t = \frac{P\left(\text{observation} \mid \text{vrai coin}\right)}{P\left(\text{observation} \mid \text{coin modélisé}\right)} = \frac{P_v}{P_c} = \frac{p_1^{N_H} \cdot p_2^{N_T}}{q_1^{N_H} \cdot q_2^{N_T}}$$

avec p_1, q_1 la probabilité d'avoir un head avec le coin 1 et le coin 2 respectivement et même chose pour p_2, q_2 pour tail, puis N_H, N_T le nombre de head et tail observés respectivement.

On normalise $T = \left(\frac{P_v}{P_c}\right)^{\frac{1}{n}}$ puis on applique le log : $T_{\log} = \frac{1}{n} \log\left(\frac{P_v}{P_c}\right)$, puis les prop du log :

$$T_{\log} = \frac{N_H}{N} \log\left(p_1\right) + \frac{N_T}{N} \log\left(p_2\right) - \frac{N_H}{N} \log\left(q_1\right) - \frac{N_T}{N} \log\left(q_2\right)$$

Là, on peut faire la simplification suivante : avec N qui va à l'infini $\frac{N_H}{N}$, va être très près de la vraie probabilité d'avoir un head, et pareil pour $\frac{N_T}{N}$ donc :

$$\lim_{(N,N_H,N_T)\to\infty} T_{\log} = p_1 \log\left(p_1\right) + p_2 \log\left(p_2\right) - p_1 \log\left(q_1\right) - p_2 \log\left(q_2\right)$$

Et en réarrageant les termes :

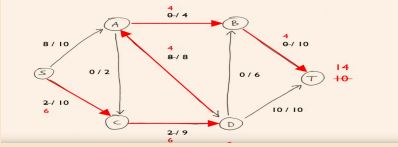
$$\lim_{(N,N_H,N_T)\to\infty} T_{\log} = p_1 \log\left(\frac{p_1}{q_1}\right) + p_2 \log\left(\frac{p_2}{q_2}\right) = D_{KL}\left(P \mid\mid Q\right) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

? Pourquoi D_{KL} est une bonne mesure de la distance ?

C'est une bonne mesure de la distance entre les deux distributions P et Q.

- Si les deux sont exactement égales, le ratio sera de **1** et le log sera égal à zéro.
- Si Q est très différent de P, alors forcément, certains q_i seront plus petits que certains p_i, ce qui fera augmenter très fortement la somme (en effet les termes de q_i un peu plus grand que ceux de p_i, eux, diminuent moins la somme à cause de la forme du $\log$). c'est pour ça d'ailleurs que $D_{KL} \geq 0$!

$$D_{KL}(P \,||\, Q) = \sum_i P(i) \log\left(\frac{Q(i)}{P(i)}\right) \geq \log\left(\sum_i P(i) \cdot \frac{Q(i)}{P(i)}\right)$$
$$= \log \sum_i P(i) = \log 1 = 0$$



tout ce à quoi on vient d'arriver tient aussi avec plus de deux classes !

Nous on veut donc minimiser cette distance D_{KL}.

$$D_{KL}(P_{\text{true}} \mid P_{\text{pred}}) = \sum_y P_{\text{true}}(y \mid x_i) \log\left(\frac{P_{\text{true}}(y \mid x_i)}{P_{\text{pred}(y \mid x_i;\, \theta)}}\right)$$
$$= \sum_y P_{\text{true}}(y \mid x_i) \log\left(P_{\text{true}}(y \mid x_i)\right) - P_{\text{true}}(y \mid x_i) \log\left(P_{\text{pred}}(y \mid x_i;\, \theta)\right)$$

Or ici tout le premier terme est inutile, il ne dépend pas de θ !

On retrouve donc

$$\text{argmin}_\theta D_{KL}(P_{\text{true}} \,||\, P_{\text{pred}}) = \text{argmin}_\theta - \sum_y P_{\text{true}}(y \mid x_i) \log\left(P_{\text{pred}}(y \mid x_i;\, \theta)\right)$$

La formule de l'entropie croisée !

Supervisé vs non supervisé

Supervised : on donne à l'algo une série d'input / output attendus

Fonctions

Sigmoid: $\mathbb{R} \to [0, 1]$

$$\frac{1}{1 + e^{-x}}$$

Softmax c'est la même chose mais ça généralise l'idée quand on veut que K classes, sommées, donnent 1.

Chaque $\text{softmax(z)}_i \in [0,1]$ et $\sum_{i=1}^{k} \text{softmax}(z)_i = 1$.

$$\frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}$$

Différentes normes

L_1 Manhattan distance : $\sum |x_d - x'_d|$.
L_2 Euclidean distance : $\sqrt{\sum (x_d - x'_d)^2}$

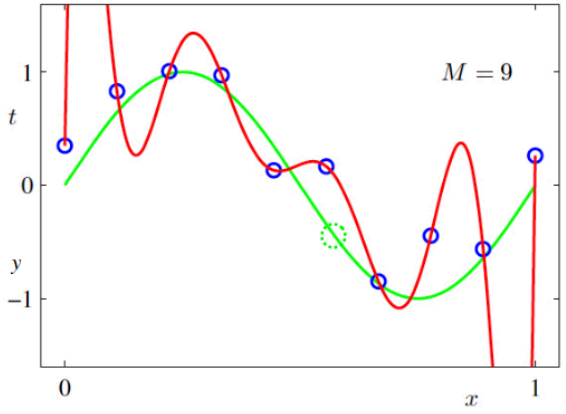
$$L_p \text{ distance } = \left(\sum_{k=1}^{K} |x_i^{(k)} - x_j^{(k)}|^p \right)^{\frac{1}{p}}$$

Entraînements et ensemble de tests

On définit toujours un ensemble de test pour juger les performances du modèle, et un ensemble d'entraînement pour faire apprendre au classificateur.

On veut toujours choisir le bon degré de "conformisation aux données d'entraînement". On ne veut pas que le modèle ne puisse que reconnaisse que les données d'entraînement mais qu'il se rapproche de la courbe réelle.

M=9 -> Overfitting



Cross-validation

1. On divise le dataset en **k sous-ensembles** (appelés *folds*).
2. Pour chaque itération :
   - On prend **k−1 folds pour entraîner** le modèle.
   - On utilise le **fold restant pour tester**.
3. On répète cela **k fois** (chaque fold sert une fois de test).
4. On **moyenne les scores** obtenus (accuracy, F1-score, etc.).

Avantages

- Donne une **meilleure estimation** de la performance du modèle.
- Réduit le **risque de surajustement à un seul jeu de test**.
- Utile quand on a **peu de données**.

Inconvénients

- **Coût computationnel élevé** (surtout pour grands modèles).
- Plus complexe à implémenter que juste train/test split.

Mesurer les performances d'un modèle

Matrice de confusion

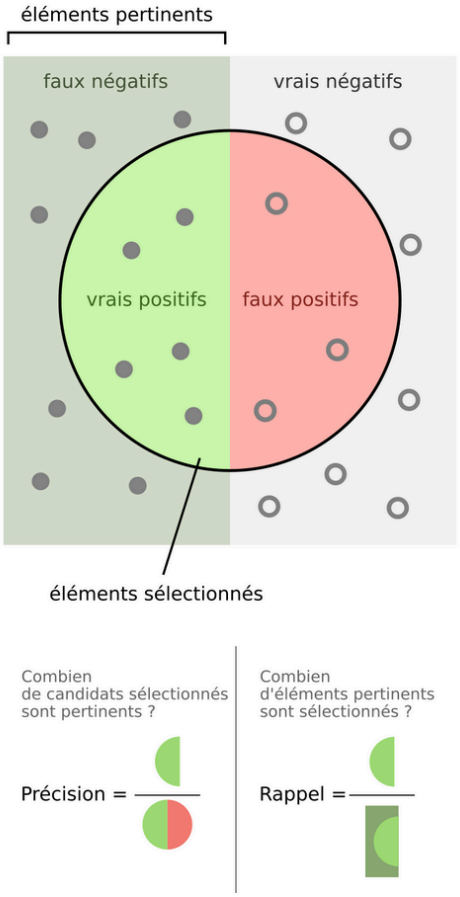
C'est un tableau qui compare les prédictions par rapport aux valeurs réelles.

	Prédit : 0	**Prédit : 1**
Réel : 0	**TN** (Vrais Négatifs)	**FP** (Faux Positifs)
Réel : 1	**FN** (Faux Négatifs)	**TP** (Vrais Positifs)

Précision

Répond à la question : **"Parmi les prédictions positives du modèle, combien sont correctes ?"**

$$\text{Précision} = \quad = \frac{\#\text{ vrais positifs}}{\#\text{ vrais positifs} \ + \ \#\text{ faux négatifs}}$$



Accuracy

Le **recall** (ou **sensibilité**) est une métrique qui mesure la capacité d'un modèle à identifier correctement les **cas positifs**.

$$\text{exactitude} = \frac{\#\text{ vraies prédictions}}{\#\text{ total des prédictions}}$$

Précision	$\frac{TP}{TP + FP}$	"Parmi les prédictions positives, combien sont correctes ?"	Quand les **faux positifs** sont coûteux (ex : filtrage de spam, faux diagnostics médicaux)
Accuracy	$\frac{\#\text{ prédictions correctes}}{\#\text{ total des prédictions}}$	"Globalement, combien de prédictions sont correctes ?"	Si les classes sont **équilibrées** (ex : reconnaissance faciale)
Recall	$\frac{TP}{TP + FN}$	"Parmi tous les vrais positifs, combien ont été détectés ?"	Quand les **faux négatifs** sont critiques (ex : détection de maladies, sécurité)

Le **F1 Score** est une moyenne harmonique entre la précision et le recall, ce qui signifie qu'un modèle avec une précision très élevée mais un recall faible (ou inversement) aura un **F1-score faible**.

$$F_1 = 2 \cdot \frac{\mathrm{Précision} \cdot \mathrm{Recall}}{\mathrm{Précision} + \mathrm{Recall}}$$