IAS - TP 10 - Classifieur Bayesien naïf pour MNIST binarisé

cf Poly de cours: Modèles bayésiens

Contexte du problème

- Données : images de chiffres (0–9) dont les pixels ont été **binarisés** (0 = éteint, 1 = allumé).
- Chaque image est représentée par un vecteur

$$X_i \in \{0,1\}^D, \quad i = 1, \ldots, N,$$

où D est le nombre de pixels.

- Les étiquettes associées sont

$$y_i \in \{0, \ldots, K-1\}, \quad i = 1, \ldots, N.$$

- Objectif : apprendre un **modèle Bayesien naïf** de type Bernoulli par classe pour classer de nouvelles images.

Implem : fonction `BayesienNaif_fit`

```
def BayesienNaif_fit(X_train, y_train):
    ...
```

- `X_train` : matrice de taille (N, D).
  - N : nombre d'images d'apprentissage.
  - D : nombre de pixels par image.
- `y_train` : vecteur de taille $(N,)$ contenant les étiquettes de classe pour chaque image.
- La fonction doit **apprendre les paramètres** du modèle naïf Bayes et les retourner.

Récupération des dimensions

```
N = X_train.shape[0]    # nombre d'images
D = X_train.shape[1]    # nombre de pixels
```

- N : nombre d'exemples d'apprentissage.
- D : dimension des vecteurs d'entrée (nombre de pixels).

Nombre de classes K

```
K = (y_train.max() - y_train.min() + 1)
```

- On suppose que les labels sont des entiers consécutifs :

$$y_i \in \{0, 1, \ldots, K-1\}.$$

- Alors :

$$K = \max(y_{\text{train}}) - \min(y_{\text{train}}) + 1.$$

- Mais : cette méthode est fragile si les classes ne sont pas contiguës (par exemple $\{1, 3, 7\}$).
- Plus robuste :
  - `classes = np.unique(y_train)`
  - `K = len(classes)`
  - éventuellement réindexer les classes en $0, \ldots, K-1$.

Modèle probabiliste naïf Bayes (1)

On modélise :

- $Y \in \{0, \ldots, K-1\}$: classe du chiffre.
- $X = (X_1, \ldots, X_D)$, avec $X_d \in \{0, 1\}$: valeur binaire du pixel d.

Hypothèses du modèle :

1. **Prior de classe / Proba a priori** :

$$P(Y = k) = P_k, \quad k = 0, \ldots, K-1.$$

2. **Modèle Bernoulli par pixel, conditionnellement à la classe** :

$$P(X_d = 1 \mid Y = k) = p_{k,d}, \quad d = 1, \ldots, D.$$

la proba que le pixel x_d soit "allumé" sachant que l'image appartien à la classe k.

Modèle probabiliste naïf Bayes (2)

Pourquoi modèle "naïf" ? car hypothèse d'indépendance conditionnelle entre les différentes dimensions de l'entrée : ici on suppose que les pixels s'allument indépendamment des uns des autres. Mathématiquement :

$$P(x \mid Y = k) = \prod_{d=1}^{D} P(X_d = x_d \mid Y = k).$$

Pour des pixels binaires (modèle de Bernoulli) :

$$P(X_d = x_d \mid Y = k) = p_{k,d}^{x_d}(1 - p_{k,d})^{1-x_d}.$$

Apprentissage

Quel est l'objectif de l'apprentissage ?

Apprentissage

Quel est l'objectif de l'apprentissage ? On veut :

- Estimer les $P_k = P(Y = k)$.
- Estimer les $p_{k,d} = P(X_d = 1 \mid Y = k)$.

Initialisation des paramètres

```
p_kd = np.zeros((K, D))   # P(X_d=1 | Y=k)
P_k  = np.zeros(K)        # P(Y=k)
```

- `p_kd` : matrice de taille (K, D).
  - Ligne k : paramètres pour la classe k.
  - Colonne d : probabilité $p_{k,d}$ que le pixel d soit allumé conditionnellement à $Y = k$.
- `P_k` : vecteur de taille $(K,)$.
  - `P_k[k]` estime $P(Y = k)$ (prior de classe).

Boucle sur les classes : structure générale

```
for k in range(K):
    p_kd[k] = np.mean(X_train[y_train == k], axis=0)
    P_k[k]  = np.mean(y_train == k)
```

- On parcourt toutes les classes $k = 0, \ldots, K-1$.
- Pour chaque k :
  - On sélectionne les lignes de `X_train` correspondant à la classe k via le masque `y_train == k`.
  - On estime :
    - les probabilités conditionnelles par pixel (ligne de `p_kd`),
    - la probabilité a priori de la classe (entrée de `P_k`).

  Rappel : on fait l'hypothèse d'indépendance conditionnelleentre les pixels, et que le fait que le pixel x_d soit allumé ou non pour une image de classe k suit une loi de Bernouilli de paramètre $p_{k,d}$.

Estimation de $p_{k,d} = P(X_d = 1 \mid Y = k)$

Implémentation en une ligne avec numpy :

```
p_kd[k] = np.mean(X_train[y_train == k], axis=0).
```

- `y_train == k` : masque booléen sélectionnant les exemples de classe k.
- `X_train[y_train == k]` : sous-matrice de taille (N_k, D), avec N_k le nombre d'exemples de classe k.
- `np.mean(..., axis=0)` : moyenne colonne par colonne.

Pour un pixel d :

$$\hat{p}_{k,d} = \frac{1}{N_k} \sum_{i:y_i=k} X_{i,d},$$

qui est la **proportion d'images de classe k où le pixel d vaut 1**, donc l'estimateur de maximum de vraisemblance de $P(X_d = 1 \mid Y = k)$.

Estimation des priors $P(Y = k)$

Implémentation en une ligne avec numpy :

$$\texttt{P\_k[k]} = \texttt{np.mean(y\_train == k)}.$$

- `y_train == k` : vecteur de 0/1 de taille N.
- ici on calcule la moyenne suivante :

$$\hat{P}_k = \frac{1}{N} \sum_{i=1}^{N} 1_{\{y_i = k\}}$$

c'est la **proportion d'exemples** de classe k dans la base d'apprentissage.

- C'est l'estimateur de maximum de vraisemblance de $P(Y = k)$.

Sortie de la fonction

```
return p_kd, P_k
```

- La fonction renvoie :
  - `p_kd` : matrice (K, D) des probabilités conditionnelles des pixels
    $$\hat{p}_{k,d} \approx P(X_d = 1 \mid Y = k).$$
  - `P_k` : vecteur $(K,)$ des probabilités a priori de classe
    $$\hat{P}_k \approx P(Y = k).$$
- Ce sont les paramètres du modèle Bayesien naïf Bernoulli appris par maximum de vraisemblance.

Rappel : MLE pour une loi de Bernoulli (1)

On considère des variables aléatoires

$$X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p), \quad X_i \in \{0, 1\}, \quad p \in (0, 1).$$

On observe un échantillon $x_1, \ldots, x_n$ et on note

$$S_n = \sum_{i=1}^{n} x_i.$$

Vraisemblance (souvent notée L pour *likelihood*) :

$$L(p; x_1, \ldots, x_n) = \prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i} = p^{S_n}(1-p)^{n-S_n}.$$

On cherche l'estimateur du maximum de vraisemblance :

$$\hat{p}_{\text{MV}} = \arg \max_{p \in (0,1)} L(p; x_1, \ldots, x_n).$$

Rappel : MLE pour une loi de Bernoulli (2)

On considère la **log-vraisemblance** :

$$\ell(p) = \log L(p; x_1, \ldots, x_n) = S_n \log p + (n - S_n) \log(1 - p).$$

Dérivée pour $p \in (0, 1)$:

$$\ell'(p) = \frac{S_n}{p} - \frac{n - S_n}{1 - p}.$$

Condition de stationnarité $\ell'(p) = 0$:

$$\frac{S_n}{p} = \frac{n - S_n}{1 - p} \implies S_n(1 - p) = (n - S_n)p \implies S_n = np.$$

Rappel : MLE pour une loi de Bernoulli (3)

Donc un extremum de la fonction est

$$\hat{p}_{\mathrm{MV}} = \frac{S_n}{n} = \frac{1}{n} \sum_{i=1}^{n} x_i .$$

On retrouve la moyenne empirique !
Mais : Est-ce un minum ou un maximum ?

Rappel : MLE pour une loi de Bernoulli (3)

Donc un extremum de la fonction est

$$\hat{p}_{\mathrm{MV}} = \frac{S_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

On retrouve la moyenne empirique !
Mais : Est-ce un minum ou un maximum ?

Deuxième dérivée :

$$\ell''(p) = -\frac{S_n}{p^2} - \frac{n - S_n}{(1-p)^2} < 0,$$

donc ℓ est concave et $\hat{p}_{\mathrm{MV}}$ est bien un maximum global.

Retour au pb : Règle de classification naïve Bayes

Pour une image $x \in \{0,1\}^D$ et une classe k,

$$P(x \mid Y = k) = \prod_{d=1}^{D} p_{k,d}^{x_d}(1 - p_{k,d})^{1-x_d}.$$

En logarithme :

$$\log P(x \mid Y = k) = \sum_{d=1}^{D} \big(x_d \log p_{k,d} + (1 - x_d) \log(1 - p_{k,d})\big).$$

Retour au pb : Règle de classification naïve Bayes

Pour une nouvelle image $x = (x_1, \ldots, x_D) \in \{0,1\}^D$, la règle de décision (en log) est :

$$\hat{y}(x) = \arg\max_k \left[\log \hat{P}_k + \sum_{d=1}^{D} \left(x_d \log \hat{p}_{k,d} + (1 - x_d) \log(1 - \hat{p}_{k,d}) \right) \right].$$

Rappel : on peut l'écrire ainsi grâce à l'hypothèse d'indépendance entre les pixels x_d.

Rappel : règle de classification de Bayes (1)

Règle de Bayes (forme générale).
On cherche à prédire la classe

$$\hat{y}(x) = \arg\max_k P(Y = k \mid X = x).$$

Par la formule de Bayes :

$$P(Y = k \mid X = x) = \frac{P(Y = k)\, P(X = x \mid Y = k)}{P(X = x)}.$$

Comme $P(X = x)$ ne dépend pas de k :

$$\hat{y}(x) = \arg\max_k P(Y = k)\, P(X = x \mid Y = k).$$

Rappel : règle de classification de Bayes (2)

Hypothèse d'indépendance conditionnelle (naïve Bayes).
On suppose que les composantes de $X = (X_1, \ldots, X_D)$ sont indépendantes conditionnellement à la classe $Y = k$:

$$P(X = x \mid Y = k) = \prod_{d=1}^{D} P(X_d = x_d \mid Y = k).$$

Pour un modèle Bernoulli avec paramètres estimés $\hat{p}_{k,d} \approx P(X_d = 1 \mid Y = k)$, et en prenant le logarithme, on obtient la règle de décision suivante :

$$\hat{y}(x) = \arg\max_k \left[\log \hat{P}_k + \sum_{d=1}^{D} \left(x_d \log \hat{p}_{k,d} + (1 - x_d) \log(1 - \hat{p}_{k,d}) \right) \right].$$

Implem : `BayesienNaif_predict`

On applique la formule précédente, avec :

- `X` : matrice des nouvelles images à classer, de taille (N, D).
- `p_kd` et `P_k` : paramètres appris par `BayesienNaif_fit`.
- `vecteur_de_probas[k, i]` : score (log-probabilité) de la classe k pour l'image i.

NB : EPS : pour éviter de calculer $log(0)$.

Implem : `BayesienNaif_predict`

Décision finale :

$$\texttt{k\_best} = \arg\max_{k} \texttt{vecteur\_de\_probas[k, i]}$$

pour chaque image i.

- On choisit la classe k qui maximise

$$\log P(Y = k) + \log P(x_i \mid Y = k),$$

  c'est la règle de Bayes en version log.
- `k_best` contient donc les **labels prédits** pour toutes les images de `X`.

Bonus

Autre modélisation : avec une loi Gaussienne
Que dire de la matrice de covariance ?

Modèlisation

Données et variables.

- On représente chaque image par un vecteur

$$X \in \mathbb{R}^D$$

où D est le nombre de pixels (niveaux de gris, réels).

- La classe (chiffre) associée est

$$Y \in \{0, \ldots, K-1\}.$$

Plus besoin de binariser les images !

Modèle bayésien gaussien pour la classification d'images (1)

Modèle génératif par classe.

- Prior de classe :

$$\mathbb{P}(Y = k) = \pi_k, \quad k = 0, \ldots, K-1, \quad \sum_{k=0}^{K-1} \pi_k = 1, \ \pi_k > 0.$$

- Modèle gaussien multivarié pour les pixels conditionnellement à la classe :

$$X \mid (Y = k) \sim \mathcal{N}(\mu_k, \Sigma_k),$$

avec $\mu_k \in \mathbb{R}^D$ et $\Sigma_k \in \mathbb{R}^{D \times D}$.

Modèle bayésien gaussien pour la classification d'images (2)

Densité conditionnelle gaussienne.
Pour chaque classe k, on suppose :

$$p(x \mid Y = k) = \frac{1}{(2\pi)^{D/2}\sqrt{\det \Sigma_k}} \exp\left(-\frac{1}{2}(x - \mu_k)^\top \Sigma_k^{-1}(x - \mu_k) \right).$$

On dispose d'un échantillon d'apprentissage

$$(x_i, y_i)_{i=1}^N, \quad x_i \in \mathbb{R}^D, \ y_i \in \{0, \ldots, K-1\}.$$

Modèle bayésien gaussien pour la classification d'images (3)

Objectif de l'apprentissage : combien de params à estimer ?

Modèle bayésien gaussien pour la classification d'images (3)

Objectif de l'apprentissage : combien de params à estimer ?

- estimer les priors de classe π_k,
- estimer les moyennes μ_k,
- estimer les matrices de covariance pleines Σ_k.

NB : intérêt d'une PCA au préalable

Loi Gaussienne, cas "naïf"

Cas "naïf" (indépendance conditionnelle) :

- On impose souvent une covariance diagonale :

$$\Sigma_k = \mathrm{diag}(\sigma_{k,1}^2, \ldots, \sigma_{k,D}^2).$$

- Les pixels sont alors supposés *indépendants* conditionnellement à la classe :

$$\mathrm{Cov}(X_d, X_{d'} \mid Y = k) = 0 \quad \text{pour } d \neq d'.$$

Signification d'une matrice de covariance pleine

Avec une covariance pleine :

- On autorise des covariances non nulles entre pixels :

$$\mathrm{Cov}(X_d, X_{d'} \mid Y = k) \neq 0.$$

- Le modèle peut capturer des structures plus riches (corrélations entre parties de l'image).
- Le nombre de paramètres par classe est plus élevé :

$$\frac{D(D+1)}{2} \quad \text{paramètres dans } \Sigma_k.$$

On gagne en flexibilité (modèle plus expressif), mais on augmente le risque de sur-apprentissage si le nombre d'exemples par classe N_k n'est pas suffisant.