Université de Toulon — Langage C
Licence 2 Sciences Pour l'Ingénieur — Année 2014-2015

Projet 5 : Bibliothèque C pour la classification automatique d'images binaires

par Faicel CHAMROUKHI

Table des matières

Ce travail devra être effectué en binôme lors des séances de TP et être présenté lors de la dernière heure de la dernière séance de TP.

1 Objectifs

L'objectif de ce projet est de développer une bibliothèque implémentant un algorithme de classification automatique (en anglais *clustering*) d'images.
Le package s'appellera `ClustBinaryImages`.

2 Contexte

2.1 Qu'est ce que la classification automatique ?

En quelques mots, la classification automatique est la tâche qui consiste à regrouper, de façon non supervisée (ç-à-d sans l'aide préalable d'un expert), un ensemble d'objets ou plus largement de données, de telle manière que les objets d'un même groupe (appelé cluster) sont plus proches (au sens d'un critère de (dis)similarité choisi) les unes au autres que celles des autres groupes (clusters). Il s'agit d'une tâche principale dans la fouille exploratoire de données, et une technique d'analyse statistique des données très utilisée dans de nombreux domaines, y compris l'apprentissage automatique, la reconnaissance de formes, le traitement de signal et d'images, la recherche d'information, etc.
L'idée est donc de découvrir des groupes au sein des données, de façon automatique.

2.2 Quelles données ?

Les données traitées en classification automatique peuvent être des images, signaux, textes, autres types de mesures, etc. Dans le cadre de ce projet, les données seront des images binaires. Une image contenant n lignes et m colonnes contient donc $n \times m$ pixels. Chaque pixel est une valeur binaire. Chaque image peut donc être modélisée, par exemple, par une structure contenant au moins les valeurs des pixels et le champ "label". L'ensemble des N images à classer peut donc être modélisé par un tableau de N éléments, chaque éléments du tableau étant une structure "image" binaire.
Pour cette partie `ClustBinaryImages` du projet, on traitera des images binaires.

2.3 Qu'est ce qu'une classe ?

En gros, une classe (ou groupe) est un ensemble de données formée par des données homogènes (qui "se ressemblent" au sens d'un critère de similarité (distance, densité de probabilité, etc)). Par exemple, si on a une base d'image de chiffres manuscrits, on aura la classes des zéros, la classe des uns, etc. De même pour le cas d'images de lettres manuscrites. Une classe peut être aussi une région dans une image couleur, un évènement particulier dans un signal sonore, la classe spam et classes non spam dans le cas détection de spams dans un mail, etc.

2.4 Combien de classes ?

Le nombre de groupes (qu'on notera K), pour commencer, peut être supposé fixe (donné par l'utilisateur). C'est le cas par exemple si l'on s'intéresse à classer des images de chiffres

manuscrits (nombre de classes = 10 : $0, ..., 9$) ou de lettres manuscrites (nombre de classes = nombres de caractères de l'alphabet), etc.

Néanmoins, il existe des critères, dits de *choix de modèle*, qui permettent de choisir le nombre optimal de classes dans un jeu de données. on verra par la suite comment on pourrait calculer le nombre de groupe optimal par ce type de critères.

2.5 Exemple Introductif

Considérons n images binaires, chaque image contient d pixels, chaque pixel peut prendre la valeur 0 ou 1. On représente la $i$ème image ($i = 1, \ldots, n$) par un vecteur $\mathbf{x}_i$ de dimension d : $\mathbf{x}_i = (x_{i1}, ..., x_{ij}, ..., x_{id})^T \in \{0, 1\}^d$.

2.6 Et les stats dans tout ça, pour quoi faire ?

Une image donnée peut être considérée comme étant la réalisation d'une variable aléatoire $\mathbf{x}_i = (x_{i1}, ..., x_{ij}, ..., x_{id})^T \in \{0, 1\}^d$ binaire multidimensionnelle.
On suppose qu'un pixel x_{ij} prend la valeur 1 avec une probabilité p_{ij}. La loi de $\mathbf{x}$ sera alors caractérisée par le vecteur de paramètres $\boldsymbol{p} = (p_{i1}, \ldots, p_{id})$. On remarque que chaque pixel $x_{ij}, j = 1 \ldots, d$ obéit à une loi différente. Chaque pixel (variable binaire) $x_{ij}, j = 1 \ldots, d$ suit en effet une loi de Bernoulli [1] de paramètre p_{ij} avec $p_{ij} = \mathbb{P}(X_{ij} = 1)$.

Supposons que l'on dispose d'un ensemble d'images binaires représentant K chiffres manuscrits (0, 1, 2..) ou K lettres (A, B, C etc) manuscrits. On observe un ensemble d'images non étiquetées et on veut modéliser ces images. On peut modéliser la variable image $\mathbf{x}$ comme un mélange de K lois, chaque loi élémentaire correspond à la modélisation d'un chiffre manuscrits (pour le cas de chiffres) ou d'une lettre (pour le cas des lettres), etc.
Comme chaque pixel binaire suit une loi de Bernoulli univariée, et la loi de l'image (l'ensemble des pixels), pour une classe donnée (loi conditionnelle), suit une loi de Bernoulli multivariée on montre alors que la loi (non-conditionnelle) de l'image est donc un mélange de lois de Bernoulli multivariées.

3 Travail à réaliser

Il s'agit d'implémenter en C l'algorithme Espérance-Maximisation (EM) pour estimer les paramètres d'un modèle de mélange de lois de Bernoulli multivariées afin d'avoir les classes des données. Les paramètres sont $\boldsymbol{\Psi} = (\pi_1, \ldots, \pi_K, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_K)$ est sont estimés par la méthode du maximum de vraisemblance à partir des données d'apprentissage (iid) $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ (e.g., images binaires) afin d'avoir les paramètres du modèle et bien sur les classes des données. Il

1. D'où l'utilité du cours de probas discrètes :) La loi d'une variable binaire $x \in \{0, 1\}$ suivant une loi de Bernoulli de paramètre p est donnée par

$$\mathbb{P}(X = x) = p^x (1 - p)^{1-x}$$

s'agit de maximiser la log-vraisemblance suivante par rapport à $\mathbf{\Psi}$:

$$\mathcal{L}(\mathbf{\Psi}) \quad = \quad \sum_{i=1}^{n} \log \sum_{k=1}^{K} \pi_k \, \mathcal{B}(\mathbf{x}_i; \boldsymbol{p}_k) \tag{1}$$

où

$$\mathcal{B}(\mathbf{x}_i; \boldsymbol{p}_k) = \prod_{j=1}^{d} \mathcal{B}(x_{ij}; p_{kj}) = \prod_{j=1}^{d} p_{kj}^{x_{ij}} \, (1 - p_{kj})^{1 - x_{ij}} \tag{2}$$

est la loi de Bernoulli multivariée de la classe k. Cela s'effectue par l'algorithme EM.

3.1 Algorithme EM pour les modèles de mélanges de loi de Bernoulli multivariées

L'algorithme EM est composée des trois étapes suivantes :

0. Initialisation

On initialise les paramètres $\mathbf{\Psi}$ du modèle (à votre choix) pour donner le pas de départ de l'algorithme : fournir $\mathbf{\Psi}^{(0)}$; Il s'agit donc de donner des valeurs initiales pour les paramètres du modèle :

- la proportion de chaque classe dans l'ensemble des données : π_k (pour $k = 1, \ldots, K$). en respectant le fait que les π_k sont des probabilités (positives et somment à 1 pour $k = 1, \ldots, K$: on a donc $\pi_k > 0 \, \forall k$ et $\sum_{k=1}^{K} \pi_k = 1$)
- les paramètres des lois de Bernoulli pour chaque classe d'images k : les $\boldsymbol{p}_k$ (pour $k = 1, \ldots, K$)

Une fois l'algorithme initialisé, ensuite, répéter les deux étapes E et M suivantes jusqu'à ce qu'une condition de convergence soit vérifiée :

1. Étape E : Espérance

On calcule les probabilités a posteriori des différentes classes pour chacune des données, ç-à-d, pour une donnée $\mathbf{x}_i$, quelle est la chance qu'elle appartienne au cluster k, étant donnés la donnée et une estimation courante des paramètres $\mathbf{\Psi}^{(t)}$, t étant le nombre de l'itération courante . Les probabilités a posteriori sont données par l'équation suivante :

$$\tau_{ik}^{(t)} = \mathbb{P}(z_i = k | \mathbf{x}_i, \mathbf{\Psi}^{(t)}) = \frac{\pi_k^{(t)} \, \mathcal{B}(\mathbf{x}_i; \boldsymbol{\mu}_k^{(t)})}{\sum_{\ell=1}^{K} \pi_\ell^{(t)} \, \mathcal{B}(\mathbf{x}_i; \boldsymbol{\mu}_\ell^{(t)})} \tag{3}$$

2. Étape M : Maximisation

A partir de ces probabilités a posteriori calculées à l'étape-E, on met à jours les paramètre du modèle ; on obtient donc $\mathbf{\Psi}^{(t+1)}$ qui servira pour l'itération d'après. Les équations de mise à jours sont données par :

$$\pi_k^{(t+1)} = \frac{n_k^{(t)}}{n}, \tag{4}$$

$$p_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} \tau_{ik}^{(t)} x_{ij}}{\sum_{i=1}^{n} \tau_{ik}^{(t)}} \tag{5}$$

avec $n_k^{(t)} = \sum_{i=1}^{n} \tau_{ik}^{(t)}$.

3.2 Classification des données d'apprentissage

Une fois l'algorithme a convergé, on a donc une estimation des paramètres $\hat{\mathbf{\Psi}}$ et $\hat{\tau}_{ik}$. Les classes des données d'apprentissage s'obtiennent donc en maximisant les probabilités a posteriori après la convergence $\hat{\tau}_{ik}$, ç-à-d en affectant chaque donnée $\mathbf{x}_i$ à la classe $\hat{z}_i$ ayant la plus grande probabilité a posteriori :

$$\hat{z}_i = \arg\max_{k=1}^{K} \hat{\tau}_{ik} \quad (i = 1, \ldots, n) \tag{6}$$

L'algorithme EM est donné par le pseudo-code 1 ci-après.

Attention ! N'oubliez pas que les indices en C commencent à partir de zéro.

Algorithm 1: Algorithme EM pour les mélanges de loi de Bernoulli multivariées

Input: Données $(\mathbf{x}_1, ..., \mathbf{x}_n)$, nombre de clusters K

Algorithme EM :

Itération : $t \leftarrow 0$

`/* Initialisation : */`

Initialiser les paramètres du modèle : $\mathbf{\Psi}^{(0)} = (\pi_1^{(0)}, ..., \pi_K^{(0)}, \boldsymbol{p}_1^{(0)}, ..., \boldsymbol{p}_K^{(0)})$, avec $\boldsymbol{p}_k^{(0)} = (p_{k1}^{(0)}, \ldots, p_{kd}^{(0)});\ \forall k = 1, \ldots, K$

`/* */`

Nombre d'itérations max : maxIter $\leftarrow 1000$

Fixer un seuil petit pour le test de convergence : $\epsilon > 0$; e.g., $\epsilon = 10^{-6}$

log-vraisemblance : $\mathcal{L}^{(t)} \leftarrow -\infty$

`/* Étapes EM */`

while $((converge \neq 0)$ **and** $(t \leq maxIter))$ **do**

`/* Étape-E : Eséprance */`

for $i \leftarrow 1$ ***to*** n **do**

for $k \leftarrow 1$ ***to*** K **do**

Calculer $\tau_{ik}^{(t)}$ en utilisant l'équation (3)

end

end

`/* Étape-M : Maximisation (Mise à jour) */`

for $k \leftarrow 1$ ***to*** K **do**

Calculer $\pi_k^{(t+1)}$ en utilisant l'équation (4)

Calculer $\boldsymbol{p}_k^{(t+1)}$:

for $j \leftarrow 1$ ***to*** d **do**

Calculer $p_{kj}^{(t+1)}$ en utilisant l'équation (5)

end

end

$t \leftarrow t + 1$ (itération suivante)

`/*Test de convergence*/`

Calculer la log-vraisemblance $\mathcal{L}\left(\mathbf{\Psi}^{(q+1)}\right)$ en utilisant l'équation (1)

if $\left(\left(\frac{|\mathcal{L}^{(t+1)} - \mathcal{L}^{(t)}|}{|\mathcal{L}^{(t)}|} \leq \epsilon\right) || \, (t \geq maxIter)\right)$ **then**

$converge = 1$

end

end

Result: $\hat{\mathbf{\Psi}} \leftarrow \mathbf{\Psi}^{(t)}$: paramètres du modèle ; $\{\hat{\tau}_{ik}\} \leftarrow \{\tau_{ik}^{(t)}\}$: probabilités a posteriori

Voici encore plus en détails ce qu'il faudra faire. Afin d'implémenter l'algorithme 1 :

Fonctions à implémenter :

Créer une fonction `learn_mvBernMix_EM` (ça sera la fonction principale) qui estime les paramètres $\{\pi_k, \boldsymbol{p}_k\}$ du modèle mélange de lois de Bernoulli multivariées, et fournit également les probabilités a posteriori τ_{ik}.

Pour cela, cette fonction comprendra les trois étapes : initialisation, E et M :

• Créer une fonction `init_mvBernMix` pour initialiser les paramètres $\{\pi_k, \boldsymbol{p}_k\}$ du modèle.

• Créer une fonction `Bern` qui calcule la loi de Bernoulli multivariée donnée par l'équation (2). La fonction `mvBern` sera appelée ensuite pour le calcul des probabilités a posteriori τ_{ik} de l'itération courante comme donnée par l'équation (3) dans l'étape E. Implémenter donc le calcul des probas a posteriori.

• Implémenter la mise à jour des paramètres $\{\pi_k, \boldsymbol{p}_i\}$ du modèle dans l'étape M à partir des équations (4) et (5).

• Créer une fonction `loglik_mvBernMix` qui calcule la log-vraisemblance d'un mélange de loi de Bernoulli multivariée pour un ensemble de données. Cela s'effectue en implémentant l'équation (1). La valeur calculée par cette fonction servira du test de convergence et devra aussi être affichée à chaque itération de l'algorithme sur l'unité de sortie standard.

Une fois l'algorithme a convergé, calculer les classes par la règle donnée par l'équation (6). Tester préalablement votre algorithme sur des données simulées (vous seront fournies) avant de le tester sur des images.

3.3 Classification de données de test

On appelle données de test, des données qui n'ont pas servi pour entraîner le modèle, par opposition aux données d'apprentissage qui ellent ont servi pour apprendre le modèle. Donc une, fois le modèle a été appris à partir des données d'apprentissage, on peut également s'en servir pour estimer les classes de nouvelles données de test (que l'on suppose ayant la même densité que celles qui ont servi pour l'apprentissage) en se basant sur le modèle estimé préalablement en appliquant l'algorithme EM. La procédure de test consiste à calculer les probabilités a posteriori τ_{ik} comme dans l'étape de l'algorithme EM au moment de l'apprentissage, mais la différence cette fois-ci c'est que les paramètres existent déjà (pas la peine de les estimer), et ensuite affecter chaque exemple à la classe correspondant à la probabilité maximum (Équation 3). Cela peut se résumer par le pseudo code suivant :

Algorithm 2: Application d'un modèle de mélange de loi de Bernoulli multivariées pour la classification de données de test

Input: Données de test $(\mathbf{x}_1, ..., \mathbf{x}_n)$, paramètres du modèle
$\boldsymbol{\Psi} = (\pi_1, \ldots, \pi_K, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_K)$

`/* Procédure de test */`

`/* Calculer les probas a posteriori */`

for $i \leftarrow 1$ ***to*** n **do**
  for $k \leftarrow 1$ ***to*** K **do**
    Calculer τ_{ik} en utilisant l'équation (3)
  end
end

`/* classer les donneés par la règle du MAP */`

for $i \leftarrow 1$ ***to*** n **do**
  Calculer $\hat{z}_t$ en utilisant la règle du MAP (5) :
  $\hat{z}_i = \arg\max_{k=1}^{K} \tau_{ik}$
end

Result: $\hat{z}_i \quad (i = 1, \ldots, n)$: classes estimées

• Créer une fonction `test_mvBernMix` qui calcule les classes $\hat{z}_i$ $(i = 1, \ldots, n))$ des données de test en maximisant les probabilités a posteriori τ_{ik}.
Vous pouvez préalablement créer aussi une fonction `PpostProb_mvBernMix` qui calcule les probas a posteriori qui pourra servir aussi bien pour cette étape de test que pour l'étape d'apprentissage par l'algorithme EM.

4 Présentation des résultats

La présentation des résultats pourra se faire, en premier lieu en les affichant à l'écran ou en les écrivant dans un autre fichier.
Ensuite, pour aller plus loin, la présentation des résultats se fera par des présentations graphiques en affichant par exemple les données dans l'espace, chaque ensemble de données appartenant à une même classe est colorée par une couleur différente, etc. Pour les images, essayer de regarder si les classes trouvées correspondent bien aux vraies catégories des lettres ou chiffres manuscrits . Essayer de dresser une matrice de confusion, etc

Bonne chance !