APR (E.T.S. de Ingeniería Informática)
Curso 2022-2023

Práctica 1. Mixturas de gaussianas

Jorge Civera, Alfons Juan, Francisco Casacuberta, Enrique Vidal
Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València
Última actualización: 11/11/2022- 11:00:32

1. Trabajo previo a la sesión de prácticas

Para la realización de este proyecto de prácticas se supone que previamente has adquirido experiencia en el uso de `python` y `gnuplot`, tanto en la asignatura de Sistemas Inteligentes como en la asignatura de Percepción. Se recomienda leer el boletín en su totalidad, ya que esto permite centrarse en el trabajo a desarrollar en la sesión de laboratorio y aprovechar mejor el tiempo.

2. Mixtura de gaussianas

En esta primera práctica estudiaremos una generalización del clasificador gaussiano visto en la asignatura de Percepción, el clasificador de mixtura de Gaussianas. Las mixturas, ya sean de Gaussianas o de cualquier otra distribución, nos permiten introducir una instancia del algoritmo EM estudiado en teoría.

La estimación de un modelo de mixturas de gaussianas está implementado en la librería `scikit-learn` bajo el paquete `sklearn.mixture` en su clase `GaussianMixture`[1]. Para poder emplear este paquete simplemente debes importarlo:

```
from sklearn import mixture
```

La clase `GaussianMixture` implementa la creación de un modelo de mixturas de gaussianas mediante el constructor del mismo nombre. Por ejemplo, un modelo de K componentes se puede crear mediante

```
gmm = mixture.GaussianMixture(n_components=K)
```

El proceso de aprendizaje a partir de un conjunto de muestras `X` se realiza mediante la llamada a la función `fit`

```
gmm.fit(X)
```

[1] https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html

y la estimación del logaritmo de la verosimilitud de cada una de estas muestras se puede calcular mediante la función `score_samples`:

```
gmm.score_samples(X)
```

Estas funciones se pueden combinar para implementar un clasificador cuya probabilidad condicional está modelizada mediante una mixturas de Gaussianas (ver fichero `mixgaussian-exp.py`)

```
 1 import sys
 2 import math
 3 import numpy as np
 4 import pickle
 5 from sklearn import mixture
 6
 7 if len(sys.argv)!=5:
 8   print('Usage: %s <trdata> <trlabels> <%%trper> <%%dvper>' % sys.argv[0]);
 9   sys.exit(1);
10
11 X= np.load(sys.argv[1])['X'];
12 xl=np.load(sys.argv[2])['xl'];
13 trper=int(sys.argv[3]);
14 dvper=int(sys.argv[4]);
15
16 K=1;
17 rc=0.1;
18 seed=23;
19
20 N=X.shape[0];
21 np.random.seed(seed); perm=np.random.permutation(N);
22 X=X[perm]; xl=xl[perm];
23
24 # Selecting a subset for train and dev sets
25 Ntr=round(trper/100*N);
26 Xtr=X[:Ntr,:]; xltr=xl[:Ntr];
27 Ndv=round(dvper/100*N);
28 Xdv=X[N-Ndv:,:]; xldv=xl[N-Ndv:];
29
30 labs=np.unique(xltr).astype(int);
31 C=labs.shape[0];
32 N,D=Xtr.shape;
33 M=Xdv.shape[0];
34 gtr=np.zeros((C,N));
35 gdv=np.zeros((C,M));
36
```

```
37 # Normalise data
38 mu=np.mean(Xtr,axis=0);
39 sigma=np.std(Xtr,axis=0);
40 sigma[sigma==0]=1;
41 Xtr=(Xtr-mu)/sigma;
42 Xdv=(Xdv-mu)/sigma;
43
44 # Parameter estimation and soring samples
45 model=[]
46 for c,lab in enumerate(labs):
47   Xtrc=Xtr[xltr==lab];
48   Nc=Xtrc.shape[0];
49   pc=Nc/N;
50   gmm=mixture.GaussianMixture(n_components=K, reg_covar=rc, random_state=seed);
51   gmm.fit(Xtrc);
52   gtr[c]=math.log(pc)+gmm.score_samples(Xtr);
53   gdv[c]=math.log(pc)+gmm.score_samples(Xdv);
54   model.append((pc,gmm));
55
56 # Classification of training and eval sets for error estimation
57 idx=np.argmax(gtr,axis=0);
58 etr=np.mean(np.not_equal(labs[idx],xltr))*100;
59 idx=np.argmax(gdv,axis=0);
60 edv=np.mean(np.not_equal(labs[idx],xldv))*100;
61
62 print('  K      rc   etr   edv')
63 print('--- ------- ----- -----')
64 print(f'{K:3} {rc:3.1e} {etr:5.2f} {edv:5.2f}');
65
66 filename = 'gmm.K1.rc0.1.mod'
67 pickle.dump(model, open(filename, 'wb'))
```

Las líneas 1-5 importan las librerías `python` necesarias. A continuación, las líneas 7-14 realizan la lectura de parámetros de la línea de comandos: conjunto de datos (imágenes y etiquetas de clase) y porcentaje de datos dedicados a entrenamiento y validación. Las líneas 16-18 definen una serie de parámetros predefinidos para crear un modelo de mixturas de Gaussianas, siendo `K` el número de componentes de la mixtura, `rc`, el factor de regularización de la matriz de covarianzas y `seed`, la semilla aleatoria. Las líneas 20-28 realizan un barajado aleatorio de los datos y una partición en entrenamiento y validación.

Seguidamente, las líneas 30-35 obtienen el vector de etiquetas de clase sin repeticiones `labs`, el numero de clases `C`, el número de muestras de entrenamiento `N`, el número de dimensiones de los datos `D`, el número de muestras de validación `M`, y el valor de las funciones discriminantes para todas las clases y todas las muestras tanto en el conjunto

de entrenamiento (`gtr`), como en el conjunto de validación (`gdv`).

Después las líneas 38-42 normalizan los datos restando la media y dividiendo por la desviación típica, evitando dividir por posibles desviaciones típicas que sean cero.

En las líneas 45-54 se crea, entrena y evalúa un modelo de mixturas por cada clase. Primero, se seleccionan las muestras de la clase correspondiente (línea 47), se calcula la prior `pc`, y se inicializa el modelo de mixturas de `K` componentes en la línea 50. A continuación, se entrena el modelo con los datos de la clase (línea 51). Finalmente, se calcula el valor de la función discriminante de la clase sumando la probabilidad a priori de la clase al logaritmo de la verosimilitud de cada una de las muestras, tanto para el conjunto de entrenamiento como para el conjunto de validación (líneas 52-53). En la línea 54 se añade el modelo de la clase a un vector, que más tarde se guardará a fichero.

Por último, las líneas 57-60 realizan la clasificación de las muestras de entrenamiento y validación para calcular el error de clasificación. Estas tasas de error se imprimen por pantalla en las líneas 62-64. Como se ha mencionado anteriormente, el modelo almacenado en el vector `model` se guarda a fichero en formato `pickle` en las líneas 66-67.

2.1. Tarea MNIST

La tarea de esta práctica consiste en aplicar un clasificador basado en mixtura de gaussianas a la tarea MNIST utilizando una versión de los datos de MNIST proyectados mediante PCA a 20 dimensiones disponible en PoliformaT .

Ejercicio. A la vista de los parámetros del constructor de la clase `GaussianMixture`, entrena y ajusta los parámetros de un clasificador basado en mixturas de gaussianas que minimice el error de clasificación, teniendo en cuenta que el tamaño del clasificador guardado en fichero en formato `pickle` sin comprimir no debe superar 1 Mbyte.

Una vez determinados los parámetros óptimos del clasificador bajo la restricción de tamaño de fichero arriba mencionada, entrena con estos parámetros un clasificador final utilizando todos los datos que tienes disponibles y guárdalo en un fichero `pickle`. Puedes comprobar que el clasificador final guardado se carga correctamente desde el fichero `pickle` y clasifica un conjunto de muestras mediante el fichero `mixgaussian-eva.py`.

Deberás entregar una memoria que describa los experimentos realizados para determinar los parámetros óptimos del clasificador final y tu clasificador final en formato `pickle`. Este clasificador será evaluado por parte del profesorado en un conjunto de test independiente.