Introduction à l'Apprentissage Automatique TP3

Classification par Lois Normales par approche Bayésienne

ROUSSEL Hugo, M1 IARF TPA12

10/02/2019

Code :

Prédiction :

Calcul du “score de probabilité” de l’observation x sachant la classe i :

```
score = lambda x, i: 0.5*(n_features*math.log(2*math.pi) + \
  math.log(np.linalg.det(self.sigma[i])) + \
  np.dot((x-self.mu[i]).T, np.dot(np.linalg.inv(self.sigma[i]),self.mu[i])))
```

Plus le score est élevé, mieux c’est, pour prédire à quelle classe appartient x on fait un argmax sur toutes les classes :

```
np.argmax([(score(X[i],c) + (0 if self.priors is None else
math.log(self.priors[c])) ) for c in range(n_classes)])
```

Si des probabilités à priories sont fournies, on en tient compte.

Apprentissage :

Pour l’apprentissage il suffit :

De mémoriser la moyenne de chaque classe c :

```
np.mean([X[i] for i in range(len(X)) if y[i] == classes[c]], axis=0)
```

Et de calculer la matrice de covariance de chaque classe c :

```
np.cov([X[i] for i in range(len(X)) if y[i] == classes[c]], rowvar=False)
```

Optimisation :

Si l’on fait l’hypothèse que les composantes sont indépendantes entre elles, on peut considérer la matrice de covariance de classe comme diagonale :

```
if self.diag :
   cov = np.cov([X[i] for i in range(len(X)) if y[i] == classes[c]],
rowvar=False)
   self.sigma[c] = cov*np.identity(n_features)
```

Tests :

Résultats sur le dataset de test dans différentes situations :

	Précision
Sans probabilité à priorie ni optimisation	(base) D:\hugor\Documents\M1\iaa\TP3_etudiant>python main.py precision : 1.00 Press any key to exit...
Avec probabilité à priorie Sans optimisation	(base) D:\hugor\Documents\M1\iaa\TP3_etudiant>python main.py Priors : [0.33333333 0.33333333 0.33333333] precision : 1.00 Press any key to exit...
Avec probabilité à priorie Avec optimisation	(base) D:\hugor\Documents\M1\iaa\TP3_etudiant>python main.py Priors : [0.33333333 0.33333333 0.33333333] precision : 1.00 Press any key to exit...

Pour montrer que le nombre de classe et de dimension peut être quelconque, la solution est testé avec un dataset simpliste de 2 classes 3D

Train	Test
1.0 , 1.0, 1.0, 0 2.0 , 2.0, 2.0, 0 1.0 , 1.0, 1.0, 0 -1.0 , -1.0, -1.0, 1 -2.0 , -2.0, -2.0, 1	1.0 , 1.0, 1.0, 0 -2.0 , -2.0, -2.0, 1

Résultat d'exécution avec le dataset simpliste	(base) D:\hugor\Documents\M1\iaa\TP3_etudiant>python main.py Priors : [0.6 0.4] precision : 1.00 Press any key to exit...