Przetwarzanie i analiza obrazów 07.03.2019

Lista nr 2

Paweł Lorek

PCA + kNN

Jednym z najprostszych klasyfikatorów jest kNN (k nearest neighbours): Załóżmy, że mamy dane punkty $x_1, \ldots, x_n$ (powiedzmy d wymiarowe), i przypisanie klas: $y_i = f(x_i) \in C = \{C_1, \ldots, C_M\}$. Wówczas kNN z $k = 1$ działa następująco ($dist$ do odległość, np. euklidesowa):

$$class(x) = f\left(\arg\min_{i \in \{1,\ldots,n\}} dist(x, (x_i)\right)$$

Jeśli X_train jest macierzą $n \times d$ (n punktów wymiaru d) a Y_train kolumną rozmiaru d o wartościach całkowitych ($Y\_train[i]$ = numer klasy punktu $X\_train[i, :]$) to w `Pythonie` tak możemy używać kNN :

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

gdzie X_test to "nowe" punkty d-wymiarowe, a y_pred to wynik klasyfikacji.

Mając punkty w macierzy X i klasy w wektorze (kolumnie) Y często dane te dzielimy na część treningową (na tych danych uczymy klasyfikatora) oraz testową (na tych zaś testujemy). Prosto można to zrobić w taki sposób (20% danych będzie losowo wybranych jako testowe, a reszta jako treningowe):

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

1. Potestuj kNN na danych `Iris` :

   ```
   from sklearn import datasets
   iris = datasets.load_iris()
   X = iris.data
   Y = iris.target
   ```

   (wylicz wynik klasyfikacji: (liczba dobrze sklasyfikowanych)/wszystkie)

2. Zrób to samo na obrazkach MNIST:

   ```
   mnist = datasets.load_digits()
   ```

- Wykonaj kNN na oryginalnych danych (64 wymiarowe)
- Wykonaj kNN na danych zredukowanych za pomocą PCA (sprawdź jaki wymiar daje najlepszy wynik klasyfikacji)

3. "Labeled Faces in the Wild" (w skrócie LFW) to zestaw 1288 obrazków twarzy znanych osób. Dane można w Pythonie wczytać w taki sposób

```
from sklearn.datasets import fetch_lfw_people
lfw_people = fetch_lfw_people('data', min_faces_per_person=70,
resize=0.4)
X=lfw_people.data
Y=lfw_people.target
n_samples,h,w = lfw_people.images.shape
```

   Ostatnia instrukcja odczytuje kolejno: n_samples = liczba obrazków, h = wysokość, w = szerokość każdego obrazka (w X każdy obrazek do jeden wiersz i $h \cdot w$ kolumn)

4. Sprawdź jaki wpływ na klasyfikację ma *normalizacja* danych (przed/po PCA). Możesz to zrobić "ręcznie" albo użyć:

```
>>> import sklearn.preprocessing as prep
>>> help(prep.scale)
```