Rozpoznanie i klasyfikacja liter z ich obrazów

Maciej Niklas

Wstęp

Motywacja

Przeglądając zbiory danych zawarte w serwisie https://archive.ics.uci.edu/ natknąłem się na zbiór poświęcony rozpoznawaniu liter bazujący na statystykach zebranych z ich obrazów. Umiejętność ich rozpoznawania i klasyfikacji mogłaby mi pomóc stworzyć później projekt, który analizowałby dane graficzne i wydobywał niezbędne dane statystyczne w celu ekstrakcji zawartych na obrazie danych tekstowych. Pomysł powstał na skutek bieżącej sytuacji, gdzie na potrzeby innego przedmiotu realizowanego na Politechnice Gdańskiej studenci często zmuszeni są ręcznie przepisywać np. dane dwustu pomiarów, ponieważ zostały udostępnione w postaci zdjęcia. Zastosowań tego programu można by znaleźć jednak dużo więcej.

Cel

Celem jest identyfikacja każdej litery z dużej ilości czarno-białych, prostokątnych obrazów, jako jednej z wielkich liter alfabety angielskiego.

Opis danych

Pochodzenie

Zbiór danych, oryginalnie zatytułowany „Letter Image Recognition Data”, jest dziełem Davida J. Slate i powstał w styczniu, 1991 roku. Pochodzi z repozytorium uczenia maszynowego Uniwersytetu Kalifornijskiego w Irvine. Dane dostępne są pod adresem:
http://archive.ics.uci.edu/ml/datasets/Letter+Recognition

Struktura zbioru

Obrazy znaków oparte były na 20 różnych czcionkach, a każda litera została losowo zniekształcona, aby uzyskać zestaw danych zawierający 20000 unikalnych przypadków. Każdy z przypadków został podzielony na 16 atrybutów liczbowych, które następnie zostały przeskalowane do zakresu liczb całkowitych w granicach <0;15>.

Opis zmiennych

1. Letter wielka litera, 26 różnych wartości ze zbioru od A do Z
2. X-box pozycja pozioma obrazu
3. Y-box pozycja pionowa obrazu
4. Width szerokość obrazu
5. High wysokość obrazu
6. Onpix liczba pikseli zajmowanych przez literę
7. X-bar średnia ilość pikseli litery w poziomie
8. Y-bar średnia ilość pikseli litery w pionie
9. X2bar średnia wariancja w poziomie
10. Y2bar średnia wariancja w pionie
11. Xybar średnia korelacja między x i y
12. X2ybr średnia z x*x*y

13. Xy2br średnia z x*y*y
14. X-ege średnia ilość wierzchołków w poziomie
15. Y-ege średnia ilość wierzchołków w pionie
16. Yegvx korelacja x-ege z y-ege

Przygotowanie i analiza danych

W przypadku danych wykorzystywanych w tym zadaniu, ich proces przygotowawczy nie był wymagający. Twórca dopilnował, aby był kompletny oraz zbalansowany ilościowo pod względem dostępnych do sklasyfikowania klas. Proces przygotowania danych do obróbki polegał na przypisaniu odpowiednich nazw do kolejnych atrybutów. W następnej kolejności sprawdzeniu jakie typy zostały przypisane kolejnym atrybutom. Zgodnie z przewidywaniami zostały zinterpretowane jako typ liczby całkowitej, *int64*. Pierwszy atrybut, czyli klasy służące do klasyfikacji, został oznaczony jako typ *object*, co musiało zostać skorygowane, ponieważ jest on typek kategorycznym.

W efekcie dostępne klasy prezentują się następująco:

'T', 'I', 'D', 'N', 'G', 'S', 'B', 'A', 'J', 'M', 'X', 'O', 'R', 'F', 'C', 'H', 'W', 'L', 'P', 'E', 'V', 'Y', 'Q', 'U', 'K', 'Z'

Natomiast ich dystrybucja:

789 A 766 B 736 C 805 D 768 E 775 F 773 G 734 H 755 I 747 J 739 K 761 L 792 M
783 N 753 O 803 P 783 Q 758 R 748 S 796 T 813 U 764 V 752 W 787 X 786 Y 734 Z

W ramach przygotowania do obróbki przygotowałem funkcje pomocnicze:

- Funkcja standaryzująca zbiór danych

```
1. def Standarize(data):
2.     mean = data.mean()
3.     standardDeviation = data.std()
4.     return (data - mean) / standardDeviation
```

- Funkcja normalizująca zbiór danych

```
1. def Normalize(data):
2.     return (data - data.min()) / (data.max() - data.min())
```

- Funkcja obliczająca metryki klasyfikatora

```
1. import sklearn.metrics
2.
3. def ClassifierScore(alg, xTrain, xTest, yTrain, yTest):
4.     alg.fit(xTrain, yTrain)
5.     yTrainPred = alg.predict(xTrain)
6.     yTestPred = alg.predict(xTest)
7.     return {
8.         "Accuracy_Train": sklearn.metrics.accuracy_score(yTrainPred,
   yTrain),
9.         "Accuracy_Test": sklearn.metrics.accuracy_score(yTestPred,
   yTest),
10.          "Precision_Train":
   sklearn.metrics.precision_score(yTrainPred, yTrain, average='macro'),
11.          "Precision_Test":
   sklearn.metrics.precision_score(yTestPred, yTest, average='macro'),
```

```
12.          "Recall_Train":   sklearn.metrics.recall_score(yTrainPred,
   yTrain, average='macro'),
13.          "Recall_Test":   sklearn.metrics.recall_score(yTestPred,
   yTest, average='macro'),
14.          "F1_Train":  sklearn.metrics.f1_score(yTrainPred, yTrain,
   average='macro'),
15.          "F1_Test":  sklearn.metrics.f1_score(yTestPred, yTest,
   average='macro')
16.      }
```

W ostatnim kroku przygotowań stworzyłem osobne zbiory dla atrybutów oraz klas, aby móc je następnie podzielić na zbiór uczący oraz treningowy. Autor tego zestawu danych sugerował aby podziału dokonać w proporcjach 4:1, tak też zrobiłem.

Modelowanie danych

Do modelowania danych posłużyłem się trzeba algorytmami k-najbliższych sąsiadów, drzew decyzyjnych i lasów losowych, aby zbadać, który z nich przyniesie najlepsze rezultaty. Do każdego z wymienionych algorytmów dokonałem klasyfikacji dla zbioru oryginalnego, znormalizowanego oraz wystandaryzowanego. Następnie porównałem otrzymane metryki, aby wybrać tą z najlepszym wynikiem. Kolejnym korkiem było sprawdzenie dla jakiej liczby sąsiadów, głębokości drzew, czy liczby estymatorów wyniki były zbieżne dla zbioru testowego oraz trenującego. Na tej podstawie wykonywałem metodą zwaną *GridSearch* estymacji doboru parametrów klasyfikatora, aby ten dawał jak najlepsze wyniki.

Wnioski

Po analizie otrzymanych wyników najlepszy rezultat dał algorytm lasów losowych dla danych oryginalnych. Dokładność na poziomie 97,1%.

Dystrybucja zgodności wyników klasyfikacji dla poszczególnych klas prezentuje się następująco:

```
Z    1.000000
A    1.000000
C    0.993590
Q    0.993103
Y    0.987013
W    0.985816
X    0.982659
U    0.981250
L    0.981132
O    0.978873
N    0.977612
D    0.976608
B    0.974684
S    0.974026
M    0.971098
V    0.967320
R    0.966443
T    0.966102
P    0.963636
J    0.962264
F    0.952096
G    0.950549
K    0.944056
E    0.937931
```

```
I     0.937008
H     0.926829
```

Zbliżony wynik uzyskał także algorytm lasów losowych dla danych wystandaryzowanych oraz k-najbliższych sąsiadów dla danych oryginalnych i wystandaryzowanych, które wynosiły kolejno 97,06%, 95,98% oraz 95,26%.