PPPD - Lab. 09

Zadanie punktowane, lab 09, 2017/2018

Zbiór danych `train_wine` zawiera informacje na temat składu chemicznego win produkowanych z dwóch odmian (klas) winogron. Zbiór będziemy reprezentować jako macierz postaci:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{n1} & a_{n2} \end{bmatrix} \in \mathbb{R}^{n \times 2},$$

gdzie i-ty wiersz macierzy $\mathbf{A}$, $\mathbf{a}_i = (a_{i1}, a_{i2})$ opisuje dwie cechy fizykochemiczne i-tego wina, $i = 1, \ldots, n$.

Pierwsza kolumna (zmienna) opisuje liczbę gramów kwasu winowego na decymetr sześcienny trunku, zaś druga kolumna zawartość kwasu octowego.

Dodatkowo mamy dany wektor $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ taki, że $c_i \in \{0, 1\}$ określa do której z dwóch klas (odmian) należy i-te wino, $i = 1, \ldots, n$.

Macierz $\mathbf{A}$ i wektor $\mathbf{c}$ będziemy nazywać **zbiorem uczącym**.

Wyobraźmy sobie, że pojawia się teraz wektor opisujący własności fizykochemiczne jakiegoś nowego wina, $\mathbf{z} = (z_1, z_2)$. Wektor ten nie jest wierszem macierzy $\mathbf{A}$ – zatem jego klasa c nie jest znana. Naszym zadaniem będzie jej odgadnięcie (na podstawie informacji ze zbioru uczącego). W tym celu posłużymy się metodą *k najbliższych sąsiadów* (ang. *k-nearest neighbors*) w najprostszej wersji, tj. dla $k = 1$. Polega ona na:

1. znalezieniu wiersza macierzy $\mathbf{A}$, tj. wektora $\mathbf{a}_i$, dla którego odległość od wektora $\mathbf{z}$ jest najmniejsza, tzn. $d(\mathbf{a}_i, \mathbf{z}) = \min_{j=1,\ldots,n} d(\mathbf{a}_j, \mathbf{z})$, gdzie d oznacza odległość euklidesową; tak wyznaczony wektor $\mathbf{a}_i$ nazywamy *najbliższym sąsiadem* wektora $\mathbf{z}$;
2. przyporządkowaniu obserwacji $\mathbf{z}$ klasy jego najbliższego sąsiada, tj. c_i.

Innymi słowy, nowej obserwacji zostaje przyporządkowana klasa taka sama, jak klasa wina do niej najbardziej podobnego.

Wczytywanie danych i funkcja `distance()` [1 p.]

1. Wczytaj macierz `train_wine.csv` oraz `train_class.txt`. Upewnij się, że:

- wczytana ze zbioru `train_wine.csv` lista jest poprawnie zdefiniowaną macierzą o dwóch kolumnach;
- wczytana lista ze zbioru `train_class.txt` zawiera elementy ze zbioru $\{0, 1\}$ oraz że jej liczba elementów jest równa liczbie wierszy macierzy.

Uwaga: w każdym wierszu pliku `train_class.txt` znajduje się jedna wartość całkowita.

Uwaga: plik `train_wine.csv` możesz wczytać, korzystając z poleceń:

```
import csv
A = []
f = open("train_wine.csv", "r")      # r=do odczytu
for row in csv.reader(f):
    for i in range(len(row)):
        row[i] = float(row[i]) # konwersja z str na float
    list.append(A, row)        # == A.append(row)
f.close()
```

2. Napisz funkcję `distance(u, v)`, która wyznacza odległość Euklidesową między dwoma m-elementowymi listami liczbowymi:

$$d(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{i=1}^{m} (u_i - v_i)^2}.$$

Funkcja `nearest_neighbor_class()` oraz `knn()` [2 p.]

3. Napisz funkcję `nearest_neighbor_class(A, c, z)`, która przyjmuje jako argumenty:

   a. macierz uczącą $\mathbf{A} \in \mathbb{R}^{n \times m}$,
   b. listę reprezentującą wektor klas $\mathbf{c} \in \{0, 1\}^n$,
   c. listę reprezentującą wektor $\mathbf{z} \in \mathbb{R}^m$.

   Funkcja powinna wyznaczyć $i \in \{1, \ldots, n\}$ takie, że odległość $d(\mathbf{a}_i, \mathbf{z})$ jest najmniejsza z możliwych i zwrócić odpowiadającą wartość c_i.

4. Napisz funkcję `knn(A, c, Z)`, która przyjmuje jako argumenty:

   a. macierz uczącą $\mathbf{A} \in \mathbb{R}^{n \times m}$,
   b. listę reprezentującą wektor klas $\mathbf{c} \in \{0, 1\}^n$,
   c. macierz testową $\mathbf{Z} \in \mathbb{R}^{l \times m}$.

   Funkcja ta ma zwracać l-elementową listę o wartościach ze zbioru $\{0, 1\}$ taką, że jej i-ty element odpowiada odgadniętej klasie i-tego wiersza z macierzy $\mathbf{Z}$, por. funkcję `nearest_neighbor_class(A, c, z)`.

Wykres [2 p.]

5. Stwórz wizualizację działania implementowanej metody w następujący sposób:

   a. wygeneruj dwie listy $\mathbf{x}$ oraz $\mathbf{y}$ reprezentujące dwa ciągi arytmetyczne długości `N` (dla wybranego `N`) takie, że:

      ```
      x=[a0, a0+r, a0+2*r, ..., a1]
      y=[b0, b0+t, b0+2*t, ..., b1]
      ```

      gdzie `a0` i `a1` oznaczają, odpowiednio, najmniejszą i największą wartość w pierwszej kolumnie macierzy `A`, a `b0` i `b1` – najmniejszą i największą wartość w drugiej kolumnie macierzy `A`. Ponadto `r` i `t` to odpowiednio wyznaczone stałe rzeczywiste;

   b. dla każdego punktu z siatki (x_i, y_j) dla $i, j = 1, \ldots, N$ wyznacz odpowiadającą mu klasę (możesz skorzystać z funkcji `knn()` wywołanej na odpowiednio stworzonej macierzy zawierającej wszystkie punkty z siatki);

   c. wygeneruj rysunek, na którym naniesiesz punkty z macierzy `A` oraz z siatki (x_i, y_j) dla $i, j = 1, \ldots, N$, przy czym przynależność klasy każdego wektora oznaczysz różnymi kolorami.

      W tym celu skorzystaj z funkcji `plt.scatter()` w następujący sposób:

- kolor punktu możesz zmienić, korzystając z argumentu `color` (np. `b` – niebieski lub `r` – czerwony);
- dla punktów należących do siatki (x_i, y_j) ustaw przezroczystość koloru na wartość `0.2` (możesz to zrobić, korzystając z argumentu `alpha`) oraz kształt punktu na `.` (kształt punktu możesz zmienić, korzystając z argumentu `marker`);
- punkty do wykresu możemy dorysowywać, wywołując wielokrotnie funkcję `plt.scatter()`, np:

```
import matplotlib.pyplot as plt
fig = plt.figure()

plt.scatter(u, v, color=["b", "r", ...]) # dla list u i v (o takim samym rozmiarze)
            # punkty o wspolrzednych (u[i], v[i]) beda oznaczone
            # punktami o kolorach wskazanych przez kolejne elementy
            # listy ["b", "r", ...]
plt.scatter(w, q, marker=".", color=["b", "r", ...]) # dla list w i q (o takim samym rozmiarze)
            # punkty o wspolrzednych (w[i], q[i]) beda oznaczone
            # punktami o ksztalcie . i kolorach wskazanych przez kolejne elementy listy

fig.savefig("output.png", dpi=90)
```

Uwaga: im więcej punktów `N` rozważamy, tym dłuższy jest czas wykonania rysunku.

Ocena jakości klasyfikatora [2 p.]

6. Wczytaj pliki `test_wine.csv` (zbiór testowy). Korzystając z funkcji `knn()`, odgadnij dla każdego testowanego wina jego klasę.

7. Aby oszacować jak skutecznie funkcja `knn()` odgaduje klasę każdej testowanej obserwacji, porównaj otrzymany wynik z prawdziwą klasą zapisaną w pliku `test_class.txt`. Niech $\hat{c}_i$ oznacza prognozowaną (tzn. wyznaczoną przy użyciu funkcji `knn()`) klasę i-tego wina z macierzy testowej, a $\tilde{c}_i$ – klasę prawdziwą (na podstawie danych z pliku). Utwórz plik `output.txt`, który będzie zawierał następujące miary:

   - macierz pomyłek zdefiniowaną jako:

	0	1
0	TN	FP
1	FN	TP

     gdzie:

     – TN (ang. *true negatives*) oznacza liczbę wierszy testowych, dla których $\hat{c}_i = 0$ i $\tilde{c}_i = 0$,
     – FP (ang. *false positives*) oznacza liczbę wierszy testowych, dla których $\hat{c}_i = 1$ i $\tilde{c}_i = 0$,
     – FN (ang. *false negatives*) oznacza liczbę wierszy testowych, dla których $\hat{c}_i = 0$ i $\tilde{c}_i = 1$,
     – TP (ang. *true positives*) oznacza liczbę wierszy testowych, dla których $\hat{c}_i = 1$ i $\tilde{c}_i = 1$.

   - dokładność (ang. *accuracy*), tj. liczbę poprawnie zaklasyfikowanych win testowych:

$$A(\tilde{\mathbf{c}}, \hat{\mathbf{c}}) = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}};$$

   - precyzję (ang. *precision*), tj.:

$$P(\tilde{\mathbf{c}}, \hat{\mathbf{c}}) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}};$$

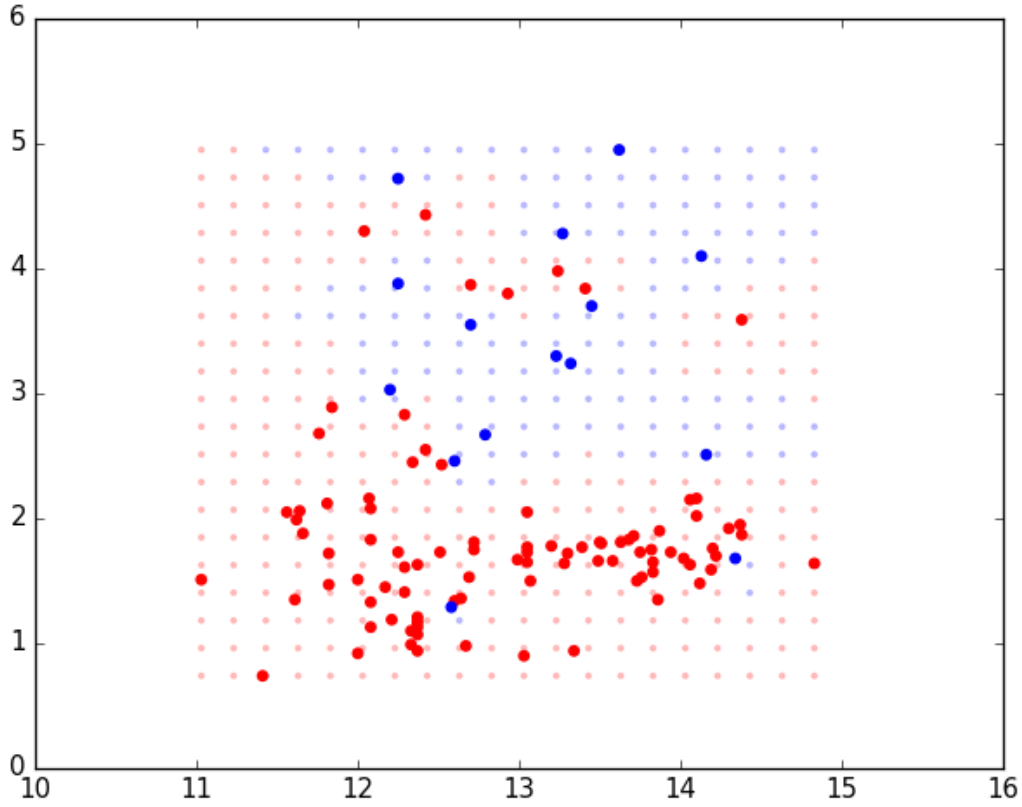
- czułość (ang. *recall*), tj.:
$$R(\tilde{\mathbf{c}}, \hat{\mathbf{c}}) = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}};$$
- miarę F_1, tj.:
$$F_1(\tilde{\mathbf{c}}, \hat{\mathbf{c}}) = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}.$$

Przykładowy plik `output.txt`:

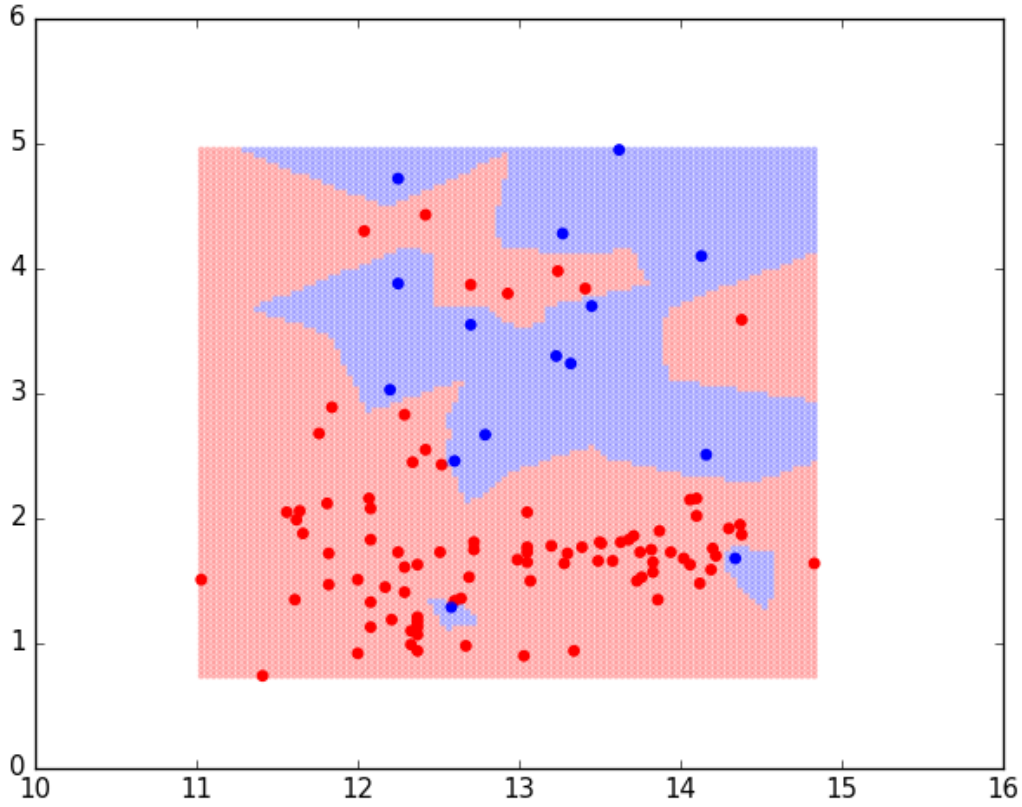
```
  |   0   |   1
-----------------
0 |  30   |   9
1 |   6   |  27

Dokladnosc =  0.79
Precyzja   =  0.75
Czulosc    =  0.82
Miara F1   =  0.78
```

Jakość kodu

- Przesłany skrypt musi wykonać się bez błędów.
- Kod musi być napisany w sposób czytelny.
- Kod musi być dobrze i dokładnie udokumentowany.



Rysunek 1: Przykładowy rysunek dla `N`= 20



Rysunek 2: Przykładowy rysunek dla `N`= 100