# PPPD - Lab. 05



*Zadanie punktowane, lab 05, 2017/2018*

## Treść zadania i punktacja

Algorytmy uczenia maszynowego dość często możemy dostrajać przy użyciu pewnej liczby tzw. hiperparametrów. Właściwy ich dobór czasem może istotnie wpłynąć na jakość uzyskiwanych wyników, np. skuteczność rozpoznawania wzorców na obrazkach.

Załóżmy więc, że mamy daną pewną funkcję `F(a, b)`, która jako argumenty przyjmuje dwie wartości rzeczywiste oraz zwraca jedną liczbę z przedziału $[0, 1]$. W naszym przykładzie $F : \mathbb{R}^2 \to [0, 1]$ może reprezentować właśnie skuteczność algorytmu klasyfikacji jako funkcję dwóch hiperparametrów $a$ i $b$, gdzie 0 oznacza metodę beznadziejnie słabą, a 1 – bezbłędnie rozpoznającą wszystkie wzorce.

W celach testowych użyjemy następującej funkcji (wklej ją po prostu na początek pisanego przez siebie skryptu):

```
import numpy as np
from sklearn import datasets, svm

def F(a, b):
    """
    Nie interesuje nas, co funkcja robi:
    traktujemy ja jako "czarna skrzynke".

    Wazne jest jedynie to, ze F(a, b) zwraca wartość z przedzialu [0,1]
    dla a, b > 0


    Przykład inspirowany http://scikit-learn.org/stable/auto_examples/
                         exercises/plot_iris_exercise.html
    """
    iris = datasets.load_iris()  # zbior iris
    X, y = iris.data, iris.target
    X, y = X[y != 0, :2], y[y != 0] # tylko klasy 1 i 2
    n_sample = len(X)
    np.random.seed(1234)
    order = np.random.permutation(n_sample)
    X = X[order]
    y = y[order].astype(np.float)
    X_train = X[:int(0.8 * n_sample)] # proba uczaca = losowe 80%
    y_train = y[:int(0.8 * n_sample)]
    X_test = X[int(0.8 * n_sample):]  # proba testowa = pozostale 20%
    y_test = y[int(0.8 * n_sample):]
    clf = svm.SVC(gamma=a, C=b) # support vector classifier
```

```
    clf.fit(X_train, y_train)
    return np.mean(clf.predict(X_test) == y_test) # accuracy, wartość z [0,1]
```

Oczywiście dobór właściwych hiperparametrów może być zadaniem bardzo trudnym. Jednym z najprostszych (i z reguły najbardziej skutecznych) podejść jest tzw. poszukiwanie po siatce (ang. *grid search*). Polega ono na obliczaniu wartości $F(a_i, b_j)$ dla wszystkich kombinacji wyrazów dwóch ciągów arytmetycznych $(a_i)_{i=1,2,\ldots,n}$ oraz $(b_j)_{j=1,2,\ldots,m}$ (tworzą one w istocie siatkę równoodległych punktów).

## Wczytywanie danych i sprawdzanie ich poprawności [2 p.]

Wczytaj z pliku `input.txt` (musisz go stworzyć samodzielnie) dane wejsciowe w postaci 6 liczb:

```
a1          # liczba rzeczywista - pierwszy wyraz ciągu arytmetycznego (a_i)
an          # liczba rzeczywista -  ostatni wyraz ciągu arytmetycznego (b_i)
n           # liczba całkowita - liczba wyrazów w ciągu (a_i)
b1          # liczba rzeczywista - pierwszy wyraz ciągu arytmetycznego (b_j)
bm          # liczba rzeczywista -  ostatni wyraz ciągu arytmetycznego (b_j)
m           # liczba całkowita - liczba wyrazów w ciągu (b_j)
```

Dzięki nim mamy dobrze określone ciągi arytmetyczne $(a_i)_{i=1,2,\ldots,n}$ oraz $(b_j)_{j=1,2,\ldots,m}$ o wyrazach rzeczywistych. Należy sprawdzić, czy $a_1 < a_n$, $b_1 < b_m$ oraz $n, m > 1$. Jeśli tak nie jest, natychmiast przerwij działanie programu.

## Wypisywanie wartości funkcji na danej siatce [2 p.]

Kolejnym krokiem jest obliczenie wartości funkcji $F$ w każdym punkcie z siatki i zapis wyników w postaci estetycznie sformatowanej tabelki, która zawarta będzie w pliku tekstowym `output.txt`.

np. dla pliku `input.txt` postaci:

```
10
100
10
0.25
2
8
```

powinniśmy uzyskać plik `output.txt` postaci podobnej do:

```
a \ b | 0.25  0.50  0.75  1.00  1.25  1.50  1.75  2.00
------|----------------------------------------------
10.0  | 0.70  0.70  0.70  0.70  0.65  0.65  0.65  0.70
20.0  | 0.70  0.70  0.70  0.70  0.70  0.70  0.70  0.75
30.0  | 0.55  0.70  0.70  0.70  0.70  0.70  0.65  0.65
40.0  | 0.40  0.70  0.70  0.70  0.65  0.60  0.60  0.60
50.0  | 0.40  0.65  0.70  0.65  0.60  0.60  0.60  0.60
60.0  | 0.45  0.65  0.65  0.65  0.60  0.60  0.60  0.60
70.0  | 0.45  0.60  0.65  0.65  0.60  0.60  0.60  0.60
80.0  | 0.45  0.60  0.65  0.65  0.65  0.65  0.65  0.65
90.0  | 0.45  0.55  0.60  0.65  0.65  0.65  0.65  0.65
100.0 | 0.45  0.50  0.60  0.65  0.65  0.65  0.65  0.65
```

## Znajdowanie wartości maksymalnej i minimalnej $F$ [1 p.]

Oblicz i wypisz na konsolę wartość `fmin` oraz `fmax` - największą i najmniejszą wartość, którą przyjmuje $F$, gdy obliczamy ją na siatce.

Ponadto wypisz na konsolę parę $(a_i, b_i)$, dla której wartość $F$ była największa - jesli jest więcej niż jedna para, zwróć dowolną z nich

Np. w powyższym przykładzie mamy `fmin=0.4`, `fmax=0.75`, a wartość maksymalna osiągana jest dla $a = 20, b = 2$.

### Rysowanie mapy ciepła [2 p.]

Jako ostatnie polecenie, narysuj (wynikiem będzie plik `output.png`) obraz danych zawartych w powyższej tabelce w postaci tzw. mapy ciepła (ang. *heat map*) - por. rysunek poniżej.

Tutaj wartość $F(a_i, b_j)$ reprezentowana jest w postaci prostokąta o środku $(a_i, b_j)$ i odpowiedniej barwie (stopniu szarości). Wartości $F$ należy przeskalować tak, żeby `fmin` miało kolor `"0.0"` (czarny), a `fmax` – `"1.0"` (biały). Wartości pośrednie będą oczywiście reprezentowane przez różne stopnie szarości.

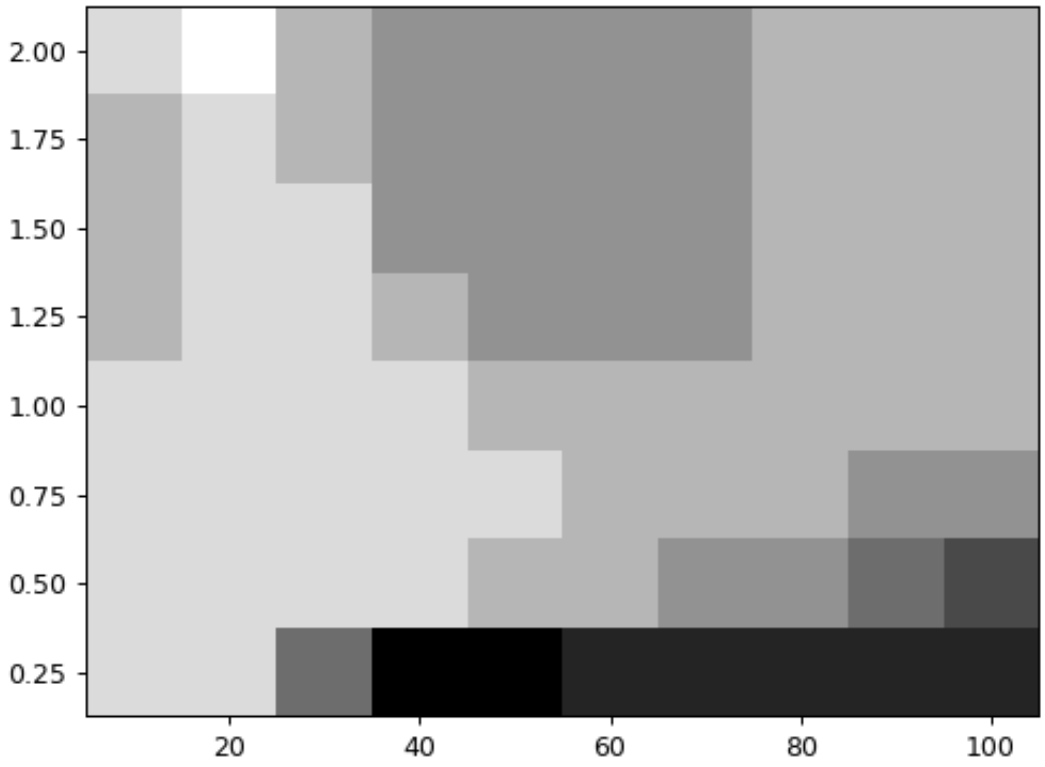
Oto fragment kodu, dzięki któremu poznasz, jak można rysować prostokąty i zapisywać je do pliku:

```
# inicjowanie rysunku:
import matplotlib.pyplot as plt
import matplotlib.patches as patches
fig = plt.figure()
ax = fig.add_subplot(111)

# ustal zakresy na osiach:
ax.set_xlim([0.0, 0.7])  # zakres wartości na osi OX [xmin, xmax]
ax.set_ylim([0.4, 2.85]) # zakres wartości na osi OY [ymin, ymax]


# przykładowy jasnoszary prostokąt o wierzchołkach (0.1, 0.5) i (0.6, 2.75):
ax.add_patch(patches.Rectangle(
        (0.1, 0.5),        # (x,y)
        0.5,               # szerokosc
        2.25,              # wysokosc
        facecolor=str(1-0.3) # stopien szarosci (od 0 do 1) jako napis
))

# ax.add_patch(patches.Rectangle(....)) można będziemy wywoływać wielokrotnie

# zapis do PNG po zakończeniu rysowania
fig.savefig('output.png', dpi=90)
```

## Uwagi i wskazówki

Nie mozna korzystać z list / tablic. Do rozwiązania wszystkich podpunktów wystarczą pętle.

Postaraj się możliwie zminimalizować liczbę wywołań funkcji $F$.




Rysunek 1: Mapa ciepła; kolor biały (1.0) oznacza maksymalną wartość funkcji $F$, a czarny (0.0) - minimalną