Middle East Technical University Department of Computer Engineering

# CENG 499

Introduction to Machine Learning

Spring 2019-2020

Homework 3: Support Vector Machines and Linear Regression

version 1.0

Due date: 9 May 2020, 23:59

## 1 Introduction

In this assignment, you will have the chance to get hands-on experience with support vector machines (SVM) and linear regression. Python is the programming language choice for this homework.

## 2 Support Vector Machines (40 pts)

In this task, you will improve your comprehension of the SVM by experimenting on various kernel functions and the hyperparameter C. You will employ the SVM implementation (called SVC, **link**) of **scikit-learn**, a popular machine learning library, for both subtasks. In each subtask, you will read the corresponding dataset with **NumPy** library, and plot the decision boundaries with **Mathplotlib**, a popular plotting library. No other external library is allowed.

### 2.1 Kernel Functions

The dataset file for this subtask is **task1_A.npz**. And you will read it by using the following code snippet. X is the feature matrix (100, 2), whereas y is the corresponding labels (100, ). Notice that the file is put under the directory task1 (Do not change the location).

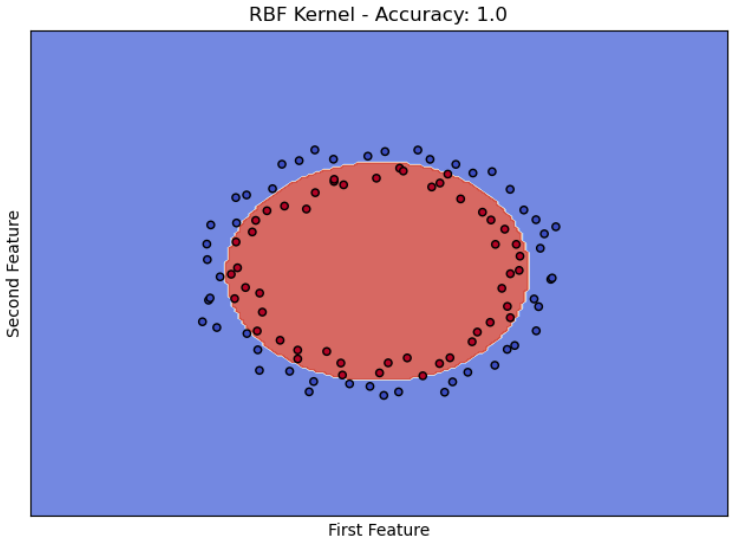
```
import numpy as np

data = np.load("task1\\task1_A.npz")
X, y = data["X"], data["y"]
```

Write your code in **task1_A.py**. When grading, your implementation will be called as follows.

```
python task1_A.py
```

By using the dataset, your program will train 4 different SVMs, each of which has different **kernel** parameter. The values you will try for the kernel parameter are 'linear', 'sigmoid', 'poly', and 'rbf'. Stick with the default values for the other parameters of the SVC. As the output, your program will show 4 decision boundary plots on the screen. We have provided an example decision boundary plot below. You are to state the axes, the kernel name, and the training accuracy in your plots.




## 2.2 The hyperparameter C

The dataset file for this subtask is **task1_B.npz**. And you will read it by using the following code snippet. X is the feature matrix (100, 2), whereas y is the corresponding labels (100, ). Notice that the file is assumed to be under the directory task1 (Do not change the location).

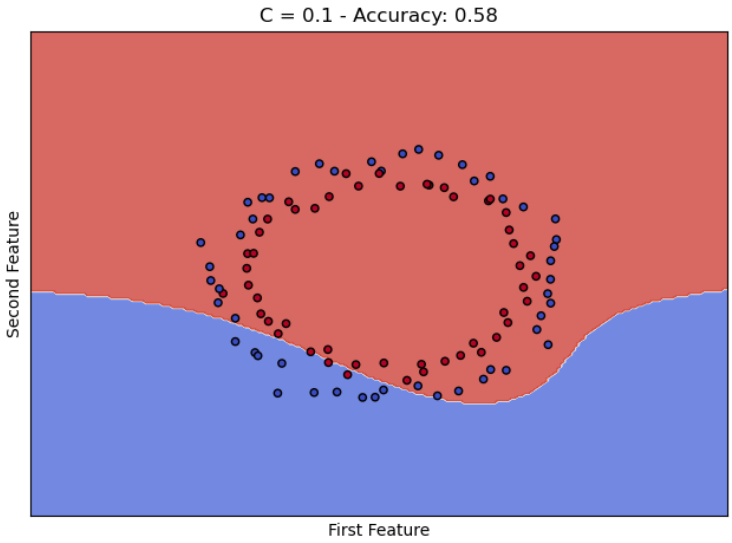
```
import numpy as np

data = np.load("task1\\task1_B.npz")
X, y = data["X"], data["y"]
```

Write your code in **task1_B.py**. When grading, your implementation will be called as follows.

```
python task1_B.py
```

By using the dataset, your program will train 4 different SVMs, each of which has different **C** parameter. The values you will try for the C parameter are 0.01, 0.1, 1, and 10. Use polynomial kernel. Stick with the default values for the other parameters of the SVC. As the output, your program will show 4 decision boundary plots on the screen. We have provided an example decision boundary plot below. You are to state the axes, the C value, and the training accuracy in your plots.




# 3 Linear Regression (60 pts)

In this section, you are going to perform a regression on a dataset by using multivariate linear regression. Specifically, you will parse the input dataset, perform normalization on it, train the model with gradient descent, and compute the performance metric on the test set. You will write your code in the function **linear_regression** in **task2.py**. When grading, we will call the function with different parameters. The dataset for this task is taken from here (**link**).

- No external library is allowed for this task.
- As we stated above, we will use different datasets. Although the datasets we are going to use in grading have the same structure as the provided dataset, they have different number of features. In short, do not assume a fixed value for the number of features.
- To normalize both sets, use the following equation. $X$ is a feature, $X_{min}$ is the minimum value for the feature $X$, and $X_{max}$ is the maximum value for the feature $X$. Also, do not apply normalization on the target value

(the last column).

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- The performance metric for this task is RMSE (Root Mean Square Error). $m$ is the number of instances in the set, $y_i$ is the value of the $i^{th}$ instance, $\hat{y}_i$ is the predicted value of the $i^{th}$ instance.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$$

- For the provided dataset, the RMSE value you should achieve is 4.76 (whenn num_epochs is 1000 and learning_rate is 0.001).

# 4 Specifications

- There is a time limit for a (sub) task. If a (sub) task takes more than 5 minutes to achieve its result, you will get a zero point from that (sub) task.
- Falsifying results, changing the composition of training and test data are strictly forbidden, and you will receive 0 if this is the case. Your programs will be examined to see if you have actually reached the results and if they are working correctly.
- Using any piece of code that is not your own is strictly forbidden and constitutes as cheating. This includes friends, previous homeworks, or the internet. The violators will be punished according to the department regulations.
- Follow the course page on ODTUClass for any updates and clarifications. Please ask your questions on the discussion forum of ODTUClass instead of e-mailing.
- You have total of 3 late days for **all** your homeworks. For each day you have submitted late, you will lose 10 points. The homeworks you submit late after your total late days have exceeded 3 will not be graded.

# 5 Submission

Submission will be done via ODTUClass. You will submit a zip file called **hw3.zip** that contains **task1_A.py**, **task1_B.py**, and **task2.py**.