Homework 2 Report **Piero Macaluso s252894**
Machine Learning and Artificial Intelligence 2018/2019 Due Date: 19/01/2019
Prof. Barbara Caputo

1 Linear SVM

In the first part of this homework I trained, validated and finally tested a Linear SVM, using the C with the highest accuracy in validation. The essential part of the code is algorithm 1.

Algorithm 1: Searching the best value of C in Linear SVM

```
87      c = 1e-3
88      c_best = 1e-3
89      a_best = 0
90      i = 1
91      xx, yy = make_meshgrid(x[:, 0], x[:, 1])
92      while c <= 1e3:
93          clf = svm.LinearSVC(C=c, random_state=r_state)
94          acc[i - 1] = clf.fit(x_train, y_train).score(x_val, y_val) * 100
95          if acc[i - 1] > a_best:
96              c_best = c
97              a_best = acc[i - 1]
98          ax = fig.add_subplot(4, 2, i)
99          plot_data(ax, x_train, y_train, xx, yy, clf)
100         ax.set_xlim(xx.min(), xx.max())
101         ax.set_ylim(yy.min(), yy.max())
102         ax.set_xlabel('Sepal length')
103         ax.set_ylabel('Sepal width')
104         ax.set_xticks(())
105         ax.set_yticks(())
106         ax.legend()
107         ax.set_title('C=%2.2E A=%2.1f%% ' % (c, acc[i - 1]))
108
109         c = c * 10
110         i = i + 1
```

As we can see from the plots figs. 1 and 2 on the next page and on page 3, the best accuracy found on validation set was **73.33%**, with C equal to **1e−1**.

I noticed that boundaries became more and more precise on the training set with the increment of C.

C is the hyper parameter of SVM which describe how much we want to avoid misclassification during the training. The optimization will choose a smaller-margin hyperplane with a **large** C. The larger is C, the better the classification on training set will be. On the contrary, a **small value of** C will cause the optimizer to use a larger-margin separating hyperplane, causing the misclassification of more training data points.

After this training, I tests the data on the test set obtaining a greater accuracy, it goes very well with an accuracy of **88.89%** as we can see in fig. 3 on page 3. I tried to repeat the code a lot of times using a different `random_state` variable, obtaining lower accuracy, greater accuracy or the same validation one. This may be due to the initial state of the algorithm of SVM or the data in train, validation and test sets: in a real case (not influenced by `random_state` for repeatability purpouse) these things will be different at each run.

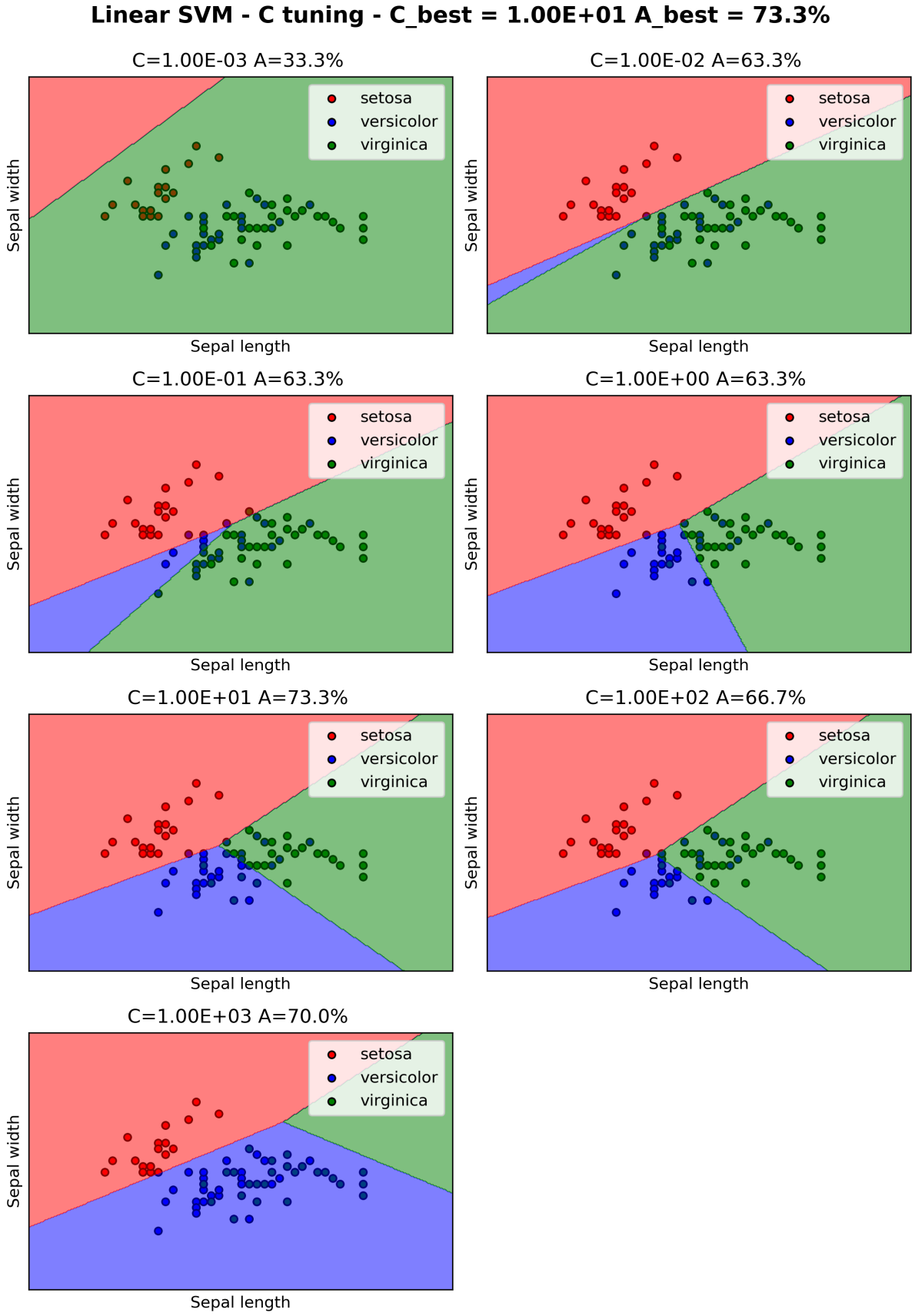

Figure 1: Decision Boundaries changing C in Linear SVM on Training set

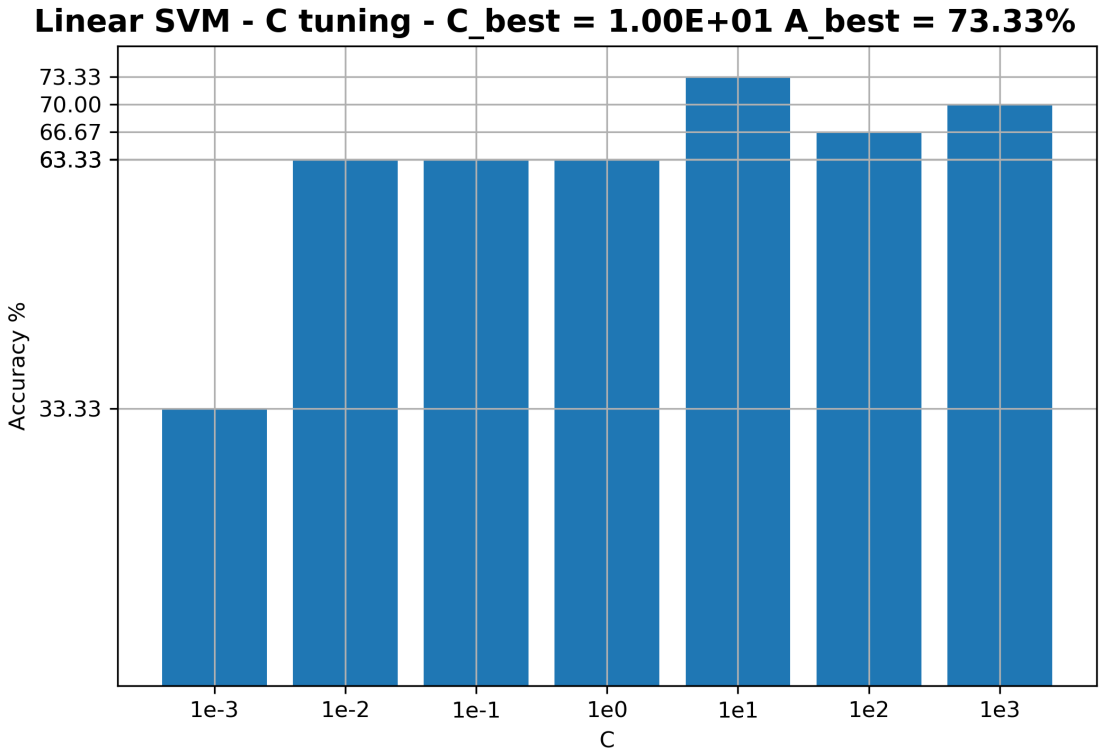

Figure 2: Accuracy changing C in Linear SVM

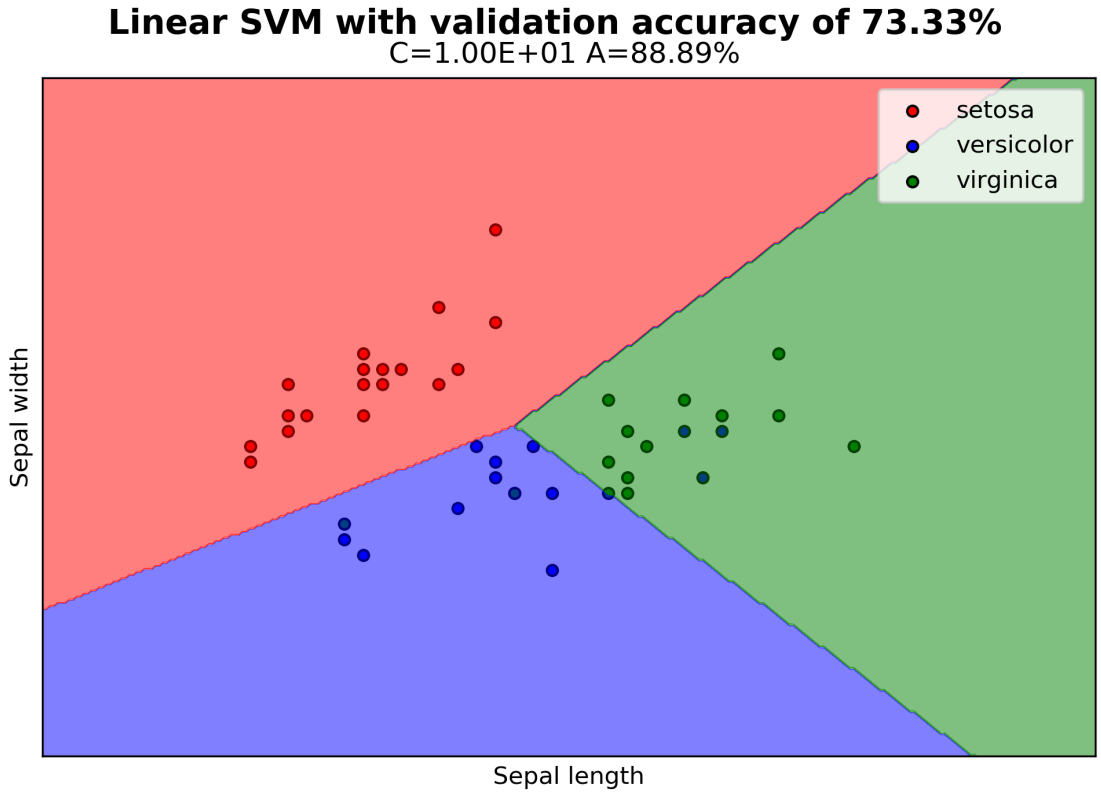

Figure 3: Results on the test set

2 RBF Kernel

In the second part of this homework I used SVM with RBF kernel. This time I had to find the best pair of C and I standardized `x` using `StandardScaler` making each feature zero-mean and unit-variance. Then I applied PCA on the normalized X obtaining the projection using all 1087 PCs as in **??** on page ??. I decided to implement a custom function called `reconstruction()` in order to re-project `x_t` using first 60, 6, 2 and last 6 principal component (PC) and visualize the reconstructed images with `show_reconstruction()` function as you can see in **??** on page ??.

3 Classification

The formulation of Naïve Bayes is described in eq. (1).

$$\hat{y} = \underset{i \in \{1,\dots,k\}}{\operatorname{argmax}} p(y_i \mid x_1, \dots, x_d) = \underset{i \in \{1,\dots,k\}}{\operatorname{argmax}} \overbrace{p(y_i)}^{Prior} \prod_{j=1}^{d} \overbrace{p(x_j \mid y_i)}^{Likelihood} \tag{1}$$

where

$\hat{y_i}$ = predicted label
y_i = i-th label with $i \in \{1, \dots, k\}$
x_j = j-th example with $j \in \{1, \dots, d\}$
$p(x \mid y)$ = Gaussian

4 Code Execution

4.1 Requirements

- Python 3
- All dependencies in `requirements.txt`.

  `$ pip install -r requirements.txt` to install them

4.2 Usage

- `$ python program.py -n <PACS_homework folder>`

  Loads Image from the specified `<PACS_homework folder>` and execute the code

- `$ python program.py -s <PACS_homework folder>`

  Loads Image from the specified `<PACS_homework folder>` and save files in `data.npy` and `label.npy` for faster next execution before executing the code

- `$ python main.py -l <data.npy file> <label.npy file>`

  Loads Image from `data.npy` and `label.npy` files and execute the code

4.3 Reproducibility

In order to reproduce the same data for this experiment you have to change the global variable `r_state` (line) from `None` to 252894 which is my badge number.

Attachments

- Source Code:
  - `main.py`
  - `requirements.txt`