# Machine Learning Fundamentals
# Lab 3

**Shubham Agarwal**
Master 2 MSIAM Data Science
shubhamagarwal92@gmail.com

## 1 Introduction and theoretical background

- Y denotes the labels. Usually Y = -1,1 (binary classification)
- $x = (x_1, \ldots, x_p) \in X \subset R^P$ are the features
- $D_n = \{(x_i, y_i), i = 1, \ldots, n\}$ is the training set
- We assume that there exists a probabilistic model explaining the generation of our observations :
  $\forall i \in \{i = 1, \ldots, n\}, (x_i, y_i) iid \sim (X, Y)$
- Using the training set $D_n$ we want to construct a prediction function $\hat{f} : X \to \{-1, 1\}$ which predicts an output y for a given new x with a minimum probability of error. Here the decision rule will be linear, that is we separate the space by an hyperplane and we predict -1 or 1 according to the position of x with respect to this hyperplane.

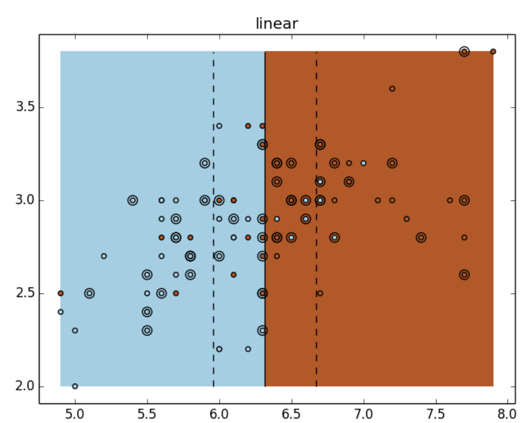
## 2 SVM in practice

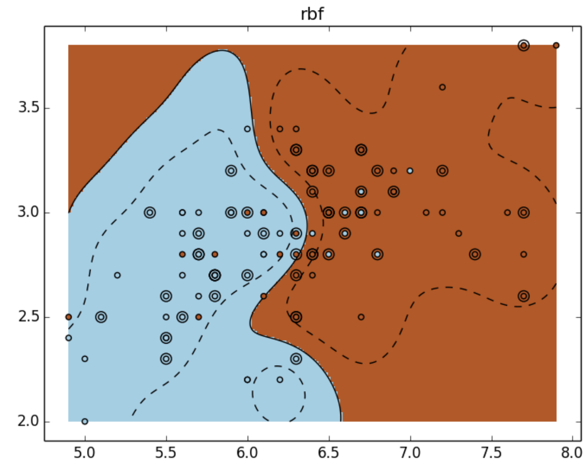
We shall use the object sklearn.svm.SVC :

from sklearn.svm import SVC

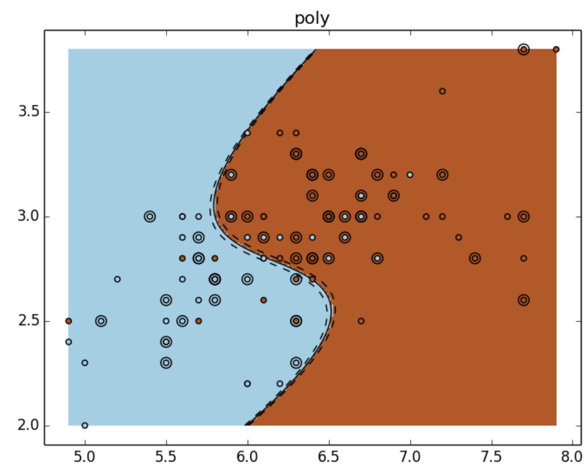
1. Implement a classifier which classifies class 1 against class 2 of the dataset iris using the two first variable and a linear kernel. Use half of the dataset for training and half of the dataset for validation.

Using half of the dataset for training and half of the dataset for validation and using default values of C and gamma for linear and polynomial kernel, we obtain the following curves:






2. Compare the result with a SVM based on polynomial kernel




3. Prove that the primal problem can also be reformulated as follows

We know that in H, we obtain the separating hyperplane maximizing the margin separating the two classes, that is solving the following optimization problem :

$(w^*, w_0^*, \xi^*) \in argmin_{w \in H, w_0 \in R, \xi \in R^n}(\frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i)$

such that $\xi_i \geq 0$ and $y_i(< w, \Phi(x_i) > +w_0) \geq 1 - \xi_i$

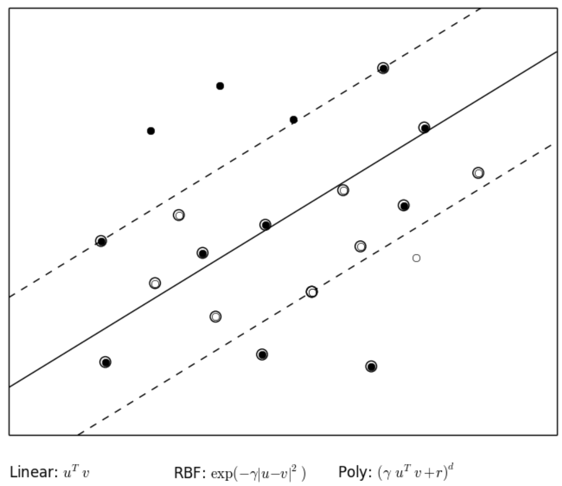
We have two constraints on $\xi_i$ with $\xi_i \geq 1 - y_i(< w, \Phi(x_i) > +w_0)$ and at the same time $\xi_i \geq 0$

Thus if we replace $\xi_i$ by the two constraints in the optimisation problem, we get

$(w^*, w_0^*) \in argmin_{w \in H, w_0 \in R}(\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}[1 - y_i(< w, \Phi(x_i) > +w_0)]_+)$

4. Use the script svm gui.py available on the website. This application allows to evaluate the impact of the choice of the kernel and the regularisation parameter C.

We can use the script svm_gui.py to generate the data points as shown below:




5. Generate a dataset with much more observations in one class than another (for e.g. 90% vs 10%).

6. Use a linear kernel and decrease the parameter C. Comment

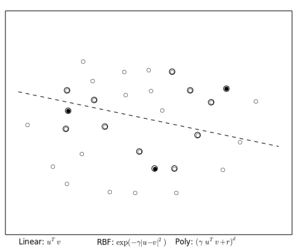


Figure 1: Biased Data with C =0.5

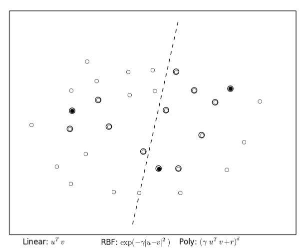


Figure 2: Biased Data with C =1

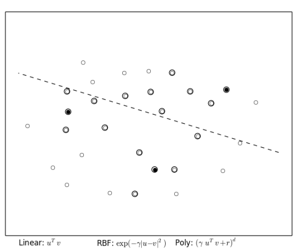


Figure 3: Biased Data with C =10

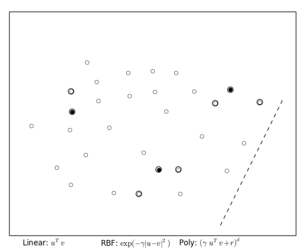
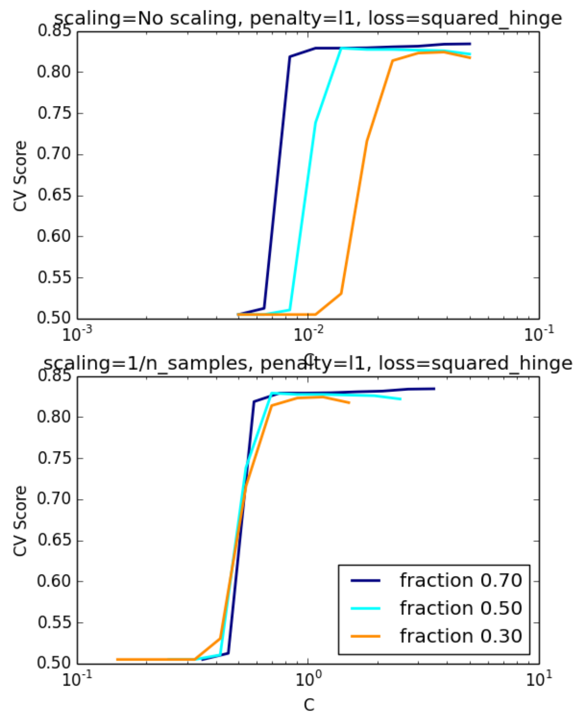
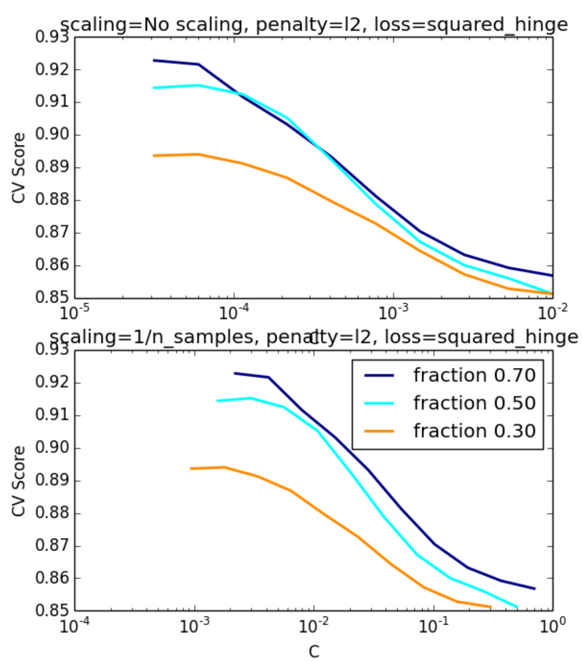


Figure 4: Biased Data with C =100

C is used to set the amount of regularization. A large value of C basically tells our model that we do not have that much faith in our data?s distribution, and will only consider points close to line of separation. A small value of C includes more/all the observations, allowing the margins to be calculated using all the data in the area.

In the l1 penalty case, the cross-validation-error correlates best with the test-error, when scaling our C with the number of samples, n, which can be seen in the first figure. For the l2 penalty case, the best result comes from the case where C is not scaled.



**A practical example : faces classification**

The following example is a faces classification problem.

1. Show the influence of the regularisation parameter

C is used to set the amount of regularization. As shown in the previous question.

Expected results for the top 5 most represented people in the dataset:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Ariel Sharon | 0.67 | 0.92 | 0.77 | 13 |
| Colin Powell | 0.75 | 0.78 | 0.76 | 60 |
| Donald Rumsfeld | 0.78 | 0.67 | 0.72 | 27 |
| George W Bush | 0.86 | 0.86 | 0.86 | 146 |
| Gerhard Schroeder | 0.76 | 0.76 | 0.76 | 25 |
| Hugo Chavez | 0.67 | 0.67 | 0.67 | 15 |
| Tony Blair | 0.81 | 0.69 | 0.75 | 36 |
| avg / total | 0.80 | 0.80 | 0.80 | 322 |

2. Explain why the features are centered and reduced

Usually, each row is an “observation” (image), and each column is a variable (pixel value). Therefore, we should center and scale the columns before doing PCA or classification. We’d like in this process for each feature to have a similar range so that our gradients (in case of gradient descent training) don’t go out of control (and that we only need one global learning rate multiplier).

3. What is the effect of the choice of a non linear kernel RBF on prediction? You may improve the prediction properties using a dimension reduction based on the object sklearn.decomposition.RandomizedPCA

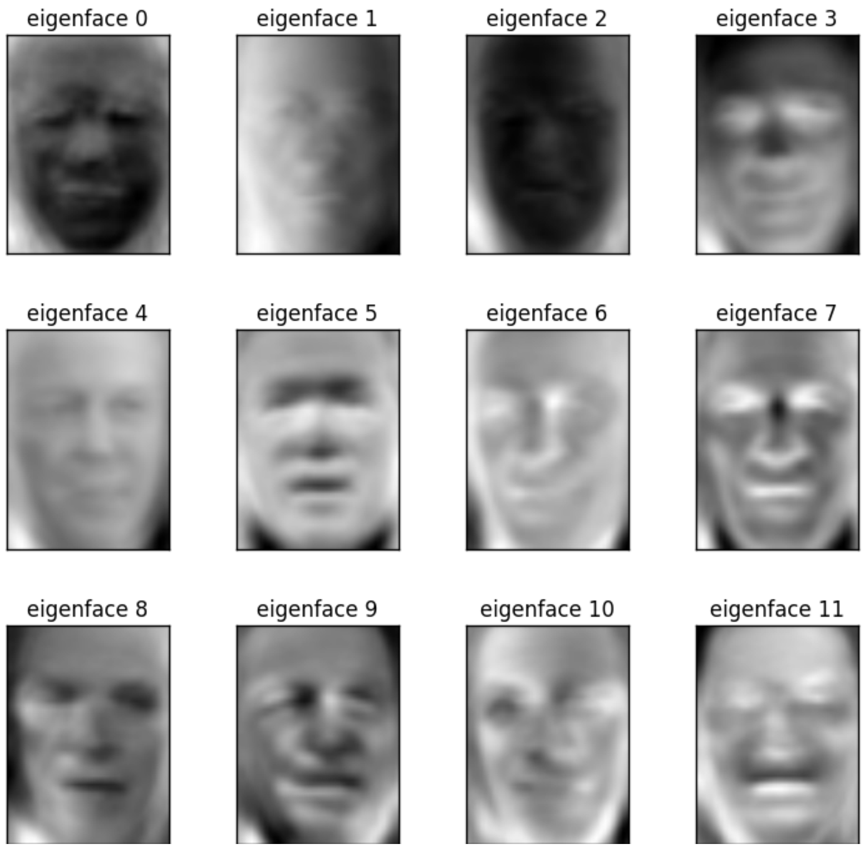
For the images, non-linear Kernel improves prediction. PCA also improves the prediction properties. The results produced are thus:

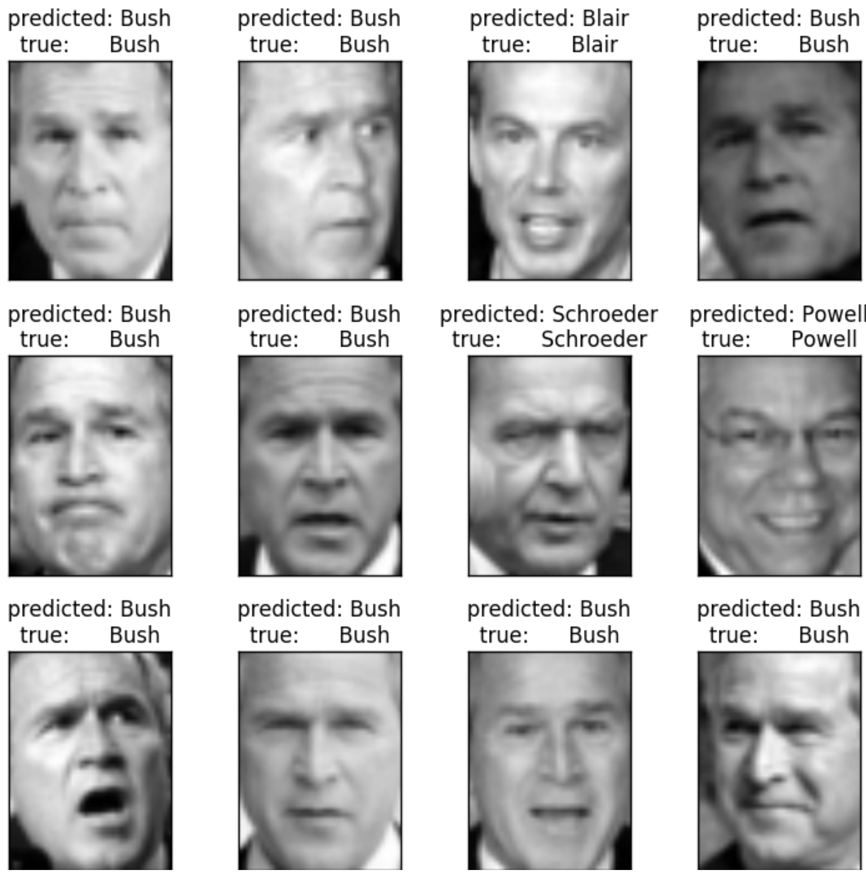
```
2017-02-14 09:44:32,991 Downloading LFW data (~200MB): http://vis-www.cs.umass.edu/lfw/lfw-funneled.tgz
2017-02-14 09:46:04,006 Decompressing the data archive to /Users/nitika/scikit_learn_data/lfw_home/lfw_funneled
2017-02-14 09:46:16,591 Loading LFW people faces from /Users/nitika/scikit_learn_data/lfw_home
2017-02-14 09:46:17,116 Loading face #00001 / 01288
2017-02-14 09:46:23,408 Loading face #01001 / 01288
Total dataset size:
n_samples: 1288
n_features: 1850
n_classes: 7
Extracting the top 150 eigenfaces from 966 faces
done in 0.482s
Projecting the input data on the eigenfaces orthonormal basis
done in 0.041s
Fitting the classifier to the training set
done in 25.333s
Best estimator found by grid search:
SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
Predicting people's names on the test set
done in 0.048s
                   precision    recall  f1-score   support

     Ariel Sharon       0.56      0.77      0.65        13
     Colin Powell       0.76      0.88      0.82        60
  Donald Rumsfeld       0.74      0.74      0.74        27
    George W Bush       0.93      0.88      0.90       146
Gerhard Schroeder       0.76      0.76      0.76        25
      Hugo Chavez       0.69      0.60      0.64        15
       Tony Blair       0.90      0.78      0.84        36

      avg / total       0.84      0.83      0.83       322

[[ 10   0   3   0   0   0   0]
 [  3  53   1   2   0   1   0]
 [  4   1  20   1   0   1   0]
 [  1  10   2 128   3   1   1]
 [  0   1   0   3  19   1   1]
 [  0   2   0   1   2   9   1]
 [  0   3   1   3   1   0  28]]
```


eigenface 0
eigenface 1
eigenface 2
eigenface 3
eigenface 4
eigenface 5
eigenface 6
eigenface 7
eigenface 8
eigenface 9
eigenface 10
eigenface 11




**More about the duality gap**

The example example-svm-plot-separating-hyperplane-py explain how access to the parameters of the model (w : coef , $w_0$ intercept).

Code in lab3c.py

1. Use this example to implement a code calculating the value of the primal and dual functional. Check that these two values are close

coef_: Weights assigned to the features (coefficients in the primal problem). This is only available in the case of a linear kernel.

intercept_: Constants in decision function.

2. How does the difference between the two vales vary when the parameter tol of SVC vary?

Tol: Tolerance for stopping criteria. (default=1e-4) When tol of SVC increases, the difference between the values increases.

```
tolerance = 0.1
Euclidean distance between coeff: [[ 0.31917728]]
Absolute distance between coeff: [ 0.30404557  0.09711039]
Euclidean distance between weights (coeff+intercept): [[ 0.31930042]]
Absolute distance between weights (coeff+intercept): [ 0.30404557
0.09711039  0.00886698]

tolerance = 0.001
Euclidean distance between coeff: [[ 0.00333285]]
Absolute distance between coeff: [ 0.001258    0.00308631]
```

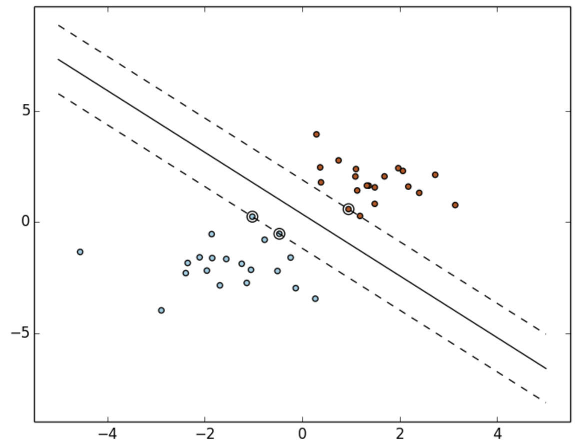



Figure 5: Hyperplane found by LinearSVC

```
Euclidean distance between weights (coeff+intercept): [[ 0.00392837]]
Absolute distance between weights (coeff+intercept): [ 0.001258
0.00308631  0.00207947]

tolerance = 0.0001
Euclidean distance between coeff: [[ 0.00020386]]
Absolute distance between coeff: [  1.98165333e-04
4.78592569e-05]
Euclidean distance between weights (coeff+intercept): [[ 0.00023808]]
Absolute distance between weights (coeff+intercept): [
1.98165333e-04   4.78592569e-05   1.22971077e-04]
```