Report for Sheet 4

Lab Course Machine Learning and Data Analysis

Hiromasa Okada

July 3, 2017

Implementation comments

In this exercise I implemented support vector machine and plot function of it. There are two type of support vector machines, the first one is trained by SMO algorithum and the second one is with direct solving of quadratic programming.

```
# SVM class with SMO algorithum
smo_svm = svm_smo(kernel, C, kernelparameter)

# SVM class with quadratic problem
qb_svm=svm_qp(kernel, C, kernelparameter)
```

Both class get $kernel \in \{'gaussian', 'polynomial', 'linear'\}$ and C as a contraint parameter for lagrange multipliers. Both two classes has same kernel function "getkernel(self, X, Y=None)" and predict function "fx(self,X1,X2,Y)". On the other hands SMO class has following functions which QP class does not have.

```
_compute_box_constraints(self, i, j, Y, alpha, C)
_update_parameters(self, E_i, E_j, i, j, K, Y, alpha, b, C)
_compute_updated_b(self, E_i, E_j, i, j, K, Y, alpha_old, alpha_new, b_old, C)
```

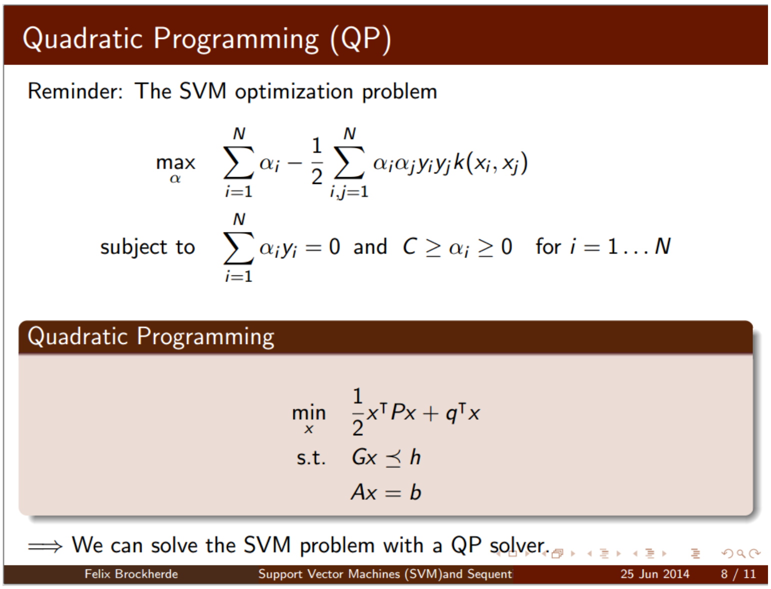
With these functions SMO class finds iteratively optimal lagrange multipliers and bias. But QP class solve quadratic programming in the function "fit(self, X, Y)" without functions above. In both classes lagrange multipliers, bias, support vector and labels are saved as object like

```
self.SV= X[alp_idx]
self.y = Y[alp_idx]
```

```
self.alpha = self.alpha[alp_idx]
self.b = np.mean(self.y - self.fx(self.SV,self.SV,self.y)[:,0])
```

Assignment 3

The formular of the SVM optimization problem and the quadratic programming are given in 8th page of the slide.



The relation between them:

$$
\begin{aligned}
&\max_{\alpha} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\
= &\min_{\alpha} \quad -\sum_{i=1}^{N} \alpha_i + \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j)
\end{aligned}
$$

$$
x = (\alpha_1, ..., \alpha_N)^T, \quad q = (-1, ... -1)^T
$$

$P = Y \odot K,$ Where $\odot$ elementwise multiplication and $Y, K \in \mathbf{R}^{N \times N}$

$$Y = yy^T, \quad y = (y_1, ...y_i)^T \in \{-1, 1\}^N, \quad K = \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_j) \\ \vdots & \ddots & \vdots \\ k(x_i, x_1) & \cdots & k(x_i, x_j) \end{pmatrix}$$

$$G = \begin{bmatrix} -I \\ I \end{bmatrix}, \quad h = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ C \\ \vdots \\ C \end{pmatrix} \quad \text{Where } I \text{ is identical matrix and } G \in \mathbf{R}^{2N \times N}, \quad h \in \mathbf{R}^{2N}$$

$$A = (y_1, ..., y_N) \in \mathbf{Z}^{1 \times N}, \ \forall_{i=0}^{N} \, y_i \in \{-1, 1\}, \quad b = [0] \in \mathbf{R}^{1 \times 1}$$

Assignment 4

1. Find parameters C and for a Gaussian kernel

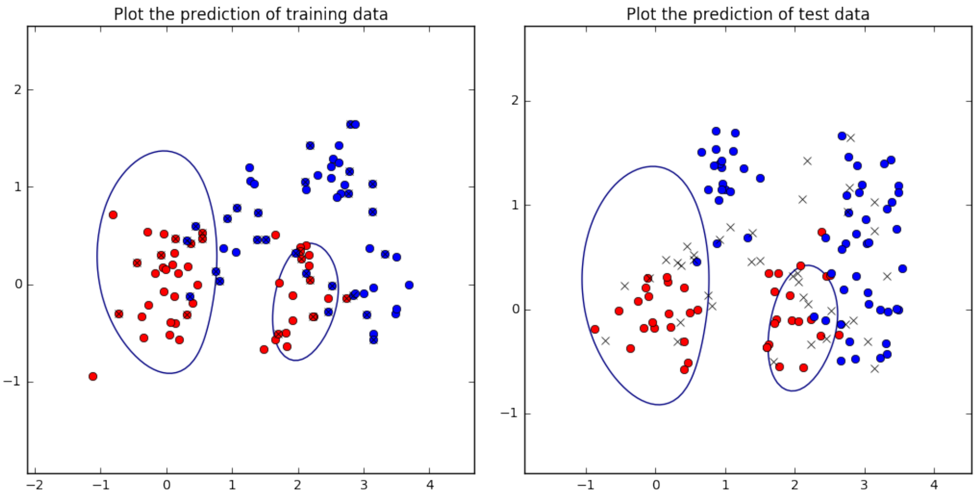
To find parameters C and for a Gaussian kernel I used the cross validation with following parameters

```
para = { 'kernel': ['gaussian'], 'kernelparameter': np.logspace(-2,2,10),
'regularization': np.linspace(1.0,3.0,10)}
```

Then the parameters C and for a Gaussian kernel are

```
Kernelparameter:  0.599484250319
C:  2.77777777778
```

After the plot of predictons of training data and test data I found that it classfied well and there was no over fitting and no under fitting by traing data so that it classfied also test data well.

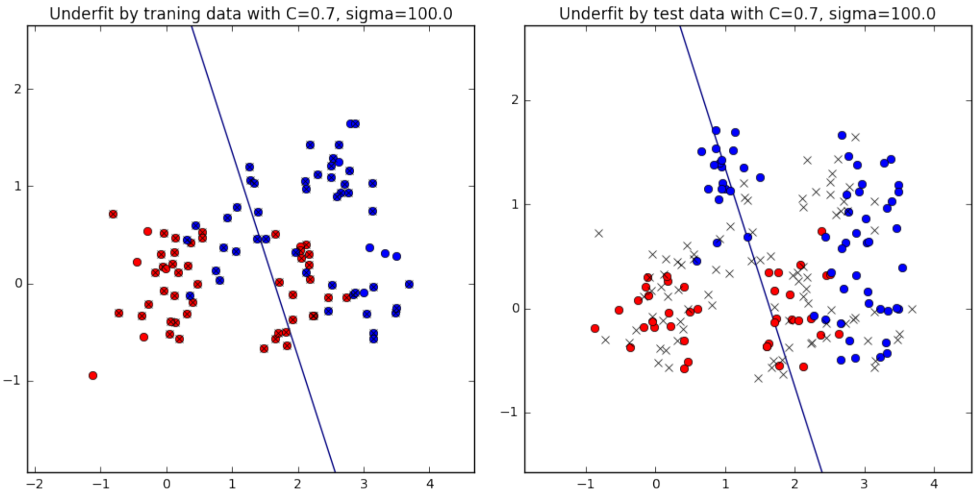


2. Over fitting and under fitting

Which parameter C and for a Gaussian kernel overfit and underfit? The parameters which cause underfitting are

```
Kernelparameter:  100.0
C:  0.7
```

and plots are

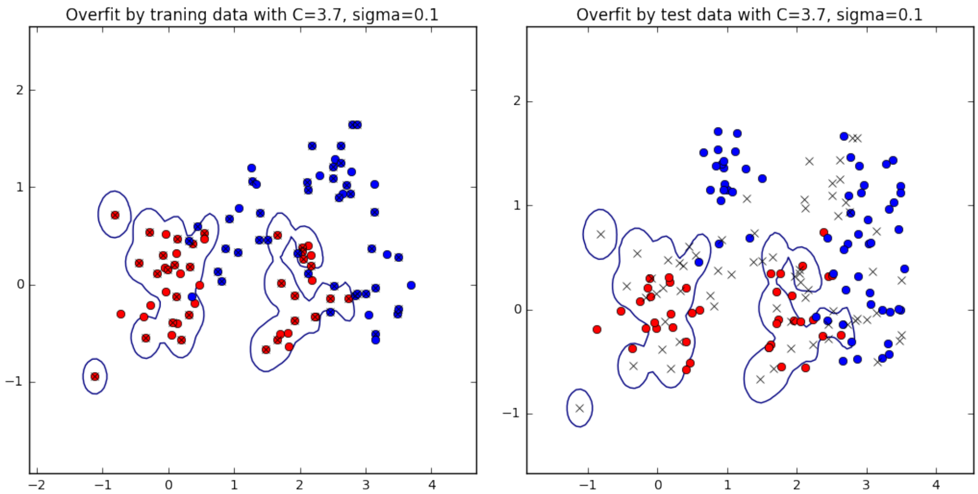


In case of the under fitting the classfier will be a linear.

The parameters which cause overfitting are

```
Kernelparameter:  0.1
C:  3.7
```

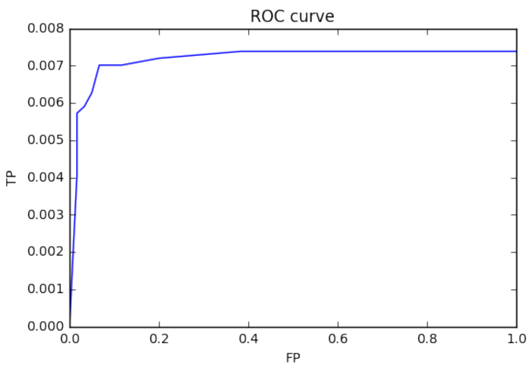
and plots are



In case of the overfitting there are more than two clusters even we don't need more. When the sigma is higher and C is lower than optimal it tends to underfit, on the other hands the sigma is lower and C is higher than optimal it tends to overfit.

3. ROC curve with different bias

For optimal C and σ, a receiver operator characteristics (ROC) curve is plotted by varying the bias parameter b of your SVM model.



The used possible bais are in this range:

```
bias = np.linspace(-10,10,110)
```

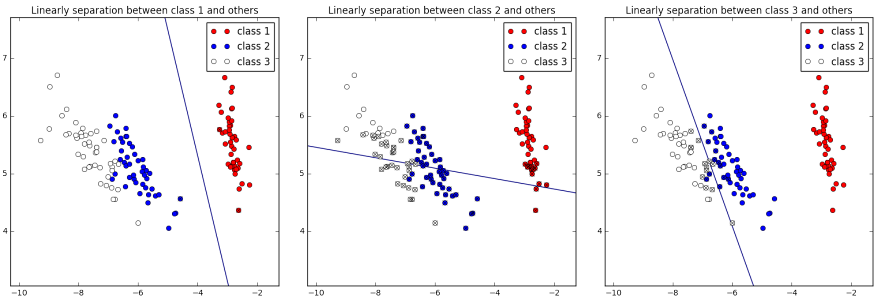
Assignment 5

Assignment 6

First to visualize the separation by a hyperplane the dimensions of datas are reduced by PCA.

```
pca= PCA(X)
X_pro = pca.project(X,2)
```

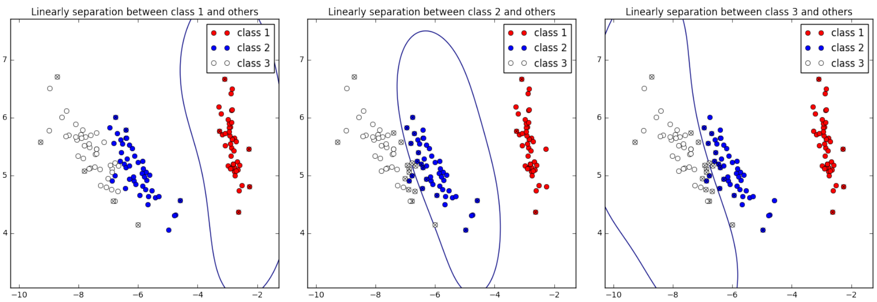
In case of linear separation class 1 and class 3 are well separated from other classes but class 2 are not. Because class 2 lies in the center of classes and it is not possible to separate by a linear line.



The losses for three linear separations are

```
Loss of linearly separation between class 1 and others:  0.0
Loss of linearly separation between class 2 and others:  0.2814814814814815
Loss of linearly separation between class 3 and others:  0.037037037037037035
```

In case of non-linear separation(gaussian) all of three classes are separated well. Non-linear separation line can be a circle line therefore it is also usuful for classification of this dataset.




```
Loss of non-linearly separation between class 1 and others:  0.0
Loss of non-linearly separation between class 2 and others:  0.2814814814814815
```

```
Loss of non-linearly separation between class 3 and others:  0.037037037037037035
```

Assignment 7