Assignment #4 – Machine Learning – Professor Haugh

Jaime Gacitua

jg3499

Monday March 7, 2016

Question 1

Scaling the variables must be part of the exploratory data analysis (EDA) at the beginning of any research task.

From one perspective, It is a good idea to scale all of the variables, because if, for example a variable has units in inches, and another variable in miles, the absolute values could lie in very different orders of magnitude. When finding the maximum margin classifier, for the case that the classes are not linearly separable, the variables with higher values will get more weight in the penalizing process. Scaling the variables will make them all be equally penalized.

Taking another perspective, we have seen in class that initially a data set can be easily linearly separable. When we scale the variables, separation between classes could become less apparent, making the classification task even more difficult.

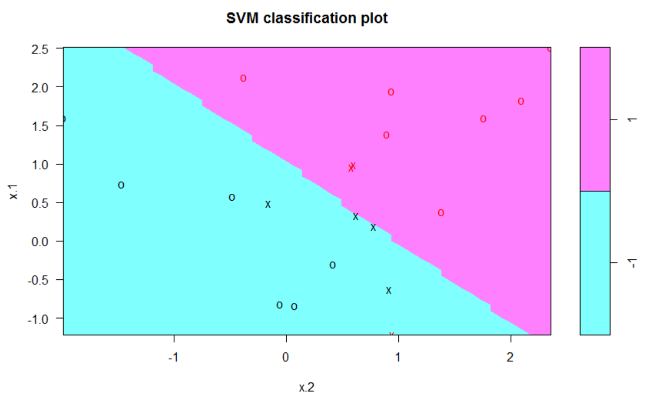
As a conclusion, I would advise running several EDA routines, to make sure the latter case does not happen.

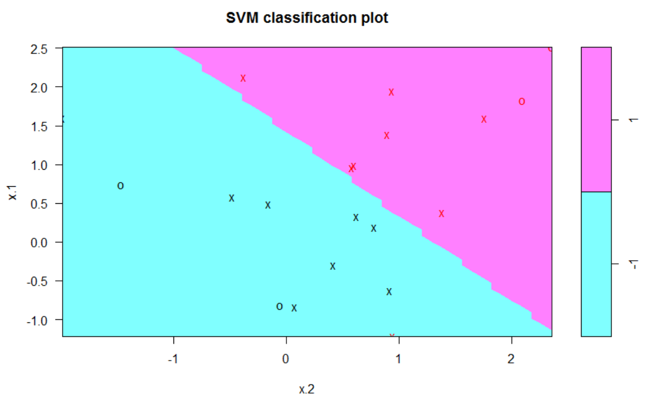
Question 2

I am running the Lab #9 from ISLR, to familiarize with the e1071 library.

Binary Classification

The first part deals with binary classification. One of the important takeaways from this part is how the output changes when we vary the cost coefficient, the first graph considers a SVM with cost value of c=10. The second graph has the same SVM but with cost c=0.1. The main difference is that the first SVM has 7 support vectors. The second SVM has 16. The support vectors are represented as crosses in the scatter plot.





Parameter Selection via Cross Validation

Following the Binary Classification exercise above, I used cross validation to select the best value for the cost parameter c, from a range of values. The function to utilize is called `tune()`, and the output for different values for c is the following:

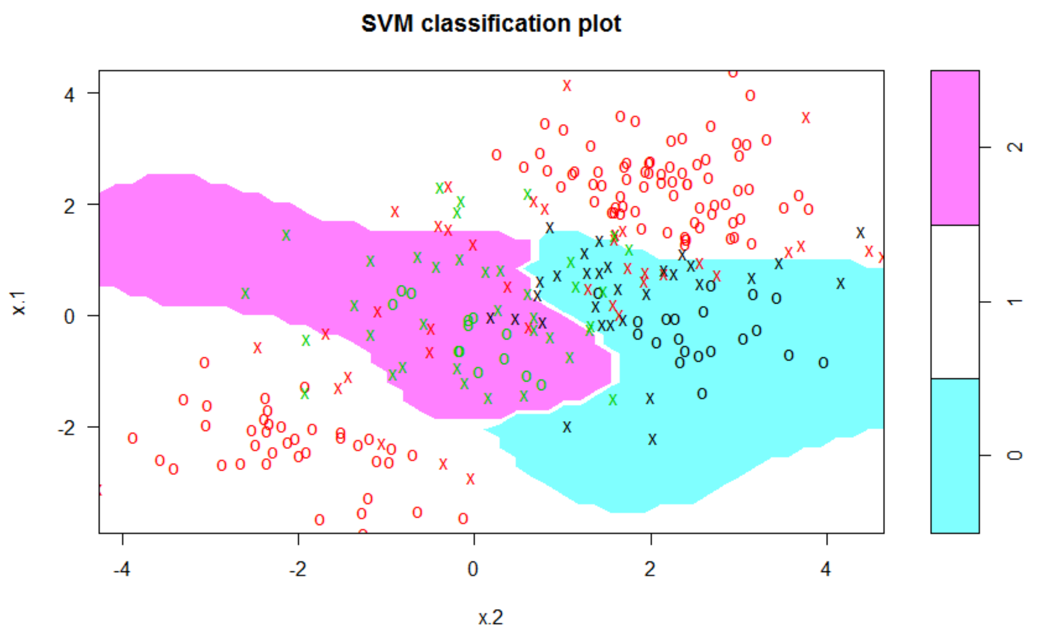
```
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.1

- Detailed performance results:
   cost error dispersion
1 1e-03  0.70  0.4216370
2 1e-02  0.70  0.4216370
3 1e-01  0.10  0.2108185
4 1e+00  0.15  0.2415229
5 5e+00  0.15  0.2415229
6 1e+01  0.15  0.2415229
7 1e+02  0.15  0.2415229
```

For this experiment, the best value for the cost is c=0.1, because it displays the least error.

Multi-class Classification

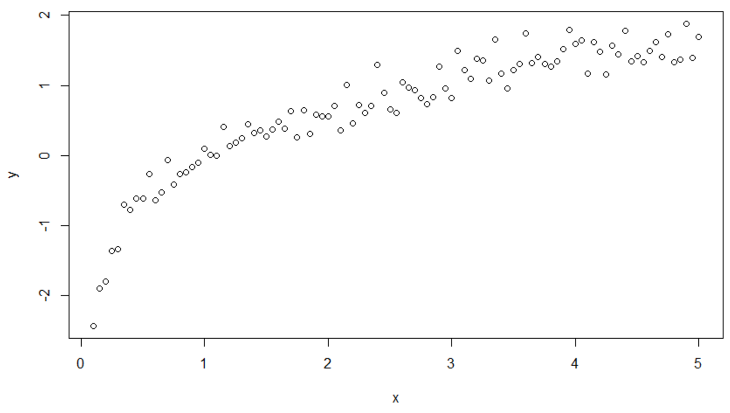
When dealing with multi class classification, this library uses a "one-against-one" approach. If there are k different classes to identify, a total of k(k-1)/2 binary classifiers are trained. The appropriate class is found by a voting scheme.

Following Lab #9 of ISLR, we get the following 3 class model:

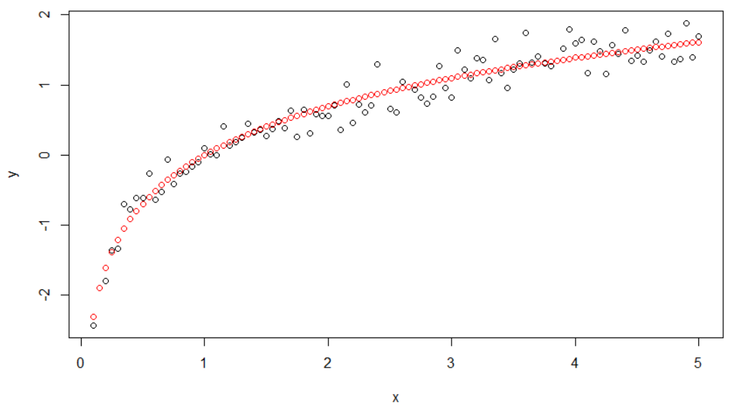


Regression

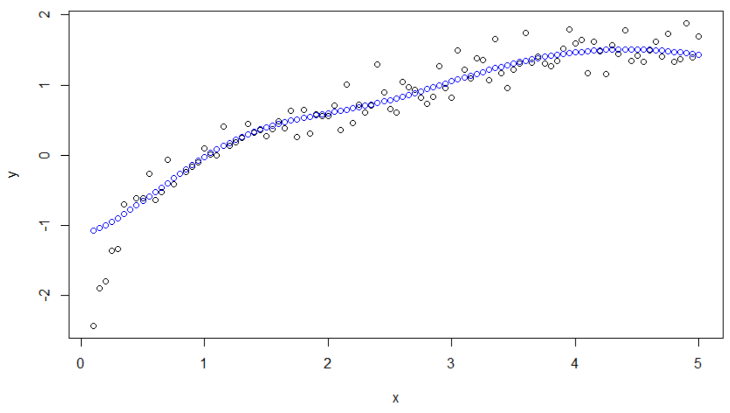
When trying regression, I created random data following a logarithmic relationship, with random noise.



The underlying model is now overlaid over the data points,



When we fit the SVM over the data points, we get the following fitted curve. One of the main shortcomings of this method is that we cannot get an analytical relationship between x and y, that we could interpret.



Categorical Data

When using categorical predictors, we must make sure that the data type of those vectors are factor. We can just feed the `svm()` method with the categorical variables, and the method will internally handle them.

Does it scale the data?

We can choose to scale the data with the parameter using the parameter “scale” in the method svm. According to the R help, we have to pass “a logical vector indicating the variables to be scaled. If scale is of length 1, the value is recycled as many times as needed. Per default, data are scaled internally (both x and y variables) to zero mean and unit variance. The center and scale values are returned and used for later predictions.”

Missing Data

When there is missing data, the `prediction()` function does not work. I need to replace the NA with values, to make the SVM work.

If most data points have one missing component

Question 3

(a) Before moving forward

I removed the column “ID” from the BreastCancer dataset, because it does not convey predictive information.

There are 16 NA values in the column “Bare.nuclei”.

This is a factor column with levels $L = \{1,2,\ldots,10\}$.
I created the new level 11, and replaced all the NA values with level 11.

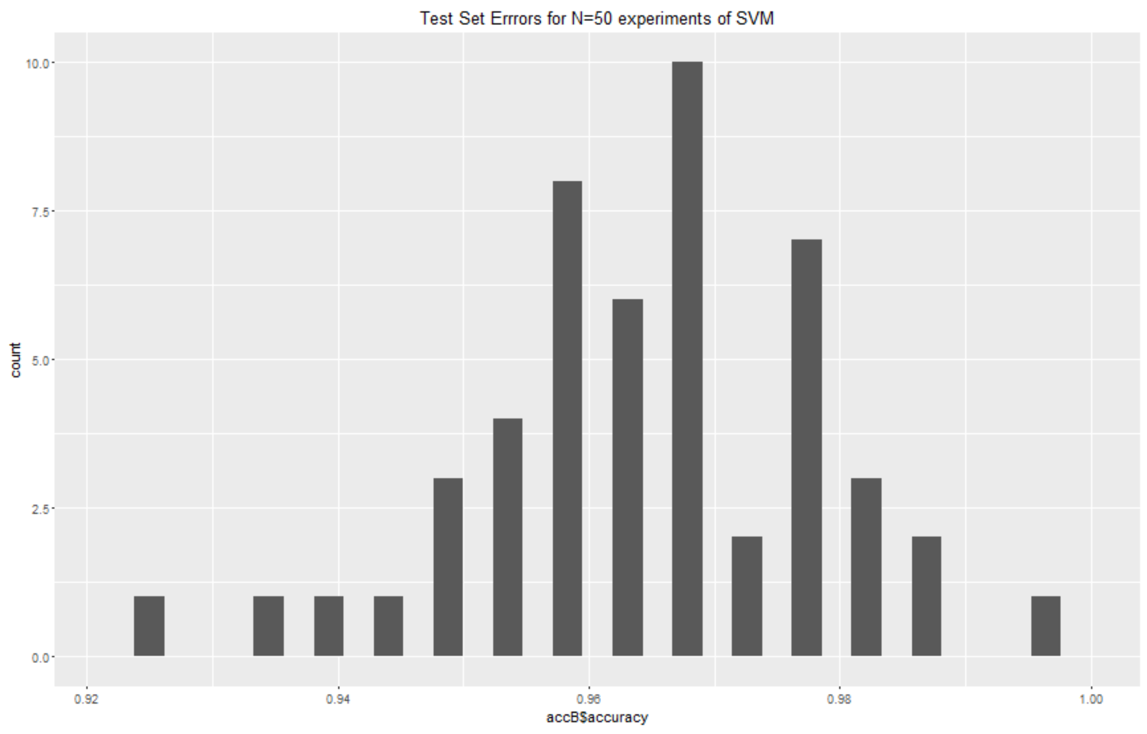
The output for the 10-fold cross validation over 70% of the training set, using a linear kernel, shows that the best cost value is $c = 0.1$, having 0.03% average error:

```
Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
  0.1

- best performance: 0.03061224

- Detailed performance results:
   cost      error dispersion
1 5e-03 0.03469388 0.02894156
2 1e-02 0.03265306 0.02581451
3 5e-02 0.03469388 0.02366330
4 1e-01 0.03061224 0.02204334
5 1e+00 0.03686224 0.02124532
6 5e+00 0.04298469 0.02806155
7 1e+01 0.04298469 0.02636089
8 1e+02 0.04298469 0.02636089
```

The predictor has a 95,7% accuracy over the training data. A serious problem though is that in 5 cases we predicted "benign", when in reality the case was "malignant".

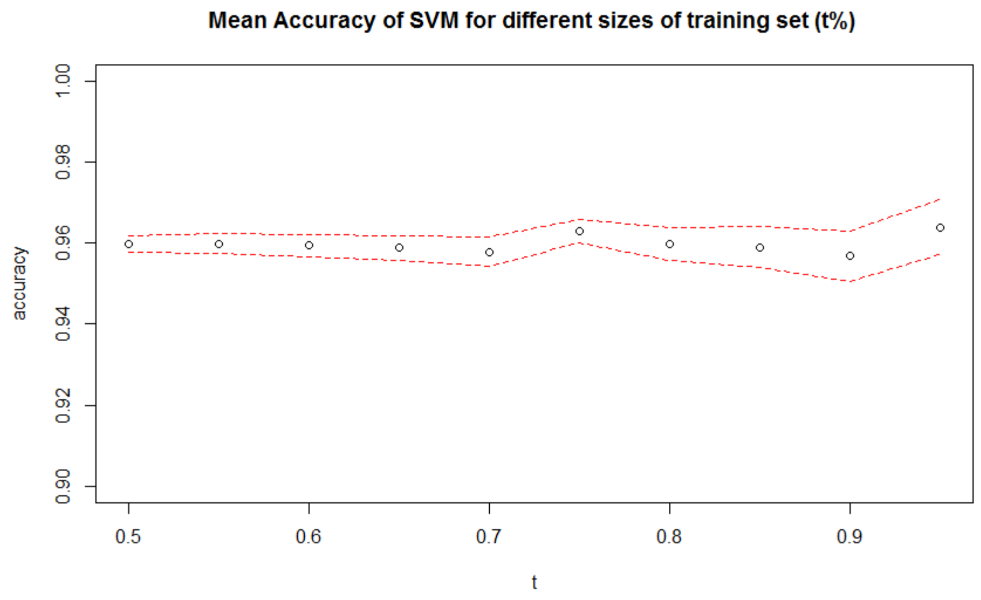
```
Confusion Matrix and Statistics

           Reference
Prediction  benign malignant
  benign       129         5
  malignant      4        72

               Accuracy : 0.9571
                 95% CI : (0.9202, 0.9802)
    No Information Rate : 0.6333
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9075
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9699
            Specificity : 0.9351
         Pos Pred Value : 0.9627
         Neg Pred Value : 0.9474
             Prevalence : 0.6333
         Detection Rate : 0.6143
   Detection Prevalence : 0.6381
      Balanced Accuracy : 0.9525

       'Positive' Class : benign
```

(b) Repeating part (a) a total of N=50 times, analyzing the accuracy, we get average $\mu_{acc} = 0.963$, and standard deviation $\sigma_{acc} = 0.0138$. A histogram of the 50 values is displayed below. We could eventually argue that they are distributed normally, but we would need to increase N to get a more convincing conclusion.



(c) Let the parameter t be the proportion of observations to be considered in the training set. I repeated part (b) for different values of $t \in \{50\%, 55\%, \dots, 95\%\}$.

The following plot displays the average values for accuracy. The points are the average accuracy for each value of t, across N=50 realizations. The red dotted lines represent the confidence interval at a 95% level.



The following facts can be drawn from the graph:

- The mean accuracy is fairly stable around 0.96.
- The mean accuracy is lowest at t=0.90, but not relevantly lower.
- The accuracy is highest when we are at t=0.95.
- The 95% confidence interval is bigger as we increase t, and is biggest at t=0.95

Key take-away,

- Increasing t does not necessarily increase the accuracy of a model.
- Increasing t increases the variance of the accuracy of the model.

Question 4

SVM solves for α in the following model,

$$\max_{\boldsymbol{\alpha} \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j t_i t_j k(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{subject to} \quad \sum_{i=1}^{n} \alpha_i t_i = 0$$

$$\alpha_i \leq C, \quad i = 1, \dots, n$$

We are using a Gaussian kernel, so $k(x_i, x_j) = \exp\left(\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$

Also, t_i, x_i, x_j, C, σ are all data to be fed to the model, and $n = 10{,}000$

On the first place, we did cross validation to tune the best values for (C, σ). Let's call them $(C^*, \sigma^*)_{10000}$, and the optimal tuned value for the SVM is α^*_{10000}, and therefore we also find b^*_{10000} using the relationship,

$$b^* = t_j - \sum_{i=1}^{n} \alpha_i^* t_i k(\mathbf{x}_i, \mathbf{x}_j)$$

When we move to the test set, we classify using the tuned values,

$$\text{sign}\left(\sum_{i=1}^{n} \alpha_i^* t_i k(\mathbf{x}_i, \mathbf{x}) + b^*\right)$$

If we now have a total of 50.000 training points. We are proposed to skip the cross validation, and use the same $(C^*, \sigma^*)_{10000}$ over the 50.000 datapoint set. Skipping the cross validation, we would keep the same $(C^*, \sigma^*)_{10000}$ and calculate α^*_{50000} and b^*_{10000}.

Problem is that in this new case, we are trained C and σ on a subset of the training set, and then training α and b over the whole training set. If the additional points are very different from the original points, the tuned $(C^*, \sigma^*)_{10000}$ are useless in the new expanded dataset. The SVM could be making its best to get a good α^*_{50000} and b^*_{10000} but the results can be very poor because we are setting it with biased parameters.

The only case when this will not have impact is when the additional 40,000 data points are very similar to the original 10,000, which means that the 10,000 data points are a great representation of the variance of the population. The tuned values of C, σ are still a good choice for the 50,000 dataset.

Question 5

(a) As seen in lecture, the non-separable case for SVM solve the following model,

$$\begin{aligned} \min_{\mathbf{w},\boldsymbol{\xi},b} \quad & \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & t_i\left(\mathbf{w}^\top \mathbf{x}_i + b\right) \geq 1 - \xi_i, \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad i = 1, \ldots, n \end{aligned}$$

Where the objective term $C\sum_{i=1}^{n} \xi_i$ is a cost function, to penalize misclassified points.

Instead of using this same cost function, the Support Vector Regression (SVR) minimizes an $\epsilon - insensitive$ error function of the form:

$$E_\epsilon = \begin{cases} 0 & if\ |y(x) - t| < \epsilon \\ |y(x) - t| - \epsilon & otherwise \end{cases}$$

Where $y(x_i) := w^T x_i + b$, and t is the dependent variable.

The advantage of using this function is that only penalizes predictions that are more than ϵ away from the target, resulting in sparse solutions, where only a subset of the training points matter.

The regularized objective function for SVR would take the form

$$C\sum_{i=1}^{N} E_\epsilon\left(\mathbf{w}^\top \mathbf{x}_i + b - t_i\right) \; + \; \frac{1}{2}||\mathbf{w}||^2$$

Now, to take into account the fact that E_ϵ has an absolute value, we use ξ and $\hat{\xi}$ auxiliary slack variables to replace that function. To understand the replacement, we lay down the following model,

$$\begin{aligned}
\min_{\mathbf{w},b,\ \xi\geq 0,\ \hat{\xi}\geq 0} \quad & C\sum_{i=1}^{n}\left(\xi_i + \hat{\xi}_i\right) \; + \; \frac{1}{2}||\mathbf{w}||^2 & & \\
\text{subject to} \quad t_i \; & \leq \; y(\mathbf{x}_i) + \epsilon + \xi_i\,, & i = 1,\ldots,n \quad & (1)\\
t_i \; & \geq \; y(\mathbf{x}_i) - \epsilon - \hat{\xi}_i\,, & i = 1,\ldots,n. \quad & (2)
\end{aligned}$$

If $y(x_i) - t_i \leq -\epsilon$, then $\xi_i > 0$, to make sure equation (1) holds. Since ξ_i is part of the minimizing objective function, we will pick the smallest value $\xi_i = |y(x_i) - t_i| - \epsilon$.

If $y(x_i) - t_i \geq \epsilon$, then $\hat{\xi}_i > 0$, to make sure equation (2) holds. Since $\hat{\xi}_i$ is part of the minimizing objective function, we will pick the smallest value $\hat{\xi}_i = |y(x_i) - t_i| - \epsilon$.

The model requires 2 input parameters, which can be tuned with cross-validation:

- C represents the cost of error
- ϵ Represents how wide the insensitivity tube of the SVR is.

(b) Introducing the Lagrangian multipliers on the constraints, we optimize the Lagrangian function:

$$\begin{aligned}
&\mathbb{L}(w, b, \xi, \hat{\xi}, \alpha, a, \hat{a}, \mu, \hat{\mu}) \\
&\quad = C\sum_{i=1}^{n}(\xi_i + \hat{\xi}_i) + \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} a_i(t_i - w^T x_i - b - \epsilon - \xi_i) + \sum_{i=1}^{n} \hat{a}_i(-t_i + w^T x_i + b - \epsilon - \hat{\xi}_i) \\
&\quad + \sum_{i=1}^{n} \mu_i \xi_i + \sum_{i=1}^{n} \hat{\mu}_i \hat{\xi}_i
\end{aligned}$$

Optimizing the Lagrangean,

$$\begin{aligned}
\frac{\delta \mathbb{L}}{\delta w_j} &= w_j - a_j \sum_{i=1}^{n} x_{ji} + \hat{a}_j \sum_{i=1}^{n} x_{ji} \;\; \forall j \in \{1, \ldots, p\} \\
\frac{\delta \mathbb{L}}{\delta b} &= -\sum_{i=1}^{n} a_i + \sum_{i=1}^{n} \hat{a}_i \\
\frac{\delta \mathbb{L}}{\delta \xi_i} &= a_i + \mu_i \; \forall i \in \{1, .., n\} \\
\frac{d\mathbb{L}}{\delta \hat{\xi}_i} &= \hat{a}_i + \hat{\mu}_i \; \forall i \in \{1, .., n\}
\end{aligned}$$

Setting all these expressions equal to zero, we get,

$$w_j = \left(a_j - \widehat{a_j}\right) \sum_{i=1}^{N} x_{ji} \ \ \forall j \in \{1, \dots, p\}$$

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} \widehat{a_\iota}$$

$$a_i = -\mu_i \ \forall i \in \{1, \dots, n\}$$

$$\widehat{a_\iota} = -\widehat{\mu_\iota} \ \forall i \in \{1, \dots, n\}$$

This in addition to

Replacing these relationships to get the optimized Lagrangean,

$$\mathbb{L}\left(w, b, \xi, \hat{\xi}, \alpha, a, \hat{a}, \mu, \hat{\mu}\ \right) = C \sum_{i=1}^{n} \left(\xi_i + \widehat{\xi_\iota}\right) + \frac{1}{2} \sum_{j=1}^{p} \left(a_j - \widehat{a_j}\right)^2 \left(\sum_{i=1}^{N} x_{ji}\right)^2 - \sum_{i=1}^{n} a_i \xi_i - \sum_{i=1}^{n} \widehat{a_\iota} \widehat{\xi_\iota}$$

$$\mathbb{L}\left(w, b, \xi, \hat{\xi}, \alpha, a, \hat{a}, \mu, \hat{\mu}\ \right) = \sum_{i=1}^{n} \left(\xi_i (C - a_i) + \widehat{\xi_\iota}(C - \widehat{a_\iota})\right) + \frac{1}{2} \sum_{j=1}^{p} \left(a_j - \widehat{a_j}\right)^2 \left(\sum_{i=1}^{N} x_{ji}\right)^2$$

I have to replace Xi using the primal’s constraints with equality, and simplify.

Appendix

Code for Question 2

```
# Chapter 9 Lab: Support Vector Machines

# Support Vector Classifier

set.seed(1)
x=matrix(rnorm(20*2), ncol=2)
y=c(rep(-1,10), rep(1,10))
x[y==1,]=x[y==1,] + 1
plot(x, col=(3-y))
dat=data.frame(x=x, y=as.factor(y))
library(e1071)
svmfit=svm(y~., data=dat, kernel="linear", cost=10,scale=FALSE)
plot(svmfit, dat)
svmfit$index
summary(svmfit)
svmfit=svm(y~., data=dat, kernel="linear", cost=0.1,scale=FALSE)
plot(svmfit, dat)
svmfit$index
set.seed(1)
tune.out=tune(svm,y~.,data=dat,kernel="linear",ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))
summary(tune.out)
bestmod=tune.out$best.model
summary(bestmod)
xtest=matrix(rnorm(20*2), ncol=2)
ytest=sample(c(-1,1), 20, rep=TRUE)
xtest[ytest==1,]=xtest[ytest==1,] + 1
testdat=data.frame(x=xtest, y=as.factor(ytest))
ypred=predict(bestmod,testdat)
table(predict=ypred, truth=testdat$y)
svmfit=svm(y~., data=dat, kernel="linear", cost=.01,scale=FALSE)
ypred=predict(svmfit,testdat)
table(predict=ypred, truth=testdat$y)
x[y==1,]=x[y==1,]+0.5
plot(x, col=(y+5)/2, pch=19)
dat=data.frame(x=x,y=as.factor(y))
svmfit=svm(y~., data=dat, kernel="linear", cost=1e5)
summary(svmfit)
plot(svmfit, dat)
svmfit=svm(y~., data=dat, kernel="linear", cost=1)
summary(svmfit)
plot(svmfit,dat)

# Support Vector Machine

set.seed(1)
x=matrix(rnorm(200*2), ncol=2)
x[1:100,]=x[1:100,]+2
x[101:150,]=x[101:150,]-2
y=c(rep(1,150),rep(2,50))
dat=data.frame(x=x,y=as.factor(y))
plot(x, col=y)
train=sample(200,100)
svmfit=svm(y~., data=dat[train,], kernel="radial",  gamma=1, cost=1)
plot(svmfit, dat[train,])
summary(svmfit)
svmfit=svm(y~., data=dat[train,], kernel="radial",gamma=1,cost=1e5)
plot(svmfit,dat[train,])
set.seed(1)
tune.out=tune(svm, y~., data=dat[train,], kernel="radial",
ranges=list(cost=c(0.1,1,10,100,1000),gamma=c(0.5,1,2,3,4)))
summary(tune.out)
table(true=dat[-train,"y"], pred=predict(tune.out$best.model,newdata=dat[-train,]))

svmfit=svm(y~., data=dat[train,], kernel="radial",gamma=2,cost=1)
plot(svmfit,dat[train,])

##################
# SVM with Multiple Classes

set.seed(1)
x=rbind(x, matrix(rnorm(50*2), ncol=2))
y=c(y, rep(0,50))
x[y==0,2]=x[y==0,2]+2
```

```
dat=data.frame(x=x, y=as.factor(y))
par(mfrow=c(1,1))
plot(x,col=(y+1))
svmfit=svm(y~., data=dat, kernel="radial", cost=10, gamma=1)
plot(svmfit, dat)

###############
## Regression mode on two dimensions
# create data
x <- seq(0.1, 5, by = 0.05)
y <- log(x) + rnorm(x, sd = 0.2)
# estimate model and predict input values
m   <- svm(x, y)
new <- predict(m, x)
# visualize
plot(x, y)
points(x, log(x), col = 2)
points(x, new, col = 4)

############
## Dealing with Categorical Data

x1<-rnorm(500)
x2<-rnorm(500)

#Categorical Predictor 1, with 5 levels
x3<-as.factor(rep(c(1,2,3,4,5),c(50,150,130,70,100)))

#Catgegorical Predictor 2, with 3 levels
x4<-as.factor(rep(c("R","B","G"),c(100,200,200)))

#Response
y<-rep(c(-1,1),c(275,225))
class<-as.factor(y)
svmdata<-cbind(class,x1,x2,x3,x4)

mod1<-svm(class~.,data=svmdata,type="C-classification")

ypred=predict(mod1,svmdata)
table(predict=ypred, truth=class)
```