Chuong Vu

COMP 5800

Homework 6

Prof. Byung Kim

Quadratic Programming for linear SVM

As the question requires, I only use 10 measurements from each iris-versicolor and iris-virginica. I begin with Iris-virginica from R language program to find the quadratic coefficient, weight vector, w and the bias b.

1. **10 MEASUREMENTS FROM IRIS-SETOSA AND IRIS-VERICOLOR.**

Let load the data into R. I work on versicolor for linearly first and then virginica.

#Set my working directory

**> setwd("D:/ChuongStuff/Bio/HW6")**

#Check the directory

**> getwd()**

[1] "D:/ChuongStuff/Bio/HW6"

# Read and Store data

**> iris.set.vir = read.table("Iris-set.ver.txt")**

#Loading library

**> if(!require(quadprog)) install.packages("quadprog"); library(quadprog)**

**> if(!require(kernlab)) install.packages("kernlab"); library(kernlab)**

**> iris.set.vir**

Sepal.Length Sepal.Width Petal.Length Petal.Width Species

1 5.1 3.5 1.4 0.2 setosa

2 4.9 3.0 1.4 0.2 setosa

3 4.7 3.2 1.3 0.2 setosa

4 4.6 3.1 1.5 0.2 setosa

5 5.0 3.6 1.4 0.2 setosa

6 5.4 3.9 1.7 0.4 setosa

7 4.6 3.4 1.4 0.3 setosa

8 5.0 3.4 1.5 0.2 setosa

9 4.4 2.9 1.4 0.2 setosa

10 4.9 3.1 1.5 0.1 setosa

11 7.0 3.2 4.7 1.4 versicolor

12 6.4 3.2 4.5 1.5 versicolor

13 6.9 3.1 4.9 1.5 versicolor

14 5.5 2.3 4.0 1.3 versicolor

15 6.5 2.8 4.6 1.5 versicolor

16 5.7 2.8 4.5 1.3 versicolor

17 6.3 3.3 4.7 1.6 versicolor

18 4.9 2.4 3.3 1.0 versicolor

19 6.6 2.9 4.6 1.3 versicolor

20 5.2 2.7 3.9 1.4 versicolor

Next, add a Y column and set value to 1

**> train <- iris.set.vir**

**> train$y <- 1**

**> train**

Sepal.Length Sepal.Width Petal.Length Petal.Width Species y

1 5.1 3.5 1.4 0.2 setosa 1

2 4.9 3.0 1.4 0.2 setosa 1

3 4.7 3.2 1.3 0.2 setosa 1

4 4.6 3.1 1.5 0.2 setosa 1

5 5.0 3.6 1.4 0.2 setosa 1

6 5.4 3.9 1.7 0.4 setosa 1

7 4.6 3.4 1.4 0.3 setosa 1

8 5.0 3.4 1.5 0.2 setosa 1

9 4.4 2.9 1.4 0.2 setosa 1

10 4.9 3.1 1.5 0.1 setosa 1

11 7.0 3.2 4.7 1.4 versicolor -1

12 6.4 3.2 4.5 1.5 versicolor -1

13 6.9 3.1 4.9 1.5 versicolor -1

14 5.5 2.3 4.0 1.3 versicolor -1

15 6.5 2.8 4.6 1.5 versicolor -1

16 5.7 2.8 4.5 1.3 versicolor -1

17 6.3 3.3 4.7 1.6 versicolor -1

18 4.9 2.4 3.3 1.0 versicolor -1

19 6.6 2.9 4.6 1.3 versicolor -1

20 5.2 2.7 3.9 1.4 versicolor -1

And then we order the training data labeling to avoid oddities

**> train <- train[order(train$y, decreasing=TRUE),]**

Set the problem data and parameters, simple linear regression

**> X <- as.matrix(train[,c("Petal.Length", "Petal.Width")])**

**> X**

Petal.Length Petal.Width

1 1.4 0.2

2 1.4 0.2

3 1.3 0.2

4 1.5 0.2

5 1.4 0.2

6 1.7 0.4

7 1.4 0.3

8 1.5 0.2

9 1.4 0.2

10 1.5 0.1

11 4.7 1.4

12 4.5 1.5

13 4.9 1.5

14 4.0 1.3

15 4.6 1.5

16 4.5 1.3

17 4.7 1.6

18 3.3 1.0

19 4.6 1.3

20 3.9 1.4

**> y <- as.matrix(train$y)**

**> y**

[,1]

[1,] 1

[2,] 1

[3,] 1

[4,] 1

[5,] 1

[6,] 1

[7,] 1

[8,] 1

[9,] 1

[10,] 1

[11,] -1

[12,] -1

[13,] -1

[14,] -1

[15,] -1

[16,] -1

[17,] -1

[18,] -1

[19,] -1

[20,] -1

**> n <- dim(X)[1]**

**> n**

[1] 20

Solve Quadratic Programming and the perturbance hack. The QP requires that the D matrix be symmetric positive definite, but the SVM problem is almost always.

As a hack, we can perturb D by a small diagonal matrix and obtain positive definite matrix. Choose eps a relatively small value for the diagonal perturbance.

**> eps <- 5e-4**

**> eps**

[1] 5e-04

# build the system matrices

**> Q <- sapply(1:n, function(i) y[i]\*t(X)[,i])**

**> Q**

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16]

Petal.Length 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 -4.7 -4.5 -4.9 -4.0 -4.6 -4.5

Petal.Width 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 -1.4 -1.5 -1.5 -1.3 -1.5 -1.3

[,17] [,18] [,19] [,20]

Petal.Length -4.7 -3.3 -4.6 -3.9

Petal.Width -1.6 -1.0 -1.3 -1.4

**> D <- t(Q)%\*%Q**

**> D**

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15]

[1,] 2.00 2.00 1.86 2.14 2.00 2.46 2.02 2.14 2.00 2.12 -6.86 -6.60 -7.16 -5.86 -6.74

[2,] 2.00 2.00 1.86 2.14 2.00 2.46 2.02 2.14 2.00 2.12 -6.86 -6.60 -7.16 -5.86 -6.74

[3,] 1.86 1.86 1.73 1.99 1.86 2.29 1.88 1.99 1.86 1.97 -6.39 -6.15 -6.67 -5.46 -6.28

[4,] 2.14 2.14 1.99 2.29 2.14 2.63 2.16 2.29 2.14 2.27 -7.33 -7.05 -7.65 -6.26 -7.20

[5,] 2.00 2.00 1.86 2.14 2.00 2.46 2.02 2.14 2.00 2.12 -6.86 -6.60 -7.16 -5.86 -6.74

[6,] 2.46 2.46 2.29 2.63 2.46 3.05 2.50 2.63 2.46 2.59 -8.55 -8.25 -8.93 -7.32 -8.42

[7,] 2.02 2.02 1.88 2.16 2.02 2.50 2.05 2.16 2.02 2.13 -7.00 -6.75 -7.31 -5.99 -6.89

[8,] 2.14 2.14 1.99 2.29 2.14 2.63 2.16 2.29 2.14 2.27 -7.33 -7.05 -7.65 -6.26 -7.20

[9,] 2.00 2.00 1.86 2.14 2.00 2.46 2.02 2.14 2.00 2.12 -6.86 -6.60 -7.16 -5.86 -6.74

[10,] 2.12 2.12 1.97 2.27 2.12 2.59 2.13 2.27 2.12 2.26 -7.19 -6.90 -7.50 -6.13 -7.05

[11,] -6.86 -6.86 -6.39 -7.33 -6.86 -8.55 -7.00 -7.33 -6.86 -7.19 24.05 23.25 25.13 20.62 23.72

[12,] -6.60 -6.60 -6.15 -7.05 -6.60 -8.25 -6.75 -7.05 -6.60 -6.90 23.25 22.50 24.30 19.95 22.95

[13,] -7.16 -7.16 -6.67 -7.65 -7.16 -8.93 -7.31 -7.65 -7.16 -7.50 25.13 24.30 26.26 21.55 24.79

[14,] -5.86 -5.86 -5.46 -6.26 -5.86 -7.32 -5.99 -6.26 -5.86 -6.13 20.62 19.95 21.55 17.69 20.35

[15,] -6.74 -6.74 -6.28 -7.20 -6.74 -8.42 -6.89 -7.20 -6.74 -7.05 23.72 22.95 24.79 20.35 23.41

[16,] -6.56 -6.56 -6.11 -7.01 -6.56 -8.17 -6.69 -7.01 -6.56 -6.88 22.97 22.20 24.00 19.69 22.65

[17,] -6.90 -6.90 -6.43 -7.37 -6.90 -8.63 -7.06 -7.37 -6.90 -7.21 24.33 23.55 25.43 20.88 24.02

[18,] -4.82 -4.82 -4.49 -5.15 -4.82 -6.01 -4.92 -5.15 -4.82 -5.05 16.91 16.35 17.67 14.50 16.68

[19,] -6.70 -6.70 -6.24 -7.16 -6.70 -8.34 -6.83 -7.16 -6.70 -7.03 23.44 22.65 24.49 20.09 23.11

[20,] -5.74 -5.74 -5.35 -6.13 -5.74 -7.19 -5.88 -6.13 -5.74 -5.99 20.29 19.65 21.21 17.42 20.04

[,16] [,17] [,18] [,19] [,20]

[1,] -6.56 -6.90 -4.82 -6.70 -5.74

[2,] -6.56 -6.90 -4.82 -6.70 -5.74

[3,] -6.11 -6.43 -4.49 -6.24 -5.35

[4,] -7.01 -7.37 -5.15 -7.16 -6.13

[5,] -6.56 -6.90 -4.82 -6.70 -5.74

[6,] -8.17 -8.63 -6.01 -8.34 -7.19

[7,] -6.69 -7.06 -4.92 -6.83 -5.88

[8,] -7.01 -7.37 -5.15 -7.16 -6.13

[9,] -6.56 -6.90 -4.82 -6.70 -5.74

[10,] -6.88 -7.21 -5.05 -7.03 -5.99

[11,] 22.97 24.33 16.91 23.44 20.29

[12,] 22.20 23.55 16.35 22.65 19.65

[13,] 24.00 25.43 17.67 24.49 21.21

[14,] 19.69 20.88 14.50 20.09 17.42

[15,] 22.65 24.02 16.68 23.11 20.04

[16,] 21.94 23.23 16.15 22.39 19.37

[17,] 23.23 24.65 17.11 23.70 20.57

[18,] 16.15 17.11 11.89 16.48 14.27

[19,] 22.39 23.70 16.48 22.85 19.76

[20,] 19.37 20.57 14.27 19.76 17.17

**> d <- matrix(1, nrow=n)**

**> d**

[,1]

[1,] 1

[2,] 1

[3,] 1

[4,] 1

[5,] 1

[6,] 1

[7,] 1

[8,] 1

[9,] 1

[10,] 1

[11,] 1

[12,] 1

[13,] 1

[14,] 1

[15,] 1

[16,] 1

[17,] 1

[18,] 1

[19,] 1

[20,] 1

**> b0 <- rbind(matrix(0, nrow=1, ncol=1), matrix(0, nrow=n, ncol=1))**

**> b0**

[,1]

[1,] 0

[2,] 0

[3,] 0

[4,] 0

[5,] 0

[6,] 0

[7,] 0

[8,] 0

[9,] 0

[10,] 0

[11,] 0

[12,] 0

[13,] 0

[14,] 0

[15,] 0

[16,] 0

[17,] 0

[18,] 0

[19,] 0

[20,] 0

[21,] 0

**> A <- t(rbind(matrix(y, nrow=1, ncol=n), diag(nrow=n)))**

**> A**

[,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21]

[1,] 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[2,] 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[3,] 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[4,] 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[5,] 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[6,] 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

[7,] 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

[8,] 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

[9,] 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

[10,] 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

[11,] -1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

[12,] -1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

[13,] -1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

[14,] -1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

[15,] -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

[16,] -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

[17,] -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

[18,] -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

[19,] -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

[20,] -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

# call the QP solver:

**> sol <- solve.QP(D +eps\*diag(n), d, A, b0, factorized=FALSE)**

**> sol**

tion

[1] -1.416748e-09 -4.448134e-10 -7.197692e-10 -1.391260e-09 7.186176e-14 2.081445e+02 0.000000e+00 -3.378839e-12 1.966469e-13 1.868617e-13

[11] -1.776079e-13 -3.744581e-14 -8.389938e-10 5.959516e-10 -1.897283e-09 4.036769e-17 -3.496459e-14 1.500599e-14 7.601255e+01 0.000000e+00

$value

[1] -142.0785

$unconstrained.solution

[1] 453.98934 453.98934 690.15822 217.82046 453.98934 1505.83639 1334.16619 217.82046 453.98934 -662.35639 777.46161 -575.05300

[13] 369.62253 4.45628 -338.88412 1185.30069 -982.89209 991.80465 1421.46958 -1111.88945

$iterations

[1] 19 0

$Lagrangian

[1] 0.0000000 1.7512800 1.7512800 1.3324953 2.1700648 1.7512800 0.0000000 0.1954268 2.1700648 1.7512800 3.7259180 1.0990622 3.4924850 1.8173460

[15] 2.4747022 3.0737002 0.3807785 4.2107688 0.7386357 0.0000000 4.4493403

$iact

[1] 21 18 13 11 16 15 14 9 5 12 6 3 10 2 4 19 17 8

**> qpsol <- matrix(sol$solution, nrow=n)**

**> qpsol**

[,1]

[1,] -1.416748e-09

[2,] -4.448134e-10

[3,] -7.197692e-10

[4,] -1.391260e-09

[5,] 7.186176e-14

[6,] 2.081445e+02

[7,] 0.000000e+00

[8,] -3.378839e-12

[9,] 1.966469e-13

[10,] 1.868617e-13

[11,] -1.776079e-13

[12,] -3.744581e-14

[13,] -8.389938e-10

[14,] 5.959516e-10

[15,] -1.897283e-09

[16,] 4.036769e-17

[17,] -3.496459e-14

[18,] 1.500599e-14

[19,] 7.601255e+01

[20,] 0.000000e+00

Now, by using svm, with dataset train we got above with linear, we get.

**> C <- 1e5** # Huge value forces hard margin problem

**> sv <- svm(y~Petal.Length+Petal.Width, data=train, kernel="linear", scale=FALSE, type="C-classification", cost=C)**

**> sv**

Call:

svm(formula = Species ~ Petal.Length + Petal.Width, data = train, kernel = "linear", type = "C-classification", cost = C,

scale = FALSE)

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 1e+05

gamma: 0.5

Number of Support Vectors: 2

Now we get the slope and intercept

**> W <- rowSums(sapply(1:length(sv$coefs), function(i) sv$coefs[i]\*sv$SV[i,]))**

**> svmline = c(sv$rho/W[2], -W[1]/W[2])**

**> W**

Petal.Length Petal.Width

-1.0958905 -0.4109589

**> svmline**

Petal.Width Petal.Length

7.366667 -2.666667

Now extract and plot the decision boundary from the results of each solver.

# build the support vectors, slopes, and intercepts

> # build the support vectors, slopes, and intercepts

**> findLine <- function(a, y, X){**

**+ nonzero <- abs(a) > 1e-5**

**+ W <- rowSums(sapply(which(nonzero), function(i) a[i]\*y[i]\*X[i,]))**

**+ b <- mean(sapply(which(nonzero), function(i) X[i,]%\*%W- y[i]))**

**+ slope <- -W[1]/W[2]**

**+ intercept <- b/W[2]**

**+ return(c(intercept,slope))**

**+ }**

**> qpline <- findLine( qpsol, y, X)**

**> qpline**

Petal.Width Petal.Length

0.002123143 0.269167256

Now, plot the result

**> library(ggplot2)**

**> plt <- ggplot(train, aes(x=Petal.Length, y=Petal.Width)) +**

**+ ggtitle("Solving the SVM QP for Setosa and Vericolor") +**

**+ geom\_point(aes(fill=factor(y)), size=3, pch=21) +**

**+ geom\_abline(intercept=qpline[1], slope=qpline[2], size=1, aes(color="quadprog"), show.legend=TRUE) +**

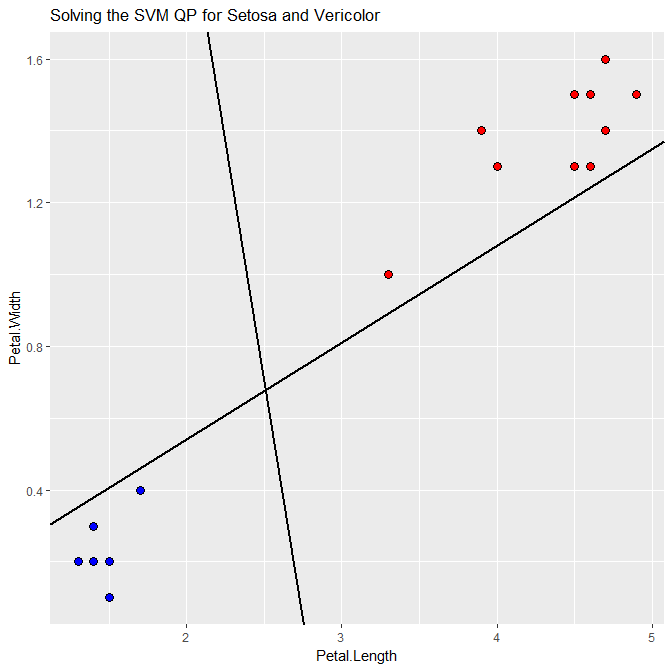
**+ geom\_abline(intercept=svmline[1], slope=svmline[2], size=1, aes(color="svm"))+**

**+ scale\_fill\_manual(values=c("red","blue"), guide='none')+**

**+ scale\_color\_manual(values=c("green", "yellow", "black"))+**

**+ theme(legend.position="bottom", legend.title=element\_blank())**

**> print(plt)**



And prints out all the results

**> print(sprintf("quadprog number of nonzeros: %d", sum(abs(qpsol)>1e-7)))**

[1] "quadprog number of nonzeros: 2"

**> print(sprintf("svm number of nonzeros: %d", length(sv$coefs)))**

[1] "svm number of nonzeros: 2"

**> print(sprintf("Quadprog: Intercept: %f Slope: %f", qpline[1], qpline[2]))**

[1] "Quadprog: Intercept: 0.002123 Slope: 0.269167"

**> print(sprintf("svm: Intercept: %f Slope: %f", svmline[1], svmline[2]))**

[1] "svm: Intercept: 7.366667 Slope: -2.666667"

1. **10 MEASUREMENTS FROM IRIS-VERICOLOR AND IRIS VIRGINICA.**

Similar with setosa and versicolor

**> sol**

$solution

[1] 1.426450e+03 2.166938e+03 1.583451e+03 1.998808e+03 2.021067e+03

[6] 1.269449e+03 2.323939e+03 1.673677e+03 1.123578e+03 2.593425e+03

[11] 2.366427e-14 9.133140e+02 1.182799e+03 2.091417e+03 5.881827e+02

[16] 2.203901e+03 9.355724e+02 3.112520e+03 2.383161e+03 0.000000e+00

$value

[1] -15795.83

$unconstrained.solution

[1] 1459.97846 2136.30317 1602.51291 1984.68185 2002.85560 1317.44402

[7] 2278.83761 1690.52608 1183.99645 2527.55899 -228.88544 1026.66392

[13] 1275.38530 2103.33133 732.50816 2209.51826 1044.83768 3037.46429

[19] 2370.22646 -95.43787

$iterations

[1] 3 0

$Lagrangian

[1] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000

[8] 0.0000000 0.0000000 0.0000000 0.0000000 0.2331537 0.0000000 0.0000000

[15] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.1602178

$iact

[1] 12 21

**> qpsol**

[,1]

[1,] 1.426450e+03

[2,] 2.166938e+03

[3,] 1.583451e+03

[4,] 1.998808e+03

[5,] 2.021067e+03

[6,] 1.269449e+03

[7,] 2.323939e+03

[8,] 1.673677e+03

[9,] 1.123578e+03

[10,] 2.593425e+03

[11,] 2.366427e-14

[12,] 9.133140e+02

[13,] 1.182799e+03

[14,] 2.091417e+03

[15,] 5.881827e+02

[16,] 2.203901e+03

[17,] 9.355724e+02

[18,] 3.112520e+03

[19,] 2.383161e+03

[20,] 0.000000e+00

**> sv**

Call:

svm(formula = Species ~ Petal.Length + Petal.Width, data = train, kernel = "linear",

type = "C-classification", cost = C, scale = FALSE)

Parameters:

SVM-Type: C-classification

SVM-Kernel: linear

cost: 1e+05

gamma: 0.5

Number of Support Vectors: 3

The slope and intercept

**> W**

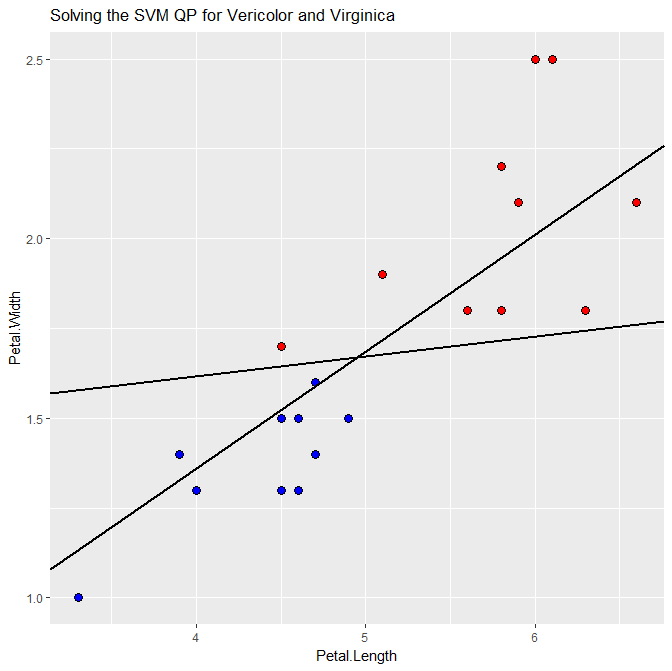
Petal.Length Petal.Width

0.9993763 -17.9975228

**> svmline**

Petal.Width Petal.Length

1.39457581 0.05552855



The results

**> print(sprintf("quadprog number of nonzeros: %d", sum(abs(qpsol)>1e-7)))**

[1] "quadprog number of nonzeros: 18"

**> print(sprintf("svm number of nonzeros: %d", length(sv$coefs)))**

[1] "svm number of nonzeros: 3"

**> print(sprintf("Quadprog: Intercept: %f Slope: %f", qpline[1], qpline[2]))**

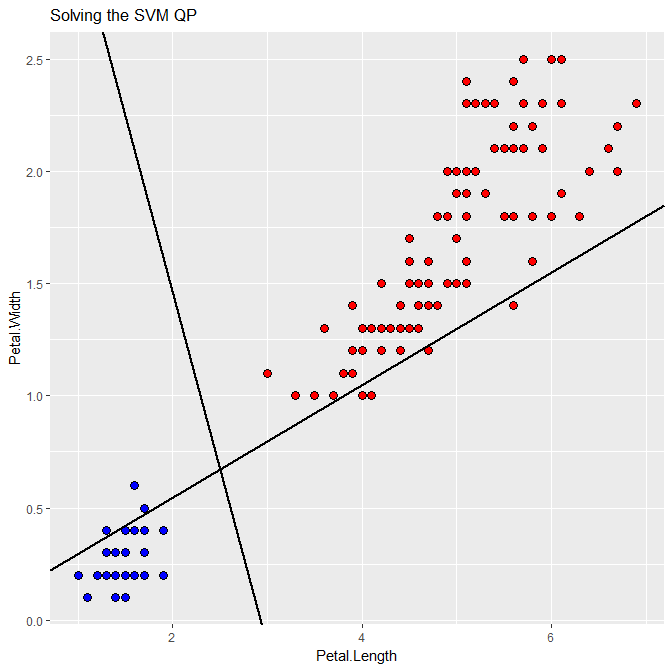
[1] "Quadprog: Intercept: 0.059053 Slope: 0.325066"

**> print(sprintf("svm: Intercept: %f Slope: %f", svmline[1], svmline[2]))**

[1] "svm: Intercept: 1.394576 Slope: 0.055529"

1. **FULL SIZE IRIS DATA**

Result output:



> # print the results

**> print(sprintf("quadprog number of nonzeros: %d", sum(abs(qpsol)>1e-7)))**

[1] "quadprog number of nonzeros: 46"

**> print(sprintf("svm number of nonzeros: %d", length(sv$coefs)))**

[1] "svm number of nonzeros: 2"

**> print(sprintf("Quadprog: Intercept: %f Slope: %f", qpline[1], qpline[2]))**

[1] "Quadprog: Intercept: 0.043690 Slope: 0.250772"

**> print(sprintf("svm: Intercept: %f Slope: %f", svmline[1], svmline[2]))**

[1] "svm: Intercept: 4.600000 Slope: -1.571429"

Briefly discuss: As we can see from 3 pictures, the result from setosa and versicolor are close to the iris data than the versicolor and virginica. The plot in the result 2 are close to each other while it should be far separate from the line.

1. **SOURCE CODE**
2. # use to remove all objects
3. rm**(**list**=**ls**(**all**=**TRUE**))**
4. #Set my working directory
5. setwd**(**"D:/ChuongStuff/Bio/HW6"**)**
6. #Check the directory
7. getwd**()**
8. ##########################################
9. # Perform setosa and vericolor
10. # Read and Store data
11. iris**.**set**.**vir **=** **read.**table**(**"Iris-set.ver.txt"**)**
12. #Loading library
13. library**(**quadprog**)**
14. library**(**e1071**)**
15. train <- iris.set.vir
16. train$y <-ifelse(train[,5]=="setosa", 1, -1)
17. # order the training data labeling to avoid oddities with
18. # see http://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html#f430
19. train <- train[order(train$y, decreasing=TRUE),]
20. # set the problem data and parameters
21. X <- as.matrix(train[,c("Petal.Length", "Petal.Width")])
22. y <- as.matrix(train$y)
23. n <- dim(X)[1]
24. eps <- 5e-4
25. # build the system matrices
26. Q <- sapply(1:n, function(i) y[i]\*t(X)[,i])
27. D <- t(Q)%\*%Q
28. d <- matrix(1, nrow=n)
29. b0 <- rbind( matrix(0, nrow=1, ncol=1) , matrix(0, nrow=n, ncol=1))
30. A <- t(rbind(matrix(y, nrow=1, ncol=n), diag(nrow=n)))
31. # call the QP solver:
32. sol <- solve.QP(D +eps\*diag(n), d, A, b0, meq=0, factorized=FALSE)
33. qpsol <- matrix(sol$solution, nrow=n)
34. ####################################################################################################################
35. # This solver is implemented to solve a soft-margin problem. If we choose C very large,
36. # then the margins will be very small and we'll approach the hard-margin classification problem that we solved above
37. ####################################################################################################################
38. C <- 1e5 # Huge value forces hard margin problem
39. sv <- svm(Species~Petal.Length+Petal.Width, data=train, kernel="linear", scale=FALSE, type="C-classification", cost=C)
40. # Now we get the slope and intercept
41. W <- rowSums(sapply(1:length(sv$coefs), function(i) sv$coefs[i]\*sv$SV[i,]))
42. svmline **=** c**(**sv$rho/W**[2],** **-**W**[1]/**W**[2])**
43. # extract and plot the decision boundary from the results of each solver.
44. # build the support vectors, slopes, and intercepts
45. findLine <- function(a, y, X){
46. nonzero <- abs(a) > **1e-5**
47. W <- rowSums(sapply(which(nonzero), function(i) a[i]\*y[i]\*X[i,]))
48. b <- mean(sapply(which(nonzero), function(i) X[i,]%\*%W- y[i]))
49. slope <- -W[1]/W[2]
50. intercept <- b/W[2]
51. **return(**c**(**intercept**,**slope**))**
52. **}**
53. qpline <- findLine( qpsol, y, X)
54. # plot the results
55. library**(**ggplot2**)**
56. plt <- ggplot(train, aes(x=Petal.Length, y=Petal.Width)) +
57. ggtitle**(**"Solving the SVM QP for Setosa and Vericolor"**)** **+**
58. geom\_point**(**aes**(**fill**=**factor**(**y**)),** size**=3,** pch**=21)** **+**
59. geom\_abline**(**intercept**=**qpline**[1],** slope**=**qpline**[2],** size**=1,** aes**(**color**=**"quadprog"**),** show**.**legend**=**TRUE**)** **+**
60. geom\_abline**(**intercept**=**svmline**[1],** slope**=**svmline**[2],** size**=1,** aes**(**color**=**"svm"**))+**
61. scale\_fill\_manual**(values=**c**(**"red"**,**"blue"**),** guide**=**'none'**)+**
62. scale\_color\_manual**(values=**c**(**"green"**,** "yellow"**,** "black"**))+**
63. theme**(**legend**.**position**=**"bottom"**,** legend**.**title**=**element\_blank**())**
64. **print(**plt**)**
65. # print the results
66. **print(sprintf(**"quadprog number of nonzeros: %d"**,** sum**(abs(**qpsol**)>1e-7)))**
67. **print(sprintf(**"svm number of nonzeros: %d"**,** **length(**sv$coefs**)))**
68. **print(sprintf(**"Quadprog: Intercept: %f Slope: %f"**,** qpline**[1],** qpline**[2]))**
69. **print(sprintf(**"svm: Intercept: %f Slope: %f"**,** svmline**[1],** svmline**[2]))**
70. ##########################################
71. # Perform vericolor and virginica
72. # Similar with setosa and vericolor but different values
73. ## Reset program ##
74. # use to remove all objects
75. rm**(**list**=**ls**(**all**=**TRUE**))**
76. #Set my working directory
77. setwd**(**"D:/ChuongStuff/Bio/HW6"**)**
78. #Check the directory
79. getwd**()**
80. # Read and Store data
81. iris**.**ver**.**vir **=** **read.**table**(**"Iris-ver.vir.txt"**)**
82. #Loading library
83. library**(**quadprog**)**
84. library**(**e1071**)**
85. train <- iris.ver.vir
86. train$y <-ifelse(train[,5]=="versicolor", 1, -1)
87. # order the training data labeling to avoid oddities with
88. # see http://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html#f430
89. train <- train[order(train$y, decreasing=TRUE),]
90. # set the problem data and parameters
91. X <- as.matrix(train[,c("Petal.Length", "Petal.Width")])
92. y <- as.matrix(train$y)
93. n <- dim(X)[1]
94. eps <- 5e-4
95. # build the system matrices
96. Q <- sapply(1:n, function(i) y[i]\*t(X)[,i])
97. D <- t(Q)%\*%Q
98. d <- matrix(1, nrow=n)
99. b0 <- rbind( matrix(0, nrow=1, ncol=1) , matrix(0, nrow=n, ncol=1))
100. A <- t(rbind(matrix(y, nrow=1, ncol=n), diag(nrow=n)))
101. # call the QP solver:
102. sol <- solve.QP(D +eps\*diag(n), d, A, b0, meq=0, factorized=FALSE)
103. qpsol <- matrix(sol$solution, nrow=n)
104. ####################################################################################################################
105. # This solver is implemented to solve a soft-margin problem. If we choose C very large,
106. # then the margins will be very small and we'll approach the hard-margin classification problem that we solved above
107. ####################################################################################################################
108. C <- 1e5 # Huge value forces hard margin problem
109. sv <- svm(Species~Petal.Length+Petal.Width, data=train, kernel="linear", scale=FALSE, type="C-classification", cost=C)
110. # Now we get the slope and intercept
111. W <- rowSums(sapply(1:length(sv$coefs), function(i) sv$coefs[i]\*sv$SV[i,]))
112. svmline **=** c**(**sv$rho/W**[2],** **-**W**[1]/**W**[2])**
113. # extract and plot the decision boundary from the results of each solver.
114. # build the support vectors, slopes, and intercepts
115. findLine <- function(a, y, X){
116. nonzero <- abs(a) > **1e-5**
117. W <- rowSums(sapply(which(nonzero), function(i) a[i]\*y[i]\*X[i,]))
118. b <- mean(sapply(which(nonzero), function(i) X[i,]%\*%W- y[i]))
119. slope <- -W[1]/W[2]
120. intercept <- b/W[2]
121. **return(**c**(**intercept**,**slope**))**
122. **}**
123. qpline <- findLine( qpsol, y, X)
124. # plot the results
125. library**(**ggplot2**)**
126. plt <- ggplot(train, aes(x=Petal.Length, y=Petal.Width)) +
127. ggtitle**(**"Solving the SVM QP for Vericolor and Virginica"**)** **+**
128. geom\_point**(**aes**(**fill**=**factor**(**y**)),** size**=3,** pch**=21)** **+**
129. geom\_abline**(**intercept**=**qpline**[1],** slope**=**qpline**[2],** size**=1,** aes**(**color**=**"quadprog"**),** show**.**legend**=**TRUE**)** **+**
130. geom\_abline**(**intercept**=**svmline**[1],** slope**=**svmline**[2],** size**=1,** aes**(**color**=**"svm"**))+**
131. scale\_fill\_manual**(values=**c**(**"red"**,**"blue"**),** guide**=**'none'**)+**
132. scale\_color\_manual**(values=**c**(**"green"**,** "yellow"**,** "black"**))+**
133. theme**(**legend**.**position**=**"bottom"**,** legend**.**title**=**element\_blank**())**
134. **print(**plt**)**
135. # print the results
136. **print(sprintf(**"quadprog number of nonzeros: %d"**,** sum**(abs(**qpsol**)>1e-7)))**
137. **print(sprintf(**"svm number of nonzeros: %d"**,** **length(**sv$coefs**)))**
138. **print(sprintf(**"Quadprog: Intercept: %f Slope: %f"**,** qpline**[1],** qpline**[2]))**
139. **print(sprintf(**"svm: Intercept: %f Slope: %f"**,** svmline**[1],** svmline**[2]))**
140. ##########################################
141. # Perform full size of iris
142. ## Reset program ##
143. # use to remove all objects
144. rm**(**list**=**ls**(**all**=**TRUE**))**
145. #Set my working directory
146. setwd**(**"D:/ChuongStuff/Bio/HW6"**)**
147. #Check the directory
148. getwd**()**
149. #Loading library
150. library**(**quadprog**)**
151. library**(**e1071**)**
152. train <- iris
153. train$y <-ifelse(train[,5]=="setosa", 1, -1)
154. # order the training data labeling to avoid oddities with
155. # see http://www.csie.ntu.edu.tw/~cjlin/libsvm/faq.html#f430
156. train <- train[order(train$y, decreasing=TRUE),]
157. # set the problem data and parameters
158. X <- as.matrix(train[,c("Petal.Length", "Petal.Width")])
159. y <- as.matrix(train$y)
160. n <- dim(X)[1]
161. eps <- 5e-4
162. # build the system matrices
163. Q <- sapply(1:n, function(i) y[i]\*t(X)[,i])
164. D <- t(Q)%\*%Q
165. d <- matrix(1, nrow=n)
166. b0 <- rbind( matrix(0, nrow=1, ncol=1) , matrix(0, nrow=n, ncol=1))
167. A <- t(rbind(matrix(y, nrow=1, ncol=n), diag(nrow=n)))
168. # call the QP solver:
169. sol <- solve.QP(D +eps\*diag(n), d, A, b0, meq=0, factorized=FALSE)
170. qpsol <- matrix(sol$solution, nrow=n)
171. ####################################################################################################################
172. # This solver is implemented to solve a soft-margin problem. If we choose C very large,
173. # then the margins will be very small and we'll approach the hard-margin classification problem that we solved above
174. ####################################################################################################################
175. C <- 1e5 # Huge value forces hard margin problem
176. sv <- svm(y~Petal.Length+Petal.Width, data=train, kernel="linear", scale=FALSE, type="C-classification", cost=C)
177. # Now we get the slope and intercept
178. W <- rowSums(sapply(1:length(sv$coefs), function(i) sv$coefs[i]\*sv$SV[i,]))
179. svmline **=** c**(**sv$rho/W**[2],** **-**W**[1]/**W**[2])**
180. # extract and plot the decision boundary from the results of each solver.
181. # build the support vectors, slopes, and intercepts
182. findLine <- function(a, y, X){
183. nonzero <- abs(a) > **1e-5**
184. W <- rowSums(sapply(which(nonzero), function(i) a[i]\*y[i]\*X[i,]))
185. b <- mean(sapply(which(nonzero), function(i) X[i,]%\*%W- y[i]))
186. slope <- -W[1]/W[2]
187. intercept <- b/W[2]
188. **return(**c**(**intercept**,**slope**))**
189. **}**
190. qpline <- findLine( qpsol, y, X)
191. # plot the results
192. library**(**ggplot2**)**
193. plt <- ggplot(train, aes(x=Petal.Length, y=Petal.Width)) +
194. ggtitle**(**"Solving the SVM QP"**)** **+**
195. geom\_point**(**aes**(**fill**=**factor**(**y**)),** size**=3,** pch**=21)** **+**
196. geom\_abline**(**intercept**=**qpline**[1],** slope**=**qpline**[2],** size**=1,** aes**(**color**=**"quadprog"**),** show**.**legend**=**TRUE**)** **+**
197. geom\_abline**(**intercept**=**svmline**[1],** slope**=**svmline**[2],** size**=1,** aes**(**color**=**"svm"**))+**
198. scale\_fill\_manual**(values=**c**(**"red"**,**"blue"**),** guide**=**'none'**)+**
199. scale\_color\_manual**(values=**c**(**"green"**,** "yellow"**,** "black"**))+**
200. theme**(**legend**.**position**=**"bottom"**,** legend**.**title**=**element\_blank**())**
201. **print(**plt**)**
202. # print the results
203. **print(sprintf(**"quadprog number of nonzeros: %d"**,** sum**(abs(**qpsol**)>1e-7)))**
204. **print(sprintf(**"svm number of nonzeros: %d"**,** **length(**sv$coefs**)))**
205. **print(sprintf(**"Quadprog: Intercept: %f Slope: %f"**,** qpline**[1],** qpline**[2]))**
206. **print(sprintf(**"svm: Intercept: %f Slope: %f"**,** svmline**[1],** svmline**[2]))**