Support Vector Machine Implementation

Chanwoo Lee, March 26, 2020

1 Linear implementation

1.1 Primal and Dual problems

Primal and dual problem of Hard SVM is as follow

$$(P) \quad \min_{\boldsymbol{w},b} \|\boldsymbol{w}\|^2 \quad \text{s.t.} \forall i, y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1, \tag{1}$$

$$(D) \quad \max_{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} \geq 0} \left(\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \right).$$

Primal and dual problem of Soft SVM is as follow

$$\begin{aligned} (P) \qquad \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i=1}^m \xi_i \\ \text{subject to} \quad & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{2}$$

$$\begin{aligned} (D) \qquad \max_{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} \geq 0} & \left(\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \right) \\ \text{subject to} \quad & \sum_{i=1}^m y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1. \cdots, m. \end{aligned}$$

Here $C = \frac{1}{\lambda}$ in the previous note. The problem (3) are equivalent to (1) as $C \to \infty$. We can obtain primal solution from solving dual problem with the relationship $\boldsymbol{w} = \sum_{i=1}^m y_i \alpha_i \boldsymbol{x}_i$. and we can get intercept as

$$b = \min_{y_i = 1} \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b = 1 \text{ or } b = \max_{y_i = -1} \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b = -1$$

1.2 Algorithm implementation

I used `e1071` package for SVM. In this package, they do not offer Hard SVM separately. However, We can get Hard SVM analysis result by setting C large in Soft SVM. In simulation 1, I made data set separable generating y from pre-assigned classification function. In simulation 2, I made data set inseparable generating y randomly and add $(1.5, 1.5)^T$ to feature vectors such that $y_i = 1$. Implementation results are in Figure 1.

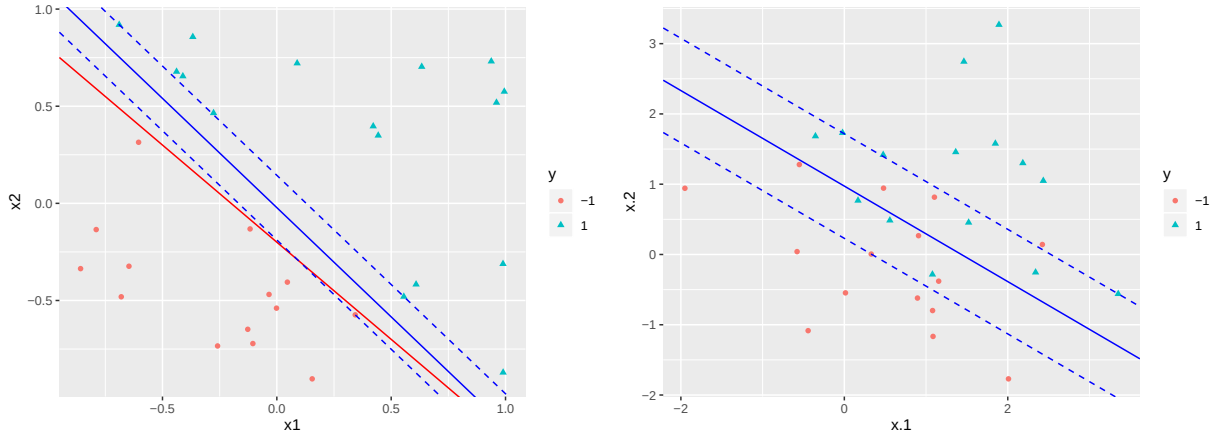


Figure 1: The left figure is the result of Hard SVM applied to separable data set. Red line is true classification rule. Blue line is estimated classification rule and dotted lines show margins of the classification. The right figure is the result of Soft SVM applied to inseparable data set. The cost parameter (C in (3)) is 10. Blue line is estimated classification rule and dotted lines are margins of the classification.

2 Kernel implementation

Primal and Dual problem of kernel method are,

$$(P) \qquad \begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i(\langle \boldsymbol{w}, h(\boldsymbol{x}_i)\rangle + b) \geq 1-\xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned} \tag{3}$$

$$(D) \qquad \begin{aligned} \max_{\boldsymbol{\alpha}\in\mathbb{R}^m:\boldsymbol{\alpha}\geq 0} & \left(\sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)\rangle\right) \\ \text{subject to} \quad & \sum_{i=1}^{m} y_i\alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1.\cdots, m. \end{aligned}$$

Define $G(\boldsymbol{x}_i, \boldsymbol{x}_J) = \langle h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)\rangle$. The package `e1071` provides 3 options for the function G.

- linear : linear SVM
- polynomial : $G(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\gamma * \boldsymbol{x}_i^T\boldsymbol{x}_j + \text{coef0})^{\text{degree}}$
- radial : $G(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2)$
- sigmoid : $G(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\gamma * \boldsymbol{x}_i^T * \boldsymbol{x}_j + \text{ coef0})$

Default for parameters is that $\gamma = \frac{1}{\dim(\boldsymbol{x})}$, coef0 $= 0$ and degree $= 3$.

I used the data ESL.mixture.rda which is available in https://web.stanford.edu/~hastie/

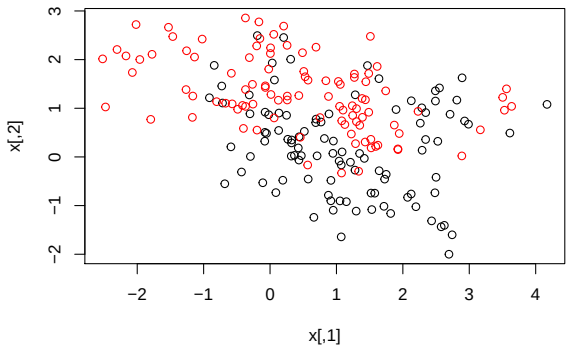


Figure 2: Red colored points have the label -1. Back colored points have the label +1

ElemStatLearn/datasets/ to apply kernel method. Figure 2 shows feature of the data. Figure 3 shows the SVM results.

3 SVM algorithm for matrix predictor

For matrix predictor $X_i \in \mathbb{R}^{m\times n}$, we have the optimization problem,

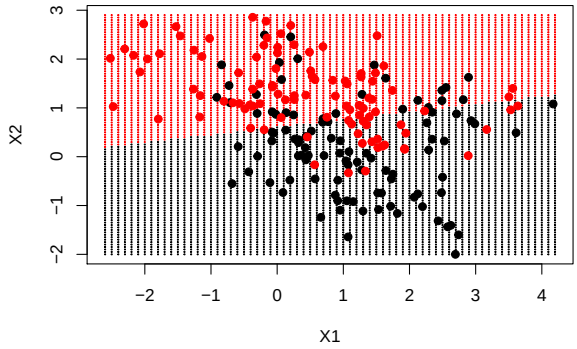
$$\begin{aligned} \min_{B,b,\boldsymbol{\xi}} & \frac{1}{2}\|B\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to } & y_i(\langle B, X_i\rangle + b) \geq 1-\xi_i, \\ & \xi_i \geq 0, \quad i = 1,\cdots,m, \\ & \text{rank}(B) \leq r. \end{aligned} \tag{4}$$

We use matrix factorization technique, where instead of optimizing with respect to B, it is factorized into two matrices $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ such that $B = UV^T$. One then optimize with respect to U and V. Fixing V, we have the following optimization problem from (4).

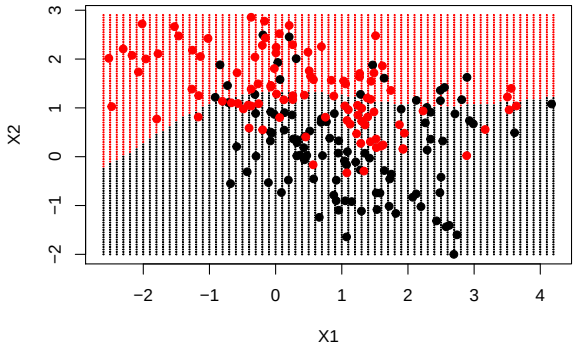
$$\begin{aligned} \min_{U,b,\boldsymbol{\xi}} & \frac{1}{2}\,\text{Vec}(U)^T\left(V^TV \otimes I_m\right)\text{Vec}(U) + C\sum_{i=1}^{m}\xi_i \\ \text{subject to } & y_i(\langle \text{Vec}(U), \text{Vec}(X_iV)\rangle + b) \geq 1-\xi_i, \\ & \xi_i \geq 0, \quad i = 1,\cdots,m. \end{aligned}$$

Thereby, we can solve the almost same problem of (3) with slight modification on the loss function. We can update V fixing U by the same way. Define $\boldsymbol{u} = \text{Vec}(U) \in \mathbb{R}^{mr}, Q = \left(V^TV \otimes I_m\right)$ and $\boldsymbol{x}_i^v = \text{Vec}(X_iV)$. Notice Q is positive definite as long as V is not a singular matrix. Therefore, Q-norm $\|\boldsymbol{u}\|_Q = \sqrt{\boldsymbol{u}^TQ\boldsymbol{u}}$ is well defined. Then primal and dual problem for updating U fixing V is as follows,

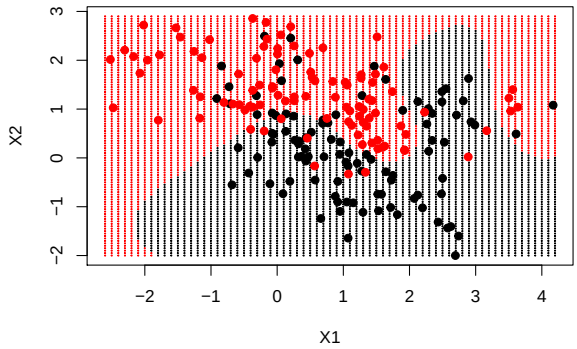
$$\begin{aligned} (P) \qquad \min_{\boldsymbol{u},b,\boldsymbol{\xi}} & \quad \frac{1}{2}\|\boldsymbol{u}\|_Q^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} & \quad y_i(\langle \boldsymbol{u}, \boldsymbol{x}_i^v\rangle + b) \geq 1-\xi_i, \end{aligned}$$



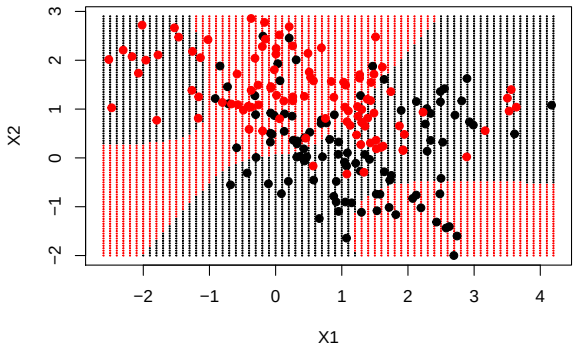
(a) Soft SVM method: MSR = 0.73



(b) Polynomial kernel method: MSR = 0.67



(c) Radial kernel method: MSR = 0.83



(d) Sigmoid kernel method: MSR = 0.52

Figure 3: One linear SVM and Three nonlinear SVMs for the esl data. Mathematical expression of each boundary is $y_i(\langle \hat{\boldsymbol{w}}, h(\boldsymbol{x})\rangle + b) \geq 1$ where h is the corresponding kernel to each type. MSR is misclassification rate

$$\xi_i \geq 0, \quad i = 1, \cdots, m.$$

$$(D) \qquad \max_{\boldsymbol{\alpha}\in\mathbb{R}^m:\boldsymbol{\alpha}\geq 0} \left(\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i^v)^T Q^{-1} \boldsymbol{x}_j^v \right)$$

$$\text{subject to} \quad \sum_{i=1}^{m} y_i \alpha_i = 0,$$

$$0 \leq \alpha_i \leq C, \quad i = 1. \cdots, m.$$

From the solution of the dual problem, we can obtain $\boldsymbol{u}$ by

$$\boldsymbol{u} = \sum_{i=1}^{m} \alpha_i y_i Q^{-1} \boldsymbol{x}_i^v = \sum_{i=1}^{m} \alpha_i y_i \operatorname{Vec}\left((X_i V (V^T V)^{-1} \right).$$

There are two approaches to solve the daul probelm

- Equivalent representation as Kernel method.

  Notice that the dual problem is equivalent to kernel method problem if we define

$$\begin{aligned} G(i,j) &= (\boldsymbol{x}_i^v)^T Q^{-1} \boldsymbol{x}_j^v = \langle X_i V, X_j V (V^T V)^{-1} \rangle = \langle X_i V (V^T V)^{-1}, X_j V \rangle \\ &= \langle X_i, X_j \rangle_H \quad \text{where } H = V(V^T V)^{-1} V^T. \end{aligned}$$

  Therefore, we can use the general kernel method algorithm.

- Equivalent representation as linear SVM.

  Notice that the dual problem is equivalent to linear SVM if we define

$$\boldsymbol{z}_i = Q^{-1/2} \boldsymbol{x}_i^v.$$

  Therefore, we can use the linear SVM algorithm from $(\boldsymbol{z}_1, y_1) \cdots, (\boldsymbol{z}_m, y_m)$.

3.1 Two special cases

Let us consider two special cases: when B is full rank, when B is rank-1 ($B = \boldsymbol{u}\boldsymbol{v}^T$).

1. When B is full rank

   There is no good reason to factorize matrix B into U and V. In this case, the primal problem is

$$\begin{aligned} \min_{B,b,\boldsymbol{\xi}} & \frac{1}{2} \| \operatorname{Vec}(B) \|^2 + C \sum_{i=1}^m \xi_i \\ \text{subject to } & y_i (\langle \operatorname{Vec}(B), \operatorname{Vec}(X_i) \rangle + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned}$$

   Therefore, everything is the same as vector predictor SVM.

2. When B is rank-1 ($B = \boldsymbol{u}\boldsymbol{v}^T$)

   While fixing $\boldsymbol{v}$, the primer and dual problems are,

$$\begin{aligned} (P) \qquad \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2} \|\boldsymbol{v}\|^2 \|\boldsymbol{u}\|^2 + C \sum_{i=1}^m \xi_i \\ \text{subject to} \quad & y_i (\boldsymbol{u}^T X_i \boldsymbol{v} + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, \cdots, m. \end{aligned}$$

$$\begin{aligned} (D) \qquad \max_{\boldsymbol{\alpha} \in \mathbb{R}^m : \boldsymbol{\alpha} \geq 0} & \left(\sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \langle \frac{X_i \boldsymbol{v}}{\|\boldsymbol{v}\|}, \frac{X_j \boldsymbol{v}}{\|\boldsymbol{v}\|} \rangle \right) \\ \text{subject to} \quad & \sum_{i=1}^m y_i \alpha_i = 0, \\ & 0 \leq \alpha_i \leq C, \quad i = 1. \cdots, m. \end{aligned}$$

We can estimate $\boldsymbol{u}$ from the dual solution using

$$\boldsymbol{u} = \frac{1}{\|\boldsymbol{v}\|} \sum_{i=1}^{m} \alpha_i y_i \frac{X_i \boldsymbol{v}}{\|\boldsymbol{v}\|}$$

4 Rcodes

```r
#SVM
library(e1071)
library(ggplot2)

## Simulation1(HARD margin)
x <- as.data.frame(cbind(runif(30,-1,1),runif(30,-1,1)))
colnames(x) = c("x1","x2")
b = c(1,1) ; b0 = .2
y <- apply(x,1,function(x) ifelse(b%*%x+b0>0,1,-1))
y <- as.factor(y)
simdat = cbind(x,y)


ggplot(data = simdat,aes(x1,x2,colour = y))+geom_point()+
  geom_abline(intercept = -b0/b[2], slope = -b[1]/b[2], color="red")

#Fit SVM to data set
model <- svm(y~.,data = simdat,kernel = "linear",cost = 10000,scale = FALSE)
bhat = t(model$coefs)%*%model$SV
b0hat = model$rho
# Plot Results
ggplot(data = simdat,aes(x1,x2,colour = y,shape = y))+geom_point()+
  geom_abline(intercept = -b0/b[2], slope = -b[1]/b[2], color="red")+
  geom_abline(intercept = (b0hat)/bhat[2],slope = -bhat[1]/bhat[2],color = "blue")
    +
  geom_abline(intercept = (b0hat+1)/bhat[2],slope = -bhat[1]/bhat[2],color = "blue
    ",lty = 2)+
  geom_abline(intercept = (b0hat-1)/bhat[2],slope = -bhat[1]/bhat[2],color = "blue
    ",lty = 2)




## Simulation2(SOFT margin)
x <- matrix(rnorm(30*2), ncol = 2)
y <- c(rep(-1,15), rep(1,15))
x[y==1,] <- x[y==1,] + 3/2
dat <- data.frame(x=x, y=as.factor(y))


# Fit SVM to data set
svmfit <- svm(y~., data = dat, kernel = "linear",cost = 10, scale = FALSE)
bhat = t(svmfit$coefs)%*%svmfit$SV
b0hat = svmfit$rho
# Plot Results
ggplot(data = dat,aes(x.1,x.2,colour = y, shape = y))+geom_point()+
  geom_abline(intercept = (b0hat)/bhat[2],slope = -bhat[1]/bhat[2],color = "blue")
    +
  geom_abline(intercept = (b0hat+1)/bhat[2],slope = -bhat[1]/bhat[2],color = "blue
    ",lty = 2)+
```

```
  geom_abline(intercept = (b0hat-1)/bhat[2],slope = -bhat[1]/bhat[2],color = "blue
    ",lty = 2)


#Simulation for Kernel
load(file = "ESL.mixture.rda")
names(ESL.mixture)
rm(x,y)
attach(ESL.mixture)
plot(x, col = y + 1)
dat = data.frame(y = factor(y), x)

## linear
fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "linear", cost = 5)
xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)
## Training error
length(which(y == predict(fit,x)))/length(y)


## Radial basis
fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "radial", cost = 5)
xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)
## Training error
length(which(y == predict(fit,x)))/length(y)

## Polynomial
fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "polynomial", cost =
    5)
xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)
## Training error
length(which(y == predict(fit,x)))/length(y)


## Sigmoid
fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "sigmoid", cost = 5)
xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)
## Training error
length(which(y == predict(fit,x)))/length(y)


## Polynomial
fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "polynomial", cost =
    5)
```

```r
xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)


## Sigmoid
fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "sigmoid", cost = 5)
xgrid = expand.grid(X1 = px1, X2 = px2)
ygrid = predict(fit, xgrid)

plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
points(x, col = y + 1, pch = 19)
```