

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

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

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

Primal and dual problem of Soft SVM is as follow

$$(P) \quad \min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i \quad (2)$$

$$\text{subject to } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i,$$

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

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

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

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

Here $C = \frac{1}{\lambda}$ in the previous note. The problem (5) are equivalent to (1) as $C \rightarrow \infty$. We can obtain primal solution from solving dual problem with the relationship $\mathbf{w} = \sum_{i=1}^m y_i \alpha_i \mathbf{x}_i$. and we can get intercept as

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

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.

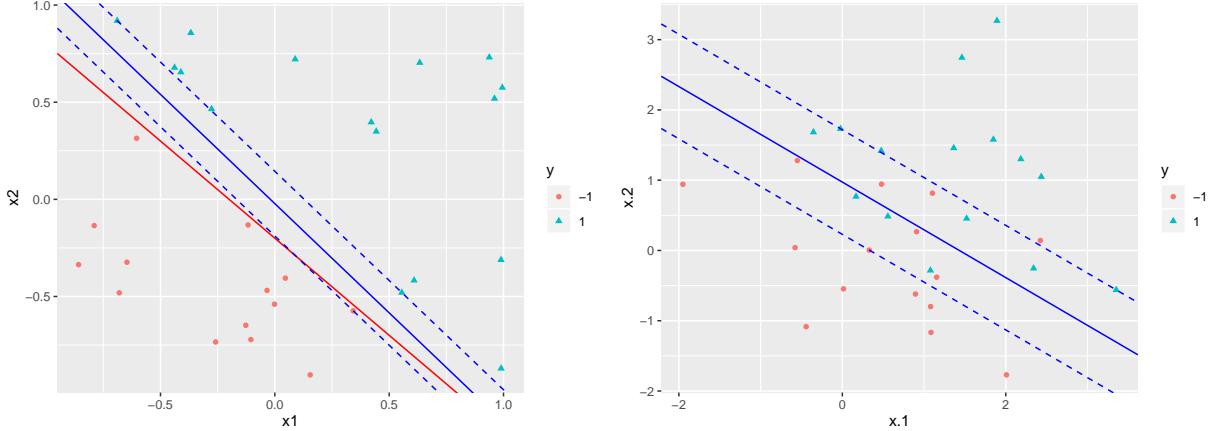


Figure 1: The left figure is the result of Hard SVM applied to separable data set. Red line is true classification rule. Blue line is estimated classification rule and dotted lines show margins of the classification. The right figure is the result of Soft SVM applied to inseparable data set. **The cost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

definition?

2 Kernel implementation

Primal and Dual problem of kernel method are,

$$(P) \quad \min_{\mathbf{w}, b, \xi} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \xi_i \quad (3)$$

subject to $y_i(\langle \mathbf{w}, h(\mathbf{x}_i) \rangle + b) \geq 1 - \xi_i,$
 $\xi_i \geq 0, \quad i = 1, \dots, m.$

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

subject to $\sum_{i=1}^m y_i \alpha_i = 0,$
 $0 \leq \alpha_i \leq C, \quad i = 1, \dots, m.$

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

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

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

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

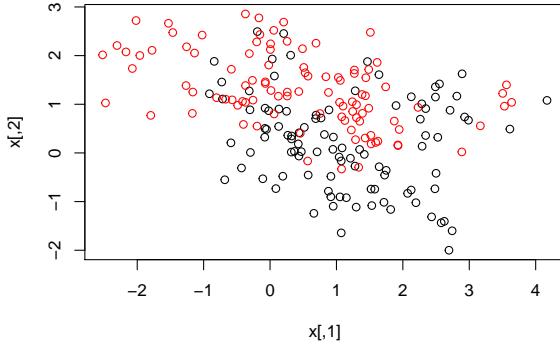


Figure 2: Red colored points have the label -1. Black colored points have the label +1

[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,

$$\min_{B,b,\xi} \frac{1}{2} \|B\|^2 + C \sum_{i=1}^m \xi_i \quad (4)$$

$$\text{subject to } y_i(\langle B, X_i \rangle + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \quad i = 1, \dots, m,$$

$$\text{rank}(B) \leq r.$$

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).

Q: identifiability?

Suppose B can be decomposed in two ways $\min_{U,b,\xi} \frac{1}{2} \text{Vec}(U)^T (V^T V \otimes I_m) \text{Vec}(U) + C \sum_{i=1}^m \xi_i$
 $U_1 V_1^T = U_2 V_2^T \rightarrow U_1 \sim U_2$ up to orthogonal transformation..

Does the algorithm give same estimated B ?

$$\text{subject to } y_i(\langle \text{Vec}(U), \text{Vec}(X_i V) \rangle + b) \geq 1 - \xi_i,$$

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

mr

Thereby, we can solve the almost same problem of (5) with slight modification on the loss function. We can update V fixing U by the same way. Define $\mathbf{u} = \text{Vec}(U) \in \mathbb{R}^{mn}$, $Q = (V^T V \otimes I_m)$ and $\mathbf{x}_i^v = \text{Vec}(X_i V)$. Notice Q is positive definite as long as V is not a singular matrix. Therefore, Q-norm $\|\mathbf{x}\|_Q = \sqrt{\mathbf{x}^T Q \mathbf{x}}$ is well defined. Then primal and dual problem for updating U fixing V is as follows,

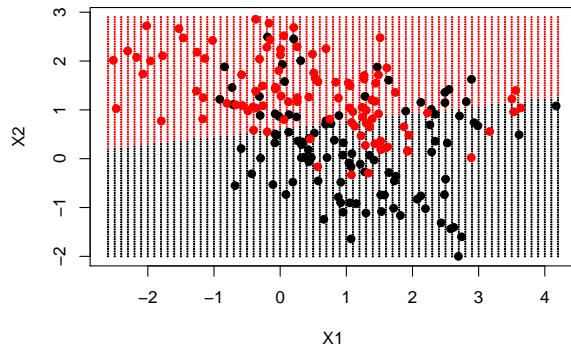
x → u. x has been reserved for feature in a space of

dim=mn

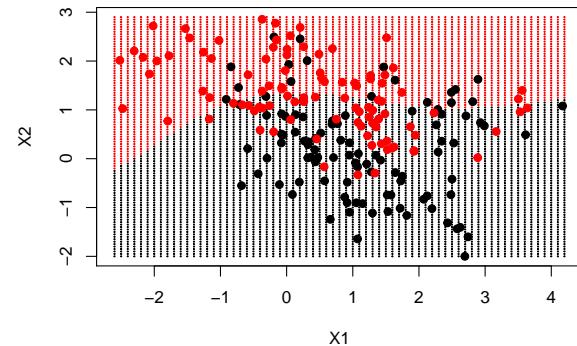
$$(P) \quad \min_{\mathbf{u}, b, \xi} \frac{1}{2} \|\mathbf{u}\|_Q^2 + C \sum_{i=1}^m \xi_i \quad (5)$$

$$\text{subject to } y_i(\langle \mathbf{u}, \mathbf{x}_i^v \rangle + b) \geq 1 - \xi_i,$$

Q: what is the mathematical expression for estimated decision boundary? (see highlighted lines in your codes)

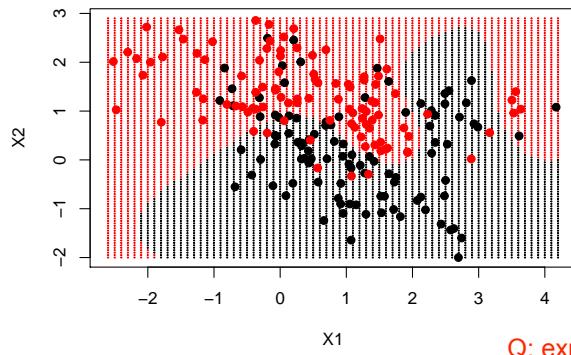


(a) Linear kernel method (Soft SVM method)

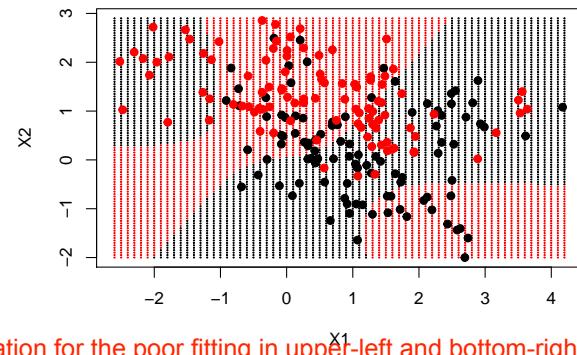


Which kernel is better? Training errors?

(b) Polynomial kernel method



(c) Radial kernel method



Q: explanation for the poor fitting in upper-left and bottom-right corners?
Underfitting? C too small?

(d) Sigmoid kernel method

Figure 3: One linear SVM and Three nonlinear SVMs for the esl data.

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

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

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

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

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

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

There are two approaches to solve the dual problem

- Equivalent representation as Kernel method.

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

$$G(i, j) = (\mathbf{x}_i^v)^T Q^{-1} \mathbf{x}_j^v = \langle X_i V, X_j V (V^T V)^{-1} \rangle = \langle X_i V (V^T V)^{-1}, X_j V \rangle.$$

$$\dots = \langle X V (V^T V)^{-1} V^T, X \rangle = \langle X, X \rangle_H, \text{ where } H = V (V^T V)^{-1} V^T$$

Therefore, we can use the general kernel method algorithm.

- Equivalent representation as linear SVM.

Q: connection to projection (hat) matrix in linear regression?
Geometric intuition for the induced kernel? Similarly for V.

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

$$z_i = Q^{-1/2} \mathbf{x}_i^v.$$

Therefore, we can use the linear SVM algorithm from $(z_1, y_1), \dots, (z_m, y_m)$.

Consider two special cases: (1) B has rank 1; (2) B has full rank. How does the problem simplify? Any insight?

4 Rcodes

```

1 # SVM
2 library(e1071)
3 library(ggplot2)
4
5 ## Simulation1(HARD margin)
6 x <- as.data.frame(cbind(runif(30,-1,1),runif(30,-1,1)))
7 colnames(x) = c("x1", "x2")
8 b = c(1,1); b0 = .2
9 y <- apply(x, 1, function(x) ifelse(b %*% x + b0 > 0, 1, -1))
10 y <- as.factor(y)
11 simdat = cbind(x,y)
12
13
14 ggplot(data = simdat, aes(x1,x2, colour = y)) + geom_point() +
15   geom_abline(intercept = -b0/b[2], slope = -b[1]/b[2], color="red")
16
17 #Fit SVM to data set
18 model <- svm(y ~ ., data = simdat, kernel = "linear", cost = 10000, scale = FALSE)
19 bhat = t(model$coefs) %*% model$SV
20 b0hat = model$rho
21 # Plot Results
22 ggplot(data = simdat, aes(x1,x2, colour = y, shape = y)) + geom_point() +
23   geom_abline(intercept = -b0/b[2], slope = -b[1]/b[2], color="red") +
24   geom_abline(intercept = (b0hat)/bhat[2], slope = -bhat[1]/bhat[2], color = "blue") +
25   geom_abline(intercept = (b0hat+1)/bhat[2], slope = -bhat[1]/bhat[2], color = "blue",
26               lty = 2) +
27   geom_abline(intercept = (b0hat-1)/bhat[2], slope = -bhat[1]/bhat[2], color = "blue",
28               lty = 2)
29
30
31 ## Simulation2(SOFT margin)
32 x <- matrix(rnorm(30*2), ncol = 2)
33 y <- c(rep(-1,15), rep(1,15))
34 x[y==1,] <- x[y==1,] + 3/2
35 dat <- data.frame(x=x, y=as.factor(y))
36
37

```

```

38 # Fit SVM to data set
39 svmfit <- svm(y~., data = dat, kernel = "linear", cost = 10, scale = FALSE)
40 bhat = t(svmfit$coefs) %*% svmfit$SV
41 b0hat = svmfit$rho
42 # Plot Results
43 ggplot(data = dat, aes(x.1,x.2, colour = y, shape = y)) + geom_point() +
44   geom_abline(intercept = (b0hat)/bhat[2], slope = -bhat[1]/bhat[2], color = "blue") +
45   geom_abline(intercept = (b0hat+1)/bhat[2], slope = -bhat[1]/bhat[2], color = "blue",
46   "lty" = 2) +
46   geom_abline(intercept = (b0hat-1)/bhat[2], slope = -bhat[1]/bhat[2], color = "blue",
47   "lty" = 2)
48
49 #Simulation for Kernel
50 load(file = "ESL.mixture.rda")
51 names(ESL.mixture)
52 rm(x,y)
53 attach(ESL.mixture)
54 plot(x, col = y + 1)
55 dat = data.frame(y = factor(y), x)
56
57 ## linear
58 fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "linear", cost = 5)
59 xgrid = expand.grid(X1 = px1, X2 = px2)
60 ygrid = predict(fit, xgrid)
61
62 plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
63 points(x, col = y + 1, pch = 19)
64
65 ## Radial basis
66 fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "radial", cost = 5)
67 xgrid = expand.grid(X1 = px1, X2 = px2)
68 ygrid = predict(fit, xgrid)
69
70 plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
71 points(x, col = y + 1, pch = 19)
72
73 ## Polynomial
74 fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "polynomial", cost =
75 5)
75 xgrid = expand.grid(X1 = px1, X2 = px2)
76 ygrid = predict(fit, xgrid)
77
78 plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
79 points(x, col = y + 1, pch = 19)
80
81
82 ## Sigmoid
83 fit = svm(factor(y) ~ ., data = dat, scale = FALSE, kernel = "sigmoid", cost = 5)
84 xgrid = expand.grid(X1 = px1, X2 = px2)
85 ygrid = predict(fit, xgrid)
86
87 plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = .2)
88 points(x, col = y + 1, pch = 19)

```