

SUPPORT VECTOR MACHINES 1

-INTRODUCTION TO DATA SCIENCE-

ISL Chapter 9.1 and 9.2

Lecturer: Darren Homrighausen, PhD

Preamble:

- The optimal separating hyperplane maximizes the minimum Euclidean distance between a separating hyperplane and the data
- Support Vector Classifiers generalize the optimal separating hyperplane by allowing for margin violations and training misclassifications

OPTIMAL SEPARATING HYPERPLANES

A main initiative in early computer science was to find **separating hyperplanes** among groups of data

The issue is that if there is a separating hyperplane, there is an infinite number

An **optimal separating hyperplane** can be generated by finding **support points** and bisecting them.

(The book calls the **optimal separating hyperplane** the **maximum margin classifier**)

BASIC LINEAR GEOMETRY

A hyperplane in \mathbb{R}^p is given by

$$\mathcal{H} = \{X \in \mathbb{R}^p : f(X) = \beta_0 + \beta^\top X = 0\}$$

1. The vector β is **normal** to \mathcal{H}

(To see this, let $X, X' \in \mathcal{H}$. Then $\beta^\top (X - X') = 0$)

2. **IMPORTANT:** If $\|\beta\|_2 = 1$, then for any point $X \in \mathbb{R}^p$, the (signed) length of its orthogonal complement to \mathcal{H} is $f(X)$

SUPPORT VECTOR MACHINES (SVM)

Let $Y_i \in \{-1, 1\}$

(It is common with SVMs to code Y this way. With logistic regression, Y is commonly phrased as $\{0, 1\}$ due to the connection with Bernoulli trials)

We will generalize this to supervisors with more than 2 levels at the end

A classification rule induced by a hyperplane is

$$g(X) = \text{sgn}(X^\top \beta + \beta_0) = \text{sgn}(f(X))$$

SEPARATING HYPERPLANES

Our classification rule is based on a hyperplane \mathcal{H}

$$g(X) = \text{sgn}(X^\top \beta + \beta_0)$$

A **correct** (training) classification: $f(X_i)Y_i > 0$ & $g(X_i)Y_i > 0$

The signed distance to \mathcal{H} is $f(X) = X^\top \beta + \beta_0$

Under classical **separability**, we can find a function f such that $Y_i f(X_i) > 0$

(That is, makes perfect classifications via $g(X) = \text{sgn}(f(X))$)

The larger the quantity $Y_i f(X_i)$, the more **separated** the classes

OPTIMAL SEPARATING HYPERPLANE

This idea can be encoded in the following **convex program**

$\max_{\beta_0, \beta} M$ subject to

$$Y_i f(X_i) \geq M \text{ for each } i \quad \text{and} \quad \|\beta\|_2 = 1$$

INTUITION:

- We know that $Y_i f(X_i) > 0 \Rightarrow g(X_i) = Y_i$. Hence, larger $Y_i f(X_i) \Rightarrow$ “more” correct classification
- For “more” to have any meaning, we need to **normalize** β , thus the other constraint

OPTIMAL SEPARATING HYPERPLANE

Taking another look at the optimization problem:

$\max_{\beta_0, \beta} M$ subject to

$$\underbrace{Y_i f(X_i) \geq M \text{ for each } i \quad \text{and} \quad \|\beta\|_2 = 1}_{\text{Combine: } \frac{Y_i f(X_i)}{\|\beta\|_2} \geq M \text{ for each } i}$$

(Note that this is the same as $Y_i f(X_i) \geq M \|\beta\|_2$)

For any $a > 0$, if we form $a f(X)$, the constraint still holds:

$$Y_i(a\beta_0 + a\beta^\top X_i) \geq \|a\beta\|_2 M$$

Hence, without any loss of generality, we can set

$$\|\beta\|_2 M \stackrel{\text{set}}{=} 1 \Leftrightarrow \left(M = \frac{1}{\|\beta\|_2} \quad \text{and} \quad Y_i f(X_i) \geq 1 \right)$$

OPTIMAL SEPARATING HYPERPLANE

Using these insights, we form the equivalent program

$$\min_{\beta_0, \beta} \frac{1}{2} \|\beta\|_2^2 \text{ subject to}$$
$$Y_i f(X_i) \geq 1 \text{ for each } i$$

(FACTS: $\operatorname{argmin} \|\beta\|_2 = \operatorname{argmin} \frac{1}{2} \|\beta\|_2^2$ and $\max M = \min \|\beta\|_2$ as $M = \frac{1}{\|\beta\|_2}$)

Hence, the **margin** around the hyperplane has width $\frac{1}{\|\beta\|_2}$

This is still a convex optimization program

(quadratic criterion, linear inequality constraints)

OPTIMAL SEPARATING HYPERPLANE

Again, we can convert this **constrained** optimization problem into a **Lagrangian**

$$\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|_2^2 - \sum_{i=1}^n \alpha_i [Y_i (X_i^\top \beta + \beta_0) - 1]$$

(It is subtraction as the constraint is \geq while lasso is \leq)

In contrast to the lasso problem, there are now n Lagrangian parameters $\alpha_1, \dots, \alpha_n$

(There are n constraints, after all)

Everything is nice and smooth \rightarrow take derivatives to optimize

ALSO: A general fact is that at the minimum,

$$\alpha_i [Y_i f(X_i) - 1] = 0 \quad \text{for all } i = 1, \dots, n$$

(This comes from the **Karush-Kuhn-Tucker** conditions for constrained optimization)

OPTIMAL SEPARATING HYPERPLANE

$$\frac{1}{2} \|\beta\|_2^2 - \sum_{i=1}^n \alpha_i [Y_i (X_i^\top \beta + \beta_0) - 1]$$

Derivatives with respect to β and β_0 imply that:

- $\beta = \sum_{i=1}^n \alpha_i Y_i X_i$
- $0 = \sum_{i=1}^n \alpha_i Y_i$

Substituting these equations back into the Lagrangian

$$\max_{\alpha_1, \dots, \alpha_n} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^n \alpha_i \alpha_k Y_i Y_k X_i^\top X_k$$

(this is all subject to $\alpha_i \geq 0$. This is known as the **Wolfe dual**. Note that this is a **maximization** problem. There is some optimization theory going on here that I'm skipping known as **strong duality** and **Slatter's condition**)

OPTIMAL SEPARATING HYPERPLANE

OBSERVE: Two important facts from the preceding slides:

$$\alpha_i[Y_i f(X_i) - 1] = 0 \quad \text{for all } i = 1, \dots, n$$

and

$$Y_i f(X_i) \geq 1 \text{ for each } i$$

Therefore,

- $\alpha_i = 0$, which happens if the constraint $Y_i f(X_i) > 1$
(That is, when the constraint is **non binding**)
- $\alpha_i > 0$, which happens if the constraint $Y_i f(X_i) = 1$
(That is, when the constraint is **binding**)

OPTIMAL SEPARATING HYPERPLANE

Taking this relationship

$$\alpha_i[Y_i f(X_i) - 1] = 0$$

and

$$\beta = \sum_{i=1}^n \alpha_i Y_i X_i$$

we see that, for $i = 1, \dots, n$,

- The points (X_i, Y_i) such that $\alpha_i > 0$ are **support vectors**
- The points (X_i, Y_i) such that $\alpha_i = 0$ are **irrelevant**

(Compare this idea to logistic regression which uses the exact same form of classifier ($\text{sgn}(f(X))$) but the fitted function always depends on **all** of the observations)

Support vector classifier

SUPPORT VECTOR CLASSIFIER

Of course, we can't realistically assume that the data are linearly separated (even in a transformed space)

In this case, the previous program has no **feasible** solution

We need to introduce **slack** variables, ξ , that allow for overlap among the classes

(ξ is pronounced “k-see” or sometimes “zai”)

These slack variables allow for us to encode **training misclassifications** into the optimization problem

SUPPORT VECTOR CLASSIFIER

$$\min_{\beta_0, \beta, \xi} \frac{1}{2} \|\beta\|_2^2 \quad \text{subject to}$$
$$Y_i f(X_i) \geq 1 \underbrace{(1 - \xi_i), \xi_i \geq 0, \sum \xi_i \leq t}_{\text{new}}, \text{ for each } i$$

(Convex optimization program. Equations 9.12-15 in ISL)

This can be rewritten as

$$\min_{\beta_0, \beta, \xi} \frac{1}{2} \|\beta\|_2^2 + \lambda \sum \xi_i \quad \text{subject to}$$
$$Y_i f(X_i) \geq 1 - \xi_i, \xi_i \geq 0, \text{ for each } i$$

Note that

- t or λ are the **tuning parameters**

(The literature usually refers to it as a **cost parameter**)

- The separable case corresponds to $t = 0$ or $\lambda = \infty$

SVMs: SLACK VARIABLES

Once we find $\hat{\beta}(\lambda)$ and $\hat{\beta}_0(\lambda)$ by solving the previous program, we make a classifier via

$$\hat{f}_\lambda(X) = \text{sgn}(\hat{\beta}(\lambda)^\top X + \hat{\beta}_0(\lambda)) = \text{sgn}(\hat{f}(X))$$

The **slack variables** give us insight into the problem

- If $\xi_i = 0$, then that observation is on **correct** side of the **margin**
- If $\xi_i \in (0, 1]$, then that observation is on the **incorrect** side of the **margin**, but still correctly classified
- If $\xi_i > 1$, then that observation is **incorrectly** classified

SVMs: TUNING PARAMETER

We can think of t as a **budget** for the problem

If $t = 0$, then there is **no budget** and we won't tolerate any margin violations

If $t > 0$, then no more than $\lfloor t \rfloor$ observations can be misclassified

A larger t then leads to larger **margins**

(we allow more margin violations)

SVMs: TUNING PARAMETER

FURTHER INTUITION:

Like the optimal hyperplane, only observations that violate the margin determine \mathcal{H}

A large t allows for many violations, hence many observations factor into the fit

A small t means only a few observations do

Hence, t calibrates a bias/variance trade-off, as expected

In practice, t or λ gets selected via cross-validation

SVMs: TUNING PARAMETER

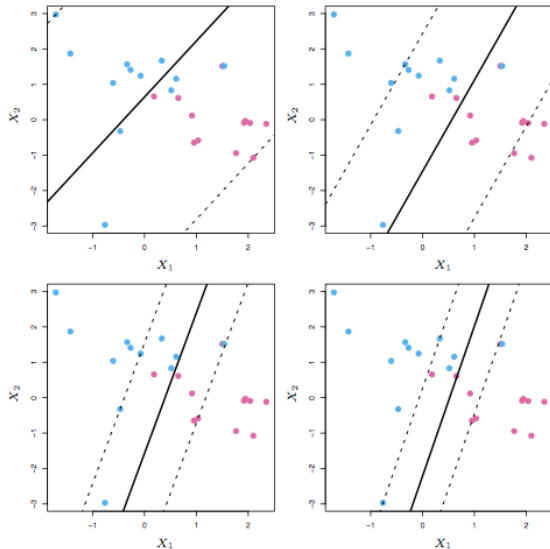


Figure 9.7 in ISL

SUPPORT VECTOR CLASSIFIER IN R

A common package to use is `e1071`

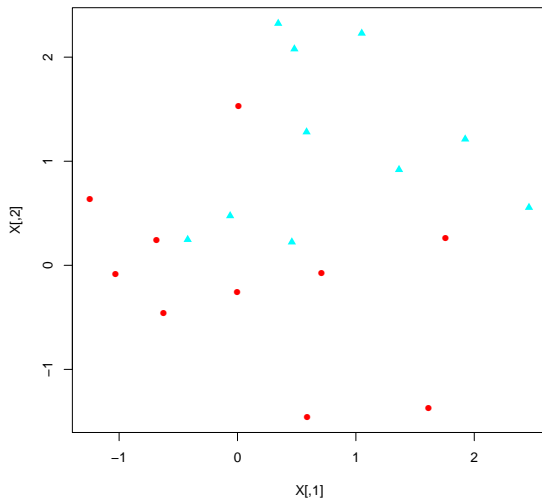
```
X = matrix(rnorm(20*2),ncol=2)
Y = c(rep(-1,10),rep(1,10))
X[Y == 1,] = X[Y == 1,] + 1
```

```
col = rep(0,length(Y))
col[Y == -1] = rainbow(2)[1]
col[Y == 1] = rainbow(2)[2]
```

```
pch = rep(0,length(Y))
pch[Y == -1] = 16
pch[Y == 1] = 17
```

```
plot(X,col=col,pch=pch)
```

SUPPORT VECTOR CLASSIFIER IN \mathbb{R}



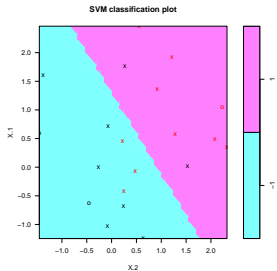
SUPPORT VECTOR CLASSIFIER IN R

```
library(e1071)
dat  =data.frame(X=X, Y=as.factor(Y))
svmfit=svm(Y~., data=dat, kernel="linear", cost=cost)
```

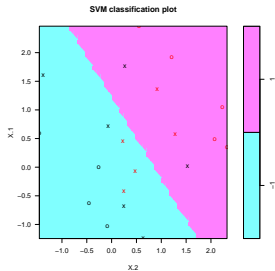
IMPORTANT: Their definition of cost is the **Lagrangian** version, which we defined as λ

Hence, a **small cost** means a large t and a **wider** margin

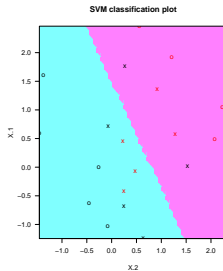
SUPPORT VECTOR CLASSIFIER IN \mathbb{R}^2



cost = .1



cost = 1

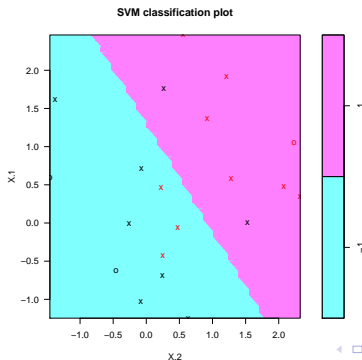


cost = 10

SUPPORT VECTOR CLASSIFIER IN R

```
tune.out = tune(svm,Y~.,data=dat,kernel="linear",  
               ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))  
best.model = tune.out$best.model
```

Note that `best.model` is an `svm` object:



NEXT TIME: KERNEL METHODS

INTUITION: Many methods have linear decision boundaries

We know that sometimes this isn't sufficient to represent data

EXAMPLE: Sometimes we need to include a polynomial effect or a log transform in multiple regression

Sometimes, a **linear** boundary, but in a different space makes all the difference..