Statistics 452: Statistical Learning and Prediction

Chapter 9: Support Vector Machines

Brad McNeney

Overview

- The context is a binary classification problem.
- We will discuss two classification rules that incrementally lead to the support vector machine.

1. Maximal margin classifier: If there exists a linear boundary in the space of explanatory variables that separates the classes in the training data, use this boundary as a classifier.
    - If one boundary exists, there will be many. Choose one that maximizes the "margin", a minimum distance between predictors and the boundary.
    - Simple classifier, but requires that a linear boundary exist.
2. Support vector classifier: Weaken the requirement of complete separation to a linear boundary that **best** separates the classes.
    - Better, but a linear boundary may not be the best choice.

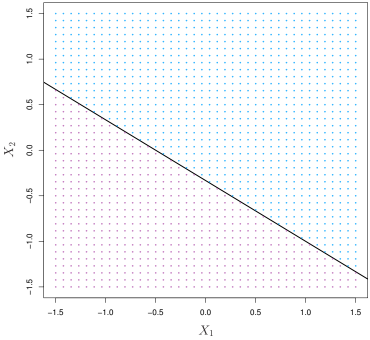
- Support vector machine: Weaken the requirement of a linear boundary, and add a trick to prevent computations from becoming prohibitive.

Linear Boundaries

- Linear boundaries can be represented as a "hyperplane", which is a $p-1$-dimensional subspace of p dimensions specified by a constraint

$$f(X) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p = 0.$$

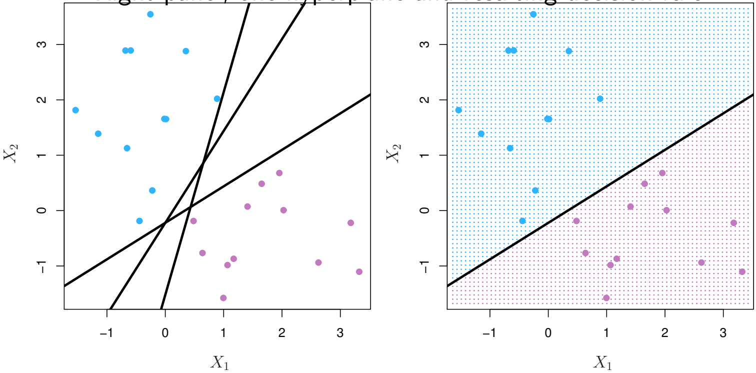
  - Classify as blue class if $f(X) > 0$ and red if $f(X) < 0$.



(Text, Figure 9.1)

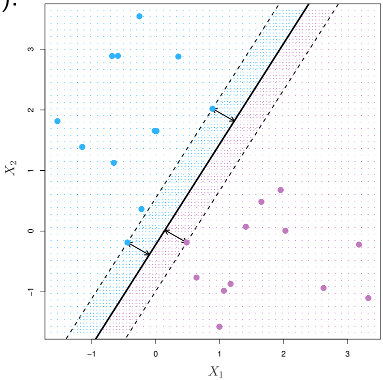
Possible Hyperplanes

- When a separating hyperplane exists, there are many.
- Text, Figure 9.2:
  - Left panel, three separating hyperplanes
  - Right panel, one hyperplane and resulting decision rule



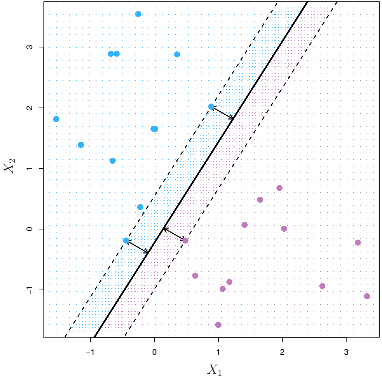
Maximal Margin Classifier

- The maximal margin hyperplane is the separating hyperplane farthest from the training observations.
  - For each possible hyperplane, find the margin: The minimum of perpendicular distances between each point and the hyperplane.
  - Choose the hyperplane with the largest margin as the boundary (Fig. 9.3).



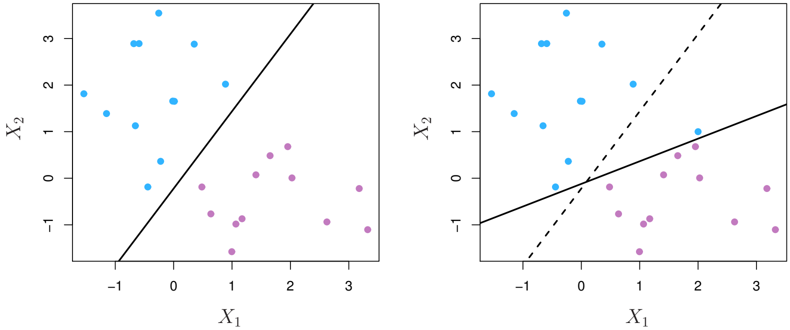
Support Vectors

- The support vectors are the points (p-dimensional vectors) that determine the maximal margin hyperplane.
  - Support in the sense that if they move, so does the hyperplane.
  - Support vectors in the diagram are two blue points and one red with perpendicular distances indicated on the figure.
  - The classifier depends on the data through the support vectors **only**.



Sensitivity to Support Vectors

- Classifier depends on data through the support vectors **only**.
- Adding/deleting a support vector can drastically change the maximal margin hyperplane.
- Example maximal margin hyperplane before (left panel) and after (right panel) adding a new support vector (text, Fig 9.5):

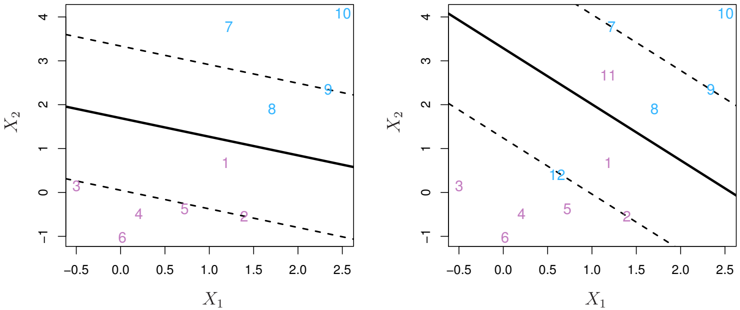


A Budget of Margin Violations

- Notice that the margin in right panel of the previous example is very small.
- We might instead seek a larger margin, but allow some points to be on the wrong side.
  - Allow vector i to be a distance ϵ_i on the wrong side of the margin, as long as $\sum_{i=1}^n \epsilon_i \leq C$, for some budget C.
  - By allowing points on the wrong side of the margin, and even wrong side of the hyperplane, we can accommodate the case where the vectors are not separable by a hyperplane.
- This is the support vector classifier.
  - The budget C is a tuning parameter.
- The support vectors are those on the margin or on the wrong side of the margin.
  - One can show that only these support vectors affect the choice of hyperplane and hence classification rule.

Illustration of Support Vector Classifier

- In the left panel of the following Figure (Fig 9.6), one observation from each class is on the wrong side of the margin, but not the hyperplane.
- In the right panel, two more points are added that are also on the wrong side of the hyperplane.
- The support vectors are those on the margin or on the wrong side of the margin.



Choosing the budget C

- Bias-variance tradeoff:
  - Small C means we require narrow margins, and are potentially over-fitting the data. Should have low bias but high variance
  - Large C means we allow wide margins and are not fitting the data as aggresively. Should have higher bias, but lower variance.
- Use cross-validation to select C.

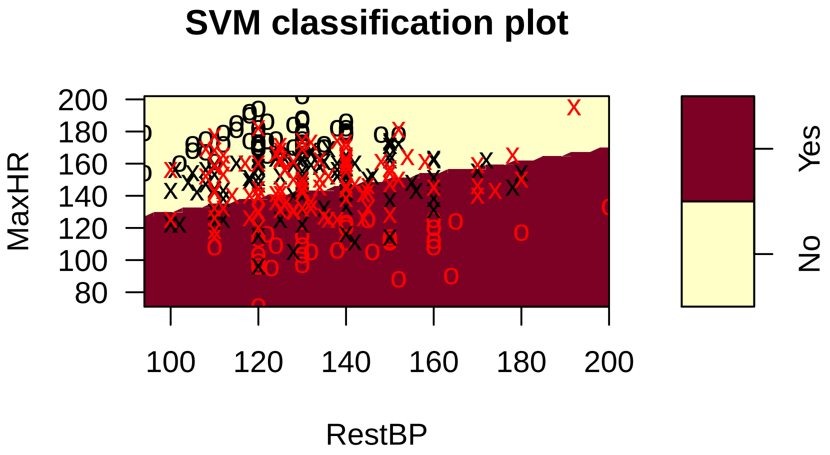
Example Support Vector Classifier for the Heart Data

- The `svm()` function from the `e1071` package fits support vector machines.
  - Set argument `kernel=linear` for the support vector classifier.

```
uu <- url("http://faculty.marshall.usc.edu/gareth-james/ISL/Heart.csv")
Heart <- read.csv(uu,row.names=1)
Heart <- na.omit(Heart)
library(dplyr)
Heart <- select(Heart,AHD,MaxHR,RestBP)
library(e1071)
svc.heart <- svm(AHD ~ .,type="C-classification",cost=1,
                 data=Heart,kernel="linear")
```

Visualize the Classifier

```r
plot(svc.heart,Heart)
```



- (Zoom for better view) Classes are colour-coded; O means on right side and X means wrong side of margin.

Choose the cost by CV

- The `tune()` function does the CV over a user-supplied grid of costs.

```r
set.seed(123)
tune.heart <- tune(svm,AHD ~ .,data=Heart,kernel="linear",
                   ranges=list(cost=c(10^{-3:3})))
summary(tune.heart)$performances
```

```
##    cost     error dispersion
## 1 1e-03 0.4610345 0.08855707
## 2 1e-02 0.3265517 0.06894093
## 3 1e-01 0.3095402 0.06761133
## 4 1e+00 0.3029885 0.07597847
## 5 1e+01 0.3064368 0.07361748
## 6 1e+02 0.3064368 0.07361748
## 7 1e+03 0.3031034 0.07949254
```

- According to these CV results, we should use cost 1.

Non-linear Decision Boundaries

- By expanding the features to include polynomial terms, the support vector classifier will have non-linear decision boundaries in the original feature space.
  - In the expanded feature space, say $X_1, X_1^2, \ldots, X_p, X_p^2$, the boundary will be a curve of the form

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_1^2 + \ldots + \beta_{2p-1} X_p + \beta_p X_p^2 = 0.$$

- We could consider higher-order polynomials, but eventually we would have so many features that computation time would become a problem.
- The support vector machine uses a computational trick to allow non-linear decision boundaries without prohibitive computation.
  - To describe the trick, recast the computation for the support vector classifier in terms of "kernels".

Support Vector Classifier *via* Kernels

- It turns out (e.g., ESL, Section 12.3) that the classification rule for a point x depends on the sign of

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x, x_i)$$

where

- $K(x, x_i) = \sum_{j=1}^{p} x_j x_{ij}$ and
- β_0 and the α_i's depend on the data only through the $n(n-1)/2$ values of $K(x_i, x_{i'})$.
- The function $K(\cdot, \cdot)$ is called the linear kernel.

Support Vector Machine

- We can extend the approach by choosing other kernel functions $K(\cdot, \cdot)$.
  - In general, kernels are functions that measure **similarity** between two feature vectors (high value for similar vectors, low value for dissimilar vectors).
- Examples of alternative kernels include
  - the polynomial kernel, for given degree d:

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d$$

  - and the radial kernel for given $\gamma > 0$:

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$

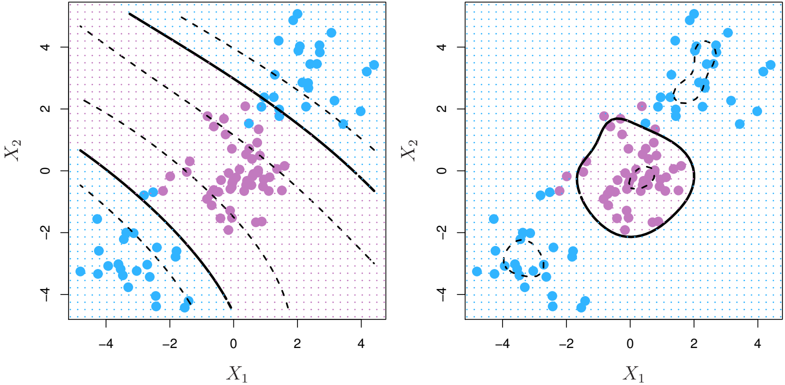
- The extension of the support vector classifier to non-linear boundaries, with an expanded feature space and computations done *via* kernels is called the support vector machine.

Computation for the SVM

- Key point: No matter how large the dimension of the expanded feature space, the computations only rely on the data through $n(n-1)/2$ values of $K(x_i, x_{i'})$.

Example Decision Boundaries for Polynomial and Radial Kernels

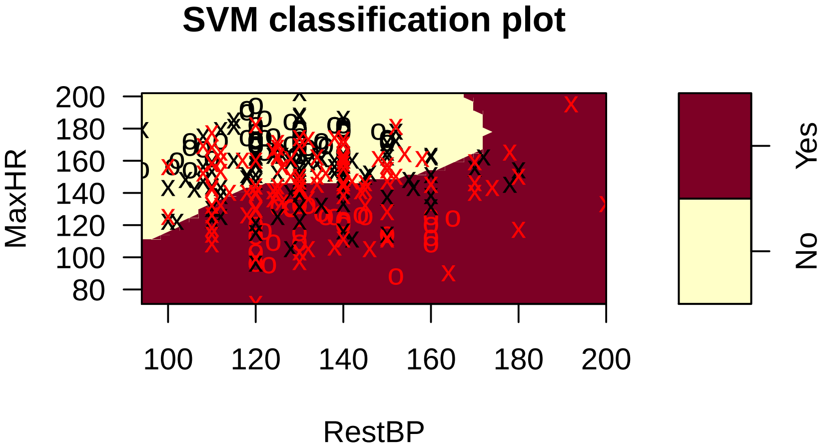
- Figure 9.9. Left panel is from the polynomial kernel with $d = 3$; right panel is from a radial kernel (value of γ not given).



SVM for Heart Data

- Use a radial kernel.

```r
svm.heart <- svm(AHD ~ .,type="C-classification",cost=1,
                 data=Heart,kernel="radial",gamma=1/2)
plot(svm.heart,Heart)
```



Choose the Cost and γ by CV

- The `tune()` function can do the CV over a grid of costs **and** other tuning parameters such as γ for the radial kernel.

```
set.seed(123)
tune2.heart <- tune(svm,AHD ~ .,data=Heart,kernel="radial",
                   ranges=list(cost=c(10^{-3:3}),
                               gamma=c(.5,1:4)))
perf<-summary(tune2.heart)$performances
perf[which.min(perf$error),]
```

```
##    cost gamma     error dispersion
## 10  0.1     1 0.2929885 0.08907935
```

Classifications

- Here we predict the training data.
  - In your weekly exercises you will split the Heart data into training and test sets and will predict the test set.

```
preds <- predict(svm.heart)
table(preds,Heart$AHD)
```

```
##
## preds  No Yes
##   No  127  47
##   Yes  33  90
```

- About a 27% misclassification.
  - Looks pretty bad, but we're using just MaxHR and RestBP at this point.

SVMs with More than Two Classes

- No natural extension of the separating hyperplanes idea to multiple classes.
- Instead, use one-versus-one or one-versus-all classification.

One-versus-One Classification

- If there are K classes, there are $K(K-1)/2$ pairwise comparisons.
- Can train $K(K-1)/2$ SVMs.
- For a test observation, get $K(K-1)/2$ predictions.
  - Each prediction is like a game in a tournament.
  - The final classification for the test observation is the winner of the tournament; i.e., the class that the test observation was most frequently assigned to.

One-verus-All Classification

- For each class, train a SVM to classify that class versus all others pooled together.
  - End up with K classifiers.
- The classification of a test observation is class with highest confidence; i.e., on the right side of the decision boundary, and farthest from the boundary.

Example: Gene Expression Data

- Four tumor types ($K = 4$) measured on 63 training and 20 test tumors.
- For each tumor, there are 2308 measurements of "gene expression".
  - With this many features relative to the number of observations, non-linear kernels may provide **too much** flexibility – use linear.
- Classify tumor type based on expression measurements.
- `svm()` uses one-vs-one classification

```r
library(ISLR)
data(Khan)
# help(Khan)
dat <- data.frame(y=as.factor(Khan$ytrain),
                  Khan$xtrain)
head(names(dat))
```

```
## [1] "y"  "X1" "X2" "X3" "X4" "X5"
```

```r
library(e1071)
fit <- svm(y~., data=dat,kernel="linear",cost=10)
pp <- predict(fit,newdata=data.frame(Khan$xtest))
table(pp,Khan$ytest)
```

```
##
## pp   1 2 3 4
##    1 3 0 0 0
##    2 0 6 2 0
##    3 0 0 4 0
##    4 0 0 0 5
```

```r
2/20
```

```
## [1] 0.1
```

The SVM and Logistic Regression

Loss + Penalty Formulation of the Support Vector Classifier

- Let $\mathbf{X}$ and $\mathbf{y}$ denote the matrix of X's and vector of y's, respectively and

  $$f(X;\beta) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$$

  be the decision boundary.

- One can show that the criterion function that the support vector classifier minimizes to estimate f is of the form

  $$\min_{\beta}\{L(\mathbf{X}, \mathbf{y}, \beta) + \lambda P(\beta)\}$$

  where $L()$ is the so-called hinge loss function

  $$L(\mathbf{X}, \mathbf{y}, \beta) = \sum_{i=1}^{n} \max[0, 1 - y_i f(x_i;\beta)],$$

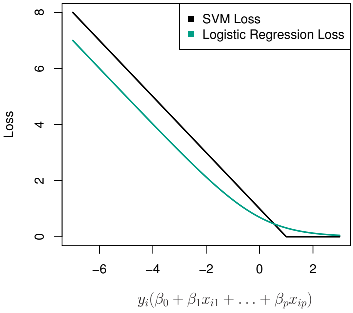
  $P(\beta) = \sum_{j=1}^{p} \beta_j^2$ is the ℓ_2 penalty function and λ is a tuning parameter.

Hinge *versus* Logistic Loss

- The hinge loss function is similar to the logistic loss function

$$\sum_{i=1}^{n} \log(1 + e^{-y_i f(x_i;\beta)}),$$

used in logistic regression.
  - Recall that logistic regression is fit by maximum likelihood.
  - ML amounts to minimizing a negative-log-likelihood loss.
  - Loss, as written above, is for outcomes coded as $-1/1$.



Support Vector Classifier/Machine *versus* Logistic Regression

- Conclude that SV Classifier is similar to logistic regression penalized with an ℓ_2 penalty.
- Can further argue that the SV Machine is similar to ℓ_2-penalized logistic regression with non-linear functions of the predictors.