STA 4241 Lecture, Week 5

Overview of what we will cover

- Support vector classifier
- Introducing nonlinear terms into a support vector classifier
- Representation of support vector classifier in terms of inner products
- Support vector machines
  - Linear kernel
  - Nonlinear kernels
- Application to stock market data
- Heart disease example

Overview of last week's lecture

- Recall that last week we learned about the maximal margin classifier
- This classifier finds a hyperplane that separates the data into two classes
- Frequently, it is not possible to separate data in this way
  - Even if it's possible, it may not be the best classifier
- We needed to relax this classifier to allow observations to be misclassified

Support vector classifier

- The support vector classifier relaxes the maximal margin classifier to allow observations on the wrong side of the margin or even the wrong side of the hyperplane
- Observations on the wrong side of the hyperplane are misclassified
  - Inevitable in most settings
- This will help address two issues
  - Lack of separability in the data
  - Overfitting and sensitivity to single observations

Support vector classifier

- Now we will solve the following optimization problem

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n,M}{\text{Maximize}} \; M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1$$

$$Y_i(\beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip}) > M(1 - \epsilon_i) \; \forall \; i = 1, \ldots, n$$

$$\epsilon_i \geq 0, \; \sum_{i=1}^{n} \epsilon_i \leq C$$

Support vector classifier

- This looks similar to the maximal margin classifier
- The main difference is the introduction of ϵ_i
  - Referred to as slack variables
- ϵ_i allow for individual observations to fall on either the wrong side of the margin or the wrong side of the hyperplane
- ϵ_i provides us information about where observation i lies relative to the hyperplane and margin

Support vector classifier

- If $\epsilon_i = 0$ then observation i falls on the correct side of the boundary
- If $\epsilon_i > 0$ then observation i falls on the wrong side of the margin
- If $\epsilon_i > 1$ then observation i falls on the wrong side of the hyperplane
- The maximal margin classifier effectively requires that every observation has $\epsilon_i = 0$
- The parameter C allows us to relax this restriction and allow for violations of the margin or hyperplane

Support vector classifier

- C is a tuning parameter
  - Determines the extent that we allow the margin to be violated
- $C = 0$ allows for no violations and we obtain the maximal margin classifier
- As C increases, we become more tolerant of violations, and therefore the margin widens

Support vector classifier

- As with most tuning parameters we have seen, this tuning parameter controls the bias-variance trade-off of our classifier
  - Similar to K from K-nearest neighbors
- A small value of C leads to a small margin, which is highly fit to the data
  - low bias, high variance
- A large value of C allows for a wide margin and more violations of the margin
  - Potentially high bias, low variance

Support vector classifier

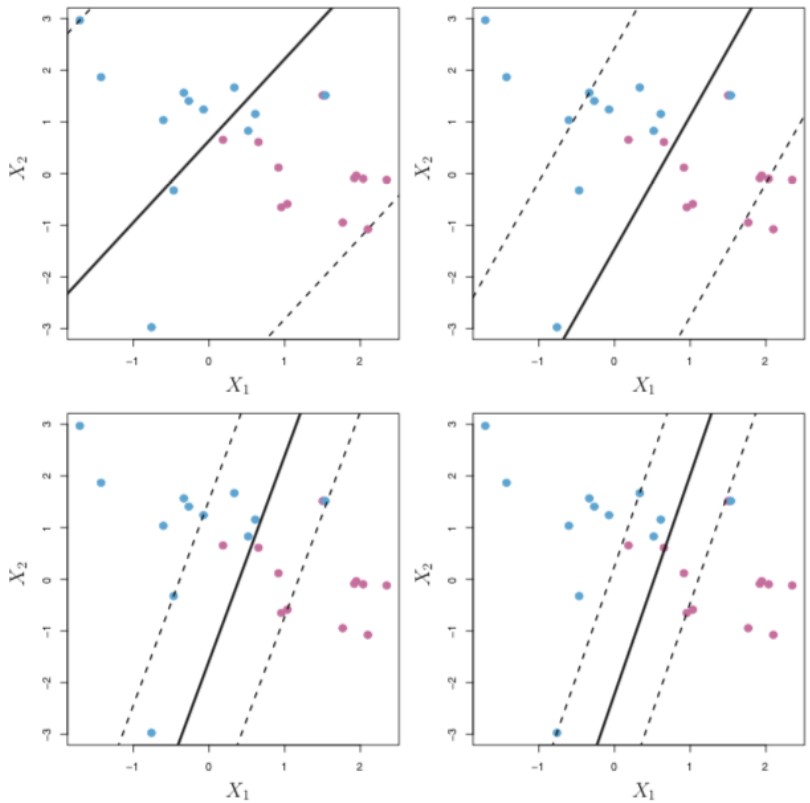
- One key fact that helps explain this bias-variance trade-off is that only points that are on or inside the margin affect the fit of the classifier
- If I move a point outside of the margin, it won't affect the model fit, unless it is moved to the inside of the margin
- Points on or inside the margin are called the support vectors and they affect the model fit

Support vector classifier

- A small value of C leads to a small margin and therefore very few points fall inside the margin
  - Very few support vectors
  - Model is fit based on very few observations
    - Similar to using a small value of K in KNN
  - High variance
- A large value of C leads to a wide margin with many support vectors
  - Low variance, but potentially biased

Support vector classifier

- Below are the hyperplanes given by four values of C in decreasing order from top-left to bottom-right
- The margin gets smaller, and the number of support vectors decreases



James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). An introduction to statistical learning. New York: springer.

Support vector classifier

- The value of C is very important for the performance of the hyperplane at separating test data points
- This parameter is generally chosen using a tool called cross-validation
- Cross-validation is a general tool that can be used to choose tuning parameters for nearly all of the approaches we have seen in this class and we will learn about it in the next couple of weeks

Support vector classifier

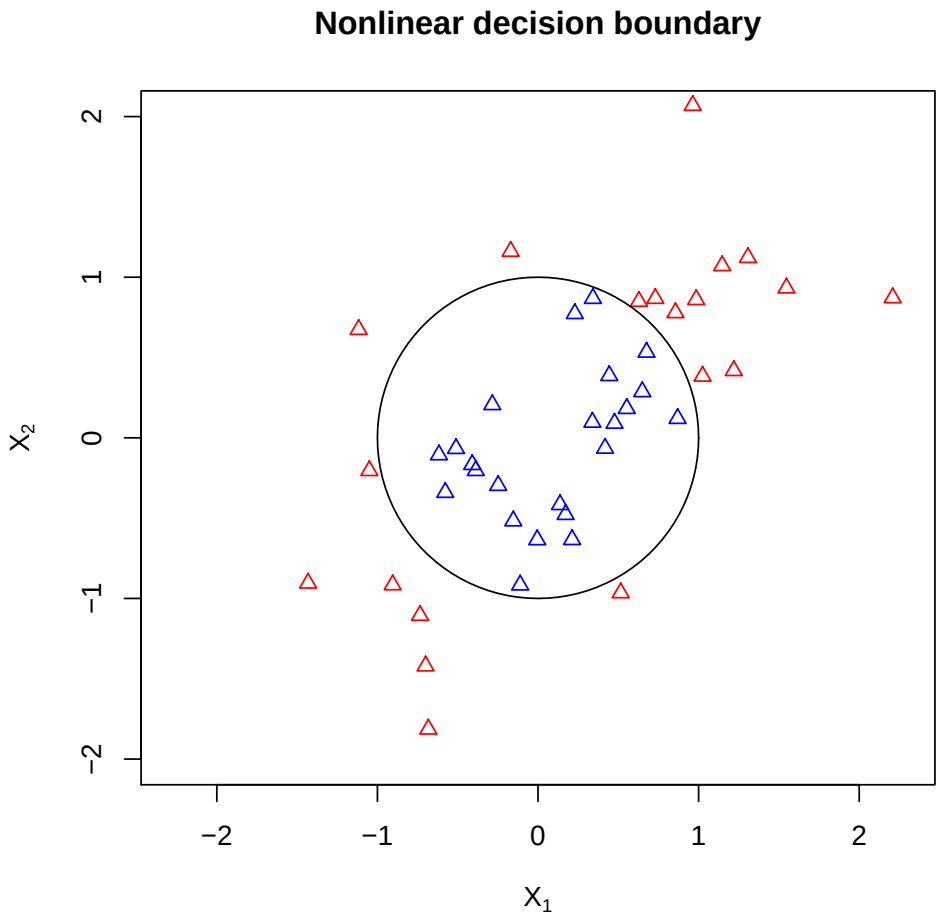
- This leads to a linear classifier
  - Linear in the covariates, $\boldsymbol{X}$
- Observations are classified based on their value of

$$\beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip}$$

- What if the true decision rule is nonlinear?

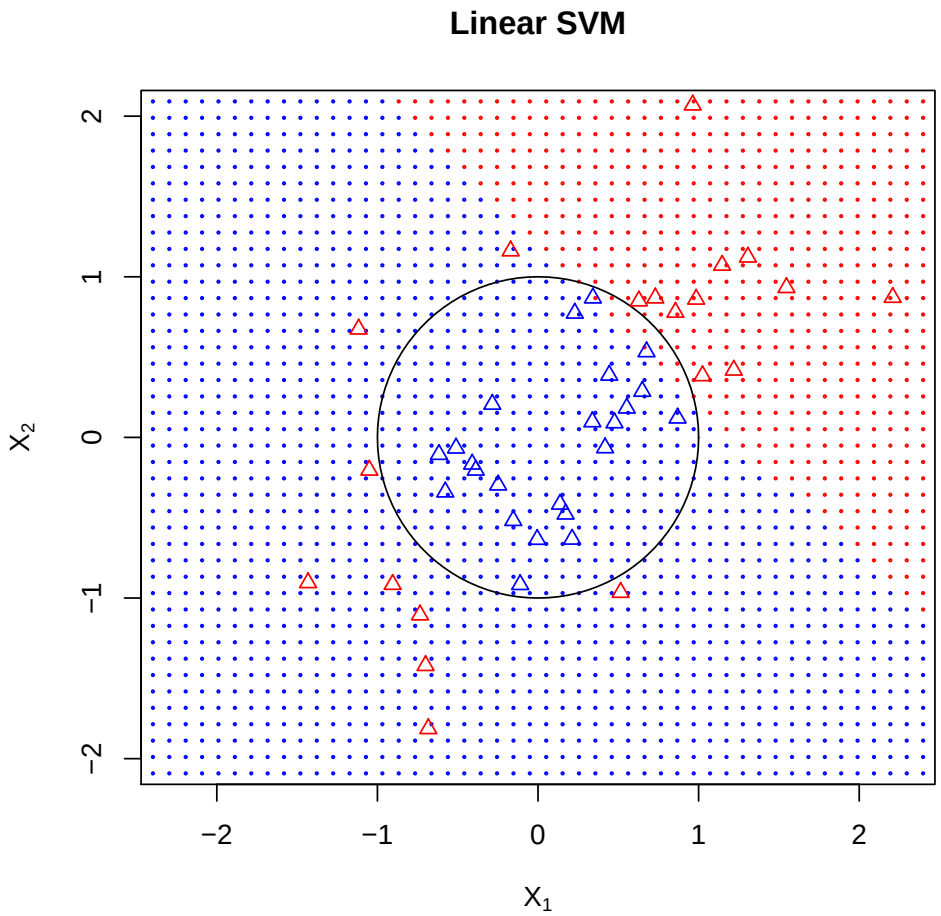
Nonlinear classifier

- How well will the support vector classifier work here?
  - Black line shows optimal decision boundary
- Not clear what the best hyperplane would be



Nonlinear classifier

- We see that the linear classifier misclassifies many of the red points in the lower left portion of the plot
- Clearly, the linear fit is inadequate here



Nonlinear classifier

- We've addressed nonlinearity in a number of ways so far in class
- Some approaches are 'local' in the sense that they only use data points that are close with respect to the covariates when making predictions
  - KNN
- Local approaches don't make strong parametric assumptions about the relationship between the covariates and outcome
  - Naturally allow for nonlinearity

Nonlinear classifier

- We've also seen that you can expand your covariate space
- Include additional functions of the covariates
  - Polynomial terms
  - $\log X, e^X, b(X), X_1X_2$, etc.
- We must choose which additional functions of the covariates to include, and how many to include
- Then we can just run our linear procedures on the expanded space

Nonlinear classifier

- We will now see that both of these methods to inducing nonlinearity apply to support vector classifiers
  - Expand covariate space
  - Use a more local approach
- We will explore both here, but generally people do not use the expanded covariate space for support vector classifiers
- The other approaches to nonlinearity with support vector classifiers are very elegant and easy to use
  - Support vector machines

Expanded covariate space

- Instead of the linear classifier:

$$\beta_0 + \sum_{j=1}^{p} \beta_j X_{ij}$$

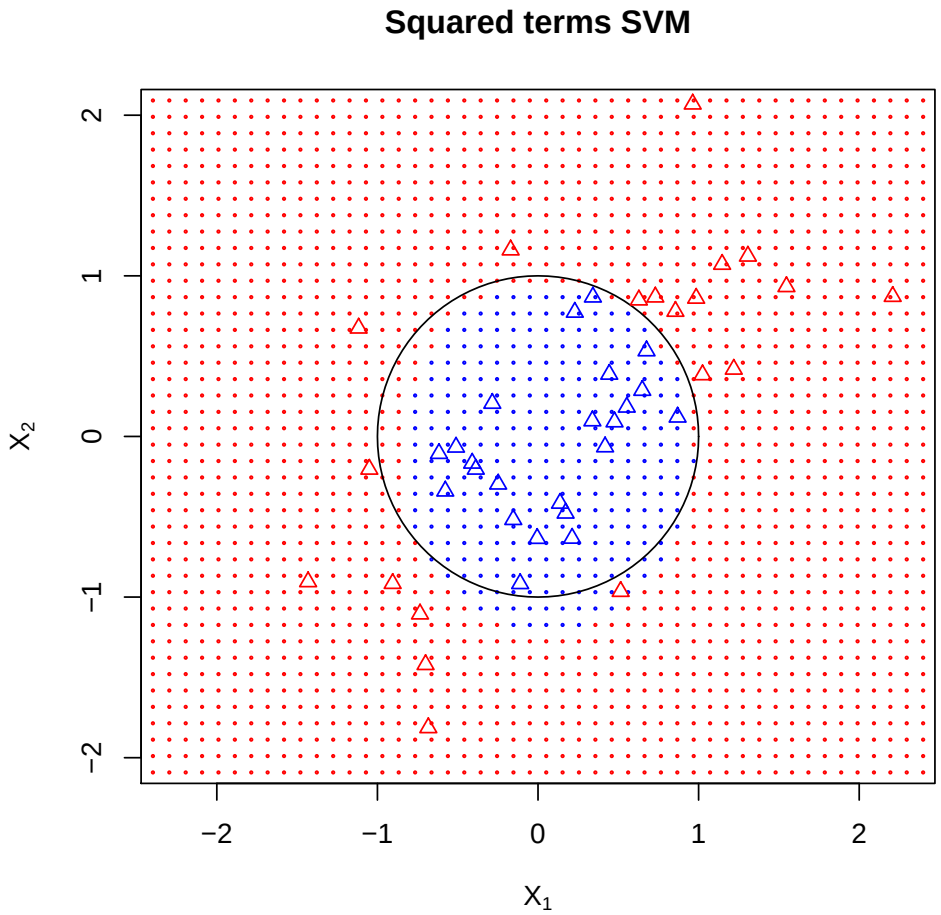
  we could use one that includes squared terms

$$\beta_0 + \sum_{j=1}^{p} \beta_{j1} X_{ij} + \sum_{j=1}^{p} \beta_{j2} X_{ij}^2$$

- We've seen in other cases that this can model relatively nonlinear data

Nonlinear classifier

- Let's apply the classifier with squared terms to the previous data
- Substantial improvement in fit



Nonlinear classifier

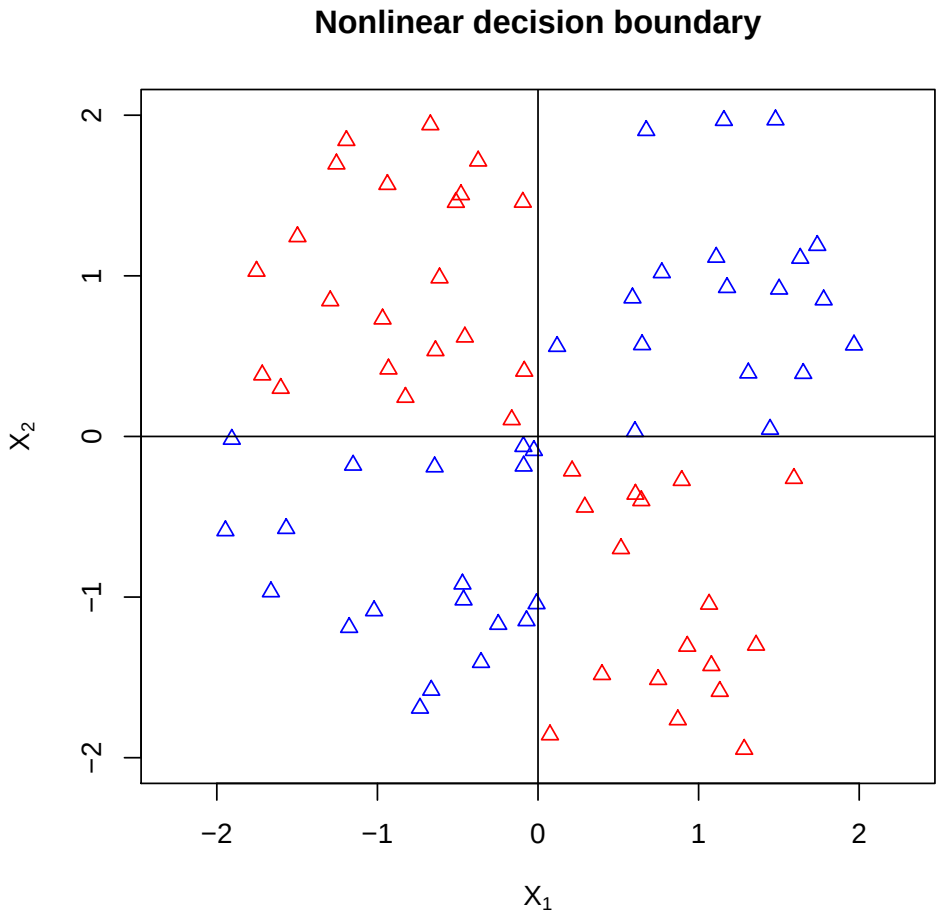
- In this case, it seemed to work relatively well
- The squared terms did a good job of approximating the nonlinear decision boundary
- The circle is defined by the equation

$$X_1^2 + X_2^2 = 1$$

so the quadratic terms capture the true relationship between the covariates and outcome

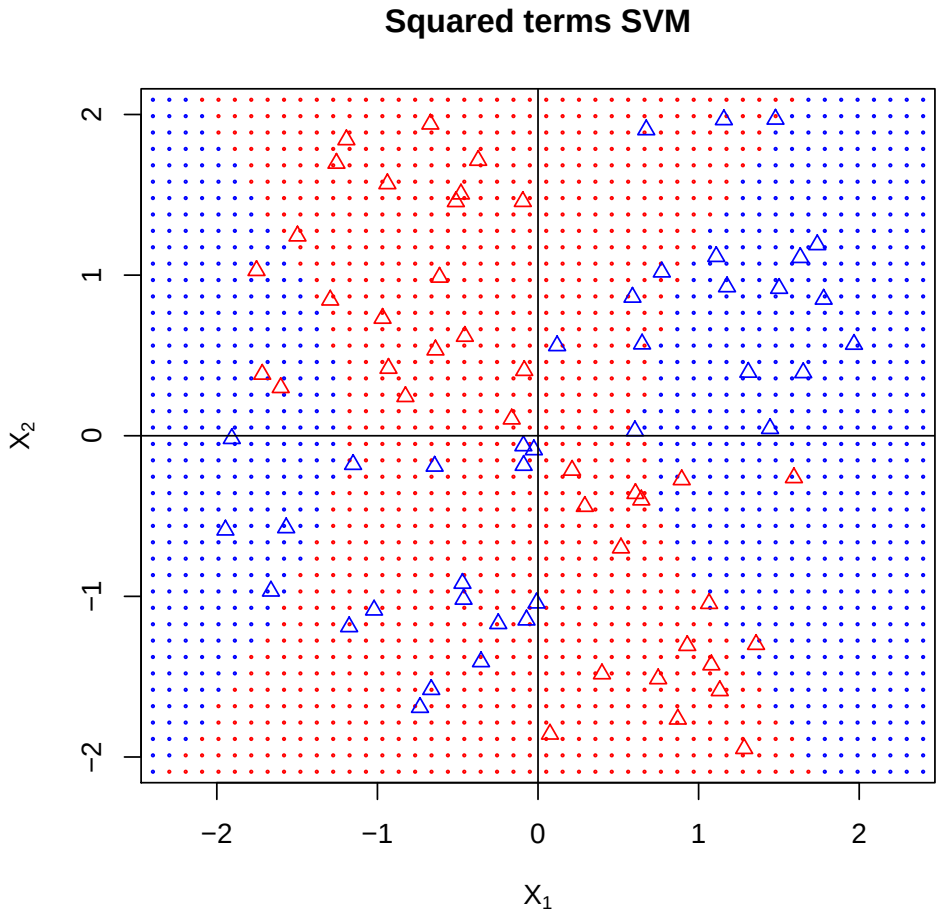
Nonlinear classifier

- What if the data doesn't fit the quadratic terms so well?
- In this case, the data are separated by the quadrants



Nonlinear classifier

- The quadratic approach is insufficient for modeling these data



Nonlinear classifier

- The extended covariate space approach only works well if you have the right functions included in your expanded space
  - Sometimes called basis functions
- Not always straightforward to choose these functions
- There are methods to choose among the different basis functions, but this can be computationally tedious, and still may not perform particularly well

Nonlinear classifier

- Further, even if you have the correct basis functions, it might require a very large number to model the true relationship between the covariates and outcome
- This can become computationally burdensome or even infeasible
- Ideally we can use an approach that adapts well to many different outcome/covariate associations
  - Support vector machine

Nonlinear classifier

- Before we discuss the support vector machine, we need to delve into the technical details of the linear support vector classifier
- This will illuminate how we can readily introduce nonlinearity into our models
- It turns out that the support vector classifier depends on the inner products between observations
  - Not the individual observations themselves

Inner products

- The inner product between two $p-$dimensional vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as
$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{j=1}^{p} a_j b_j$$
- In the context of our problem, the inner product between two observations is
$$\langle X_i, X_{i'} \rangle = \sum_{j=1}^{p} X_{ij} X_{i'j}$$
- $\binom{n}{2}$ such inner products in our data

Inner products

- You can think of inner products as a measure of similarity between two vectors
- If two data points are similar, then when one is large, the other tends to be large as well and the product becomes large
- The less similar two vectors are, the smaller the inner product becomes

Inner products

- As an example, let $\boldsymbol{a} = (-5, -4, \ldots, 4, 5)$
- If we take the inner product of $\boldsymbol{a}$ with itself, we get

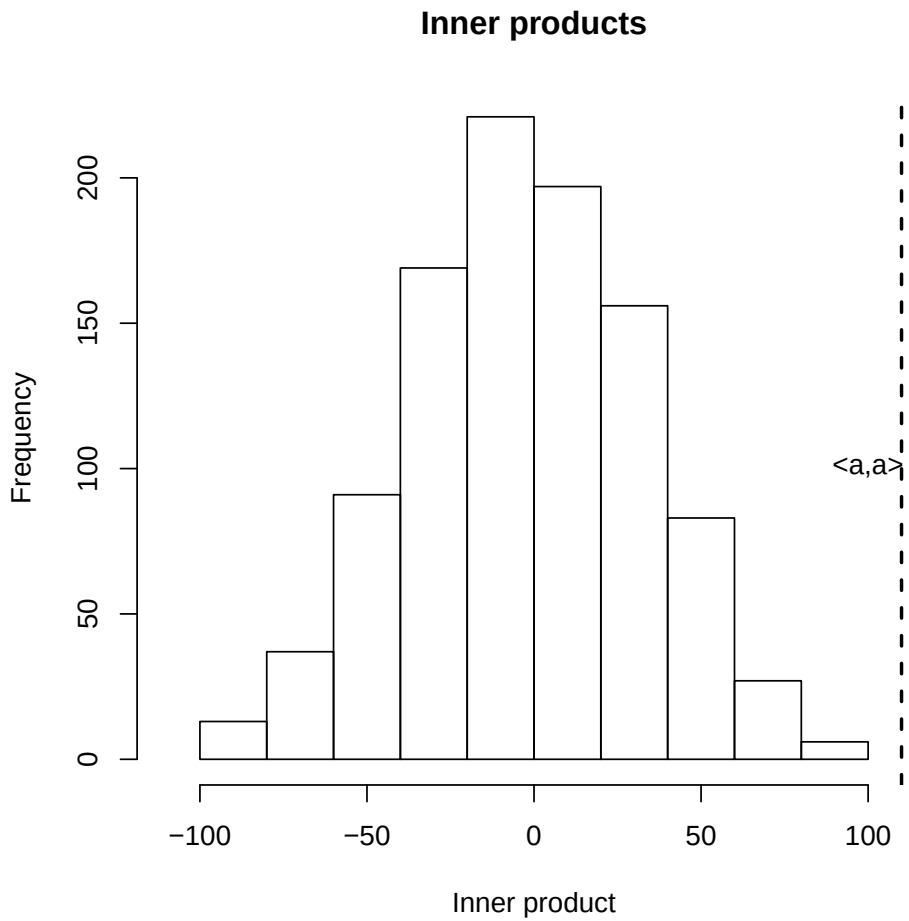
$$\langle \boldsymbol{a}, \boldsymbol{a} \rangle = 110$$

- What if instead, we randomly re-order the elements in $\boldsymbol{a}$ according to a permutation $\boldsymbol{\pi}$, then calculate

$$\langle \boldsymbol{a}, \boldsymbol{a}_{\boldsymbol{\pi}} \rangle$$

Inner products

- If we do this 1000 times we get the following histogram of inner products



Inner products

- In our data, the inner product between two points will be high if they have similar covariate values
  - Lie in similar spaces of the $p-$dimensional space defined by our covariates
  - 'Close' to each other
- This is important, because we will soon see that we can replace this notion of similarity with other notions that allow for nonlinearity

Support vector classifier

- The solution to the support vector classifier optimization problem has an interesting property
- If our goal is classifying a new observation with covariates defined by $\boldsymbol{X}$, the support vector classifier ends up taking the form

$$f(X) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle \boldsymbol{X}, \boldsymbol{X}_i \rangle$$

- α_i are parameters that are estimated from the data

Support vector classifier

- You might be asking how we went from $p + 1$ parameters denoted by $(\beta_0, \beta_1, \ldots, \beta_p)$ to n parameters
- There is a correspondence between α_i and our original $\boldsymbol{\beta}$ parameters
- We will work with the α_i for now, because this representation will prove useful for allowing nonlinearity in the classifier

Support vector classifier

$$f(X) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle \boldsymbol{X}, \boldsymbol{X}_i \rangle$$

- The parameters α_i are estimated using the $\binom{n}{2}$ inner products $\langle \boldsymbol{X}_i, \boldsymbol{X}_{i'} \rangle$
- If we want to make a classification for the new observation $\boldsymbol{X}$ we only need to calculate the n inner products $\langle \boldsymbol{X}, \boldsymbol{X}_i \rangle$

Support vector classifier

- Recall that only the support vectors affected the SVM model fit
- It turns out that $\alpha_i = 0$ for any observation that is not a support vector!
- If $\mathcal{S}$ is the set of support vectors, then the classification can be written as

$$f(X) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle \boldsymbol{X}, \boldsymbol{X}_i \rangle$$

Support vector classifier

- So we only need to calculate the inner product with the support vectors to make a new classification
- The only thing that matters for classification is how similar a new observation is to the support vectors
- This is nice computationally
  - Only have to calculate a small number of inner products
- Extends easily to nonlinear situations

Kernels

- Now we will talk about how these approaches can easily accommodate nonlinearity in the data
- The inner product described above is not the only way to measure similarity
- We will use the notion of a kernel to define similarity between two observations

$$K(\boldsymbol{X}_i, \boldsymbol{X}_{i'})$$

where $K(\cdot, \cdot)$ is the kernel function

Kernels

- Everywhere where we used the inner product before, we can now replace with a generalization of the inner product given by $K(\boldsymbol{X}_i, \boldsymbol{X}_{i'})$
- Now the decision boundary will be of the form

$$f(X) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(\boldsymbol{X}, \boldsymbol{X}_{i'})$$

- The $\binom{n}{2}$ kernels $K(\boldsymbol{X}_i, \boldsymbol{X}_{i'})$ are used to estimate the model parameters

Kernels

- The inner product described above is a kernel

$$K(\boldsymbol{X}_i, \boldsymbol{X}_{i'}) = \sum_{j=1}^{p} X_{ij} X_{i'j}$$

- This is called the linear kernel
  - Linear in the covariates
  - Quantifies similarity using Pearson correlation
- There are many choices of kernel that are nonlinear in the covariates

Kernels

- Another kernel related to ideas that we have seen before is the polynomial kernel

$$K(\boldsymbol{X}_i, \boldsymbol{X}_{i'}) = (1 + \sum_{j=1}^{p} X_{ij} X_{i'j})^d$$

- d is the degree of the polynomial and determines the amount of nonlinearity allowed
- Closely related to fitting a linear support vector classifier on the extended covariate space that includes higher order polynomials

Kernels

- The radial or Gaussian kernel is widely used

$$K(\boldsymbol{X}_i, \boldsymbol{X}_{i'}) = \exp\left(-\gamma \sum_{j=1}^{p} (X_{ij} - X_{i'j})^2 \right)$$

- Unlike the polynomial kernel, this does not have a corresponding expanded covariate space we could implement
- In fact, the expanded covariate space implied by this model is infinite dimensional!
  - Shows how we can't simply always fit a support vector classifier on an expanded space

Kernels

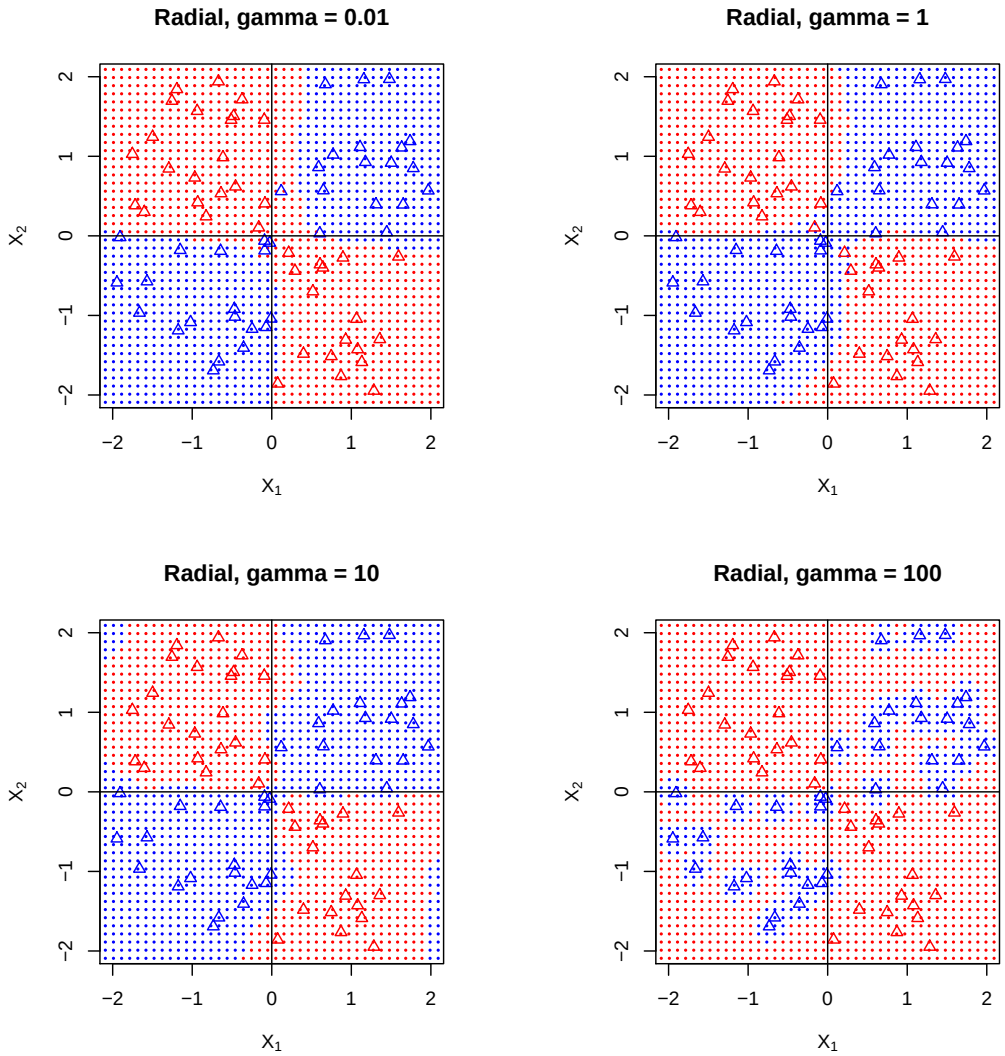
- $\gamma > 0$ is a tuning parameter that must be chosen
- The radial kernel is a local approach since

$$\exp\left(-\gamma \sum_{j=1}^{p} (X_{ij} - X_{i'j})^2 \right)$$

  will be very small for two observations that are far apart
- Only observations close to $\boldsymbol{X}$ will play a major role in classification
- γ controls how localized the kernel is

Kernels

- I fit an SVM with $\gamma \in \{0.01, 1, 10, 100\}$

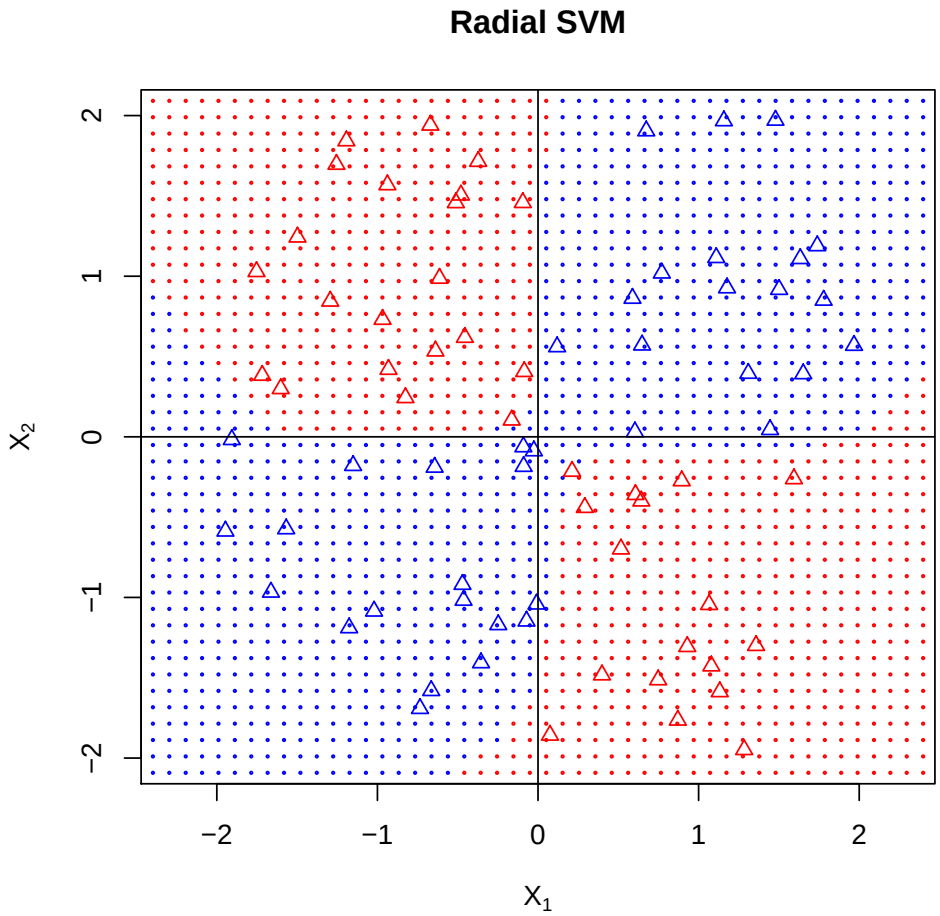


Kernels

- Larger values of γ lead to much more localized behavior
- Do we actually believe the fit when $\gamma = 100$?
  - Highly overfit to the training data
- Smaller values of γ lead to very good decision boundaries that approximate the true boundary quite well
- Generally the choice of γ presents a bias-variance trade-off

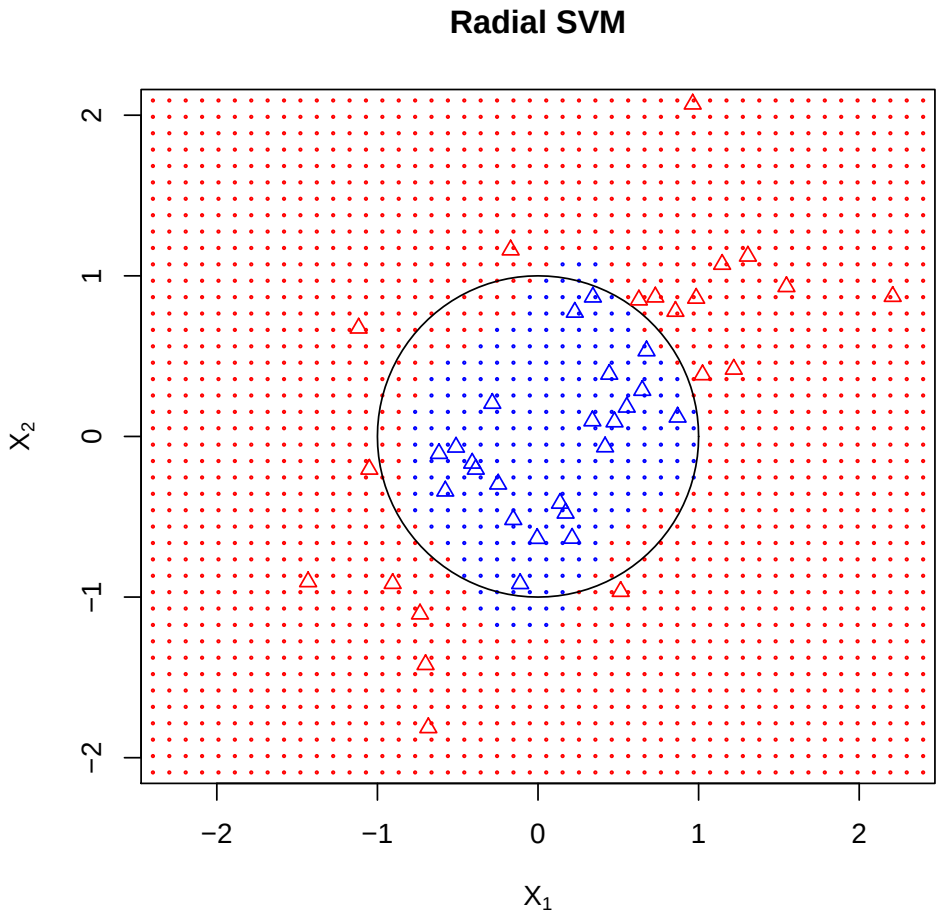
Support vector machines

- Suppose I choose γ using cross-validation
  - Will learn how to do this next week
- Very nice fit, that doesn't appear overfit



Support vector machines

- let's try the same procedure on the circle data from earlier
- The radial kernel seems promising!

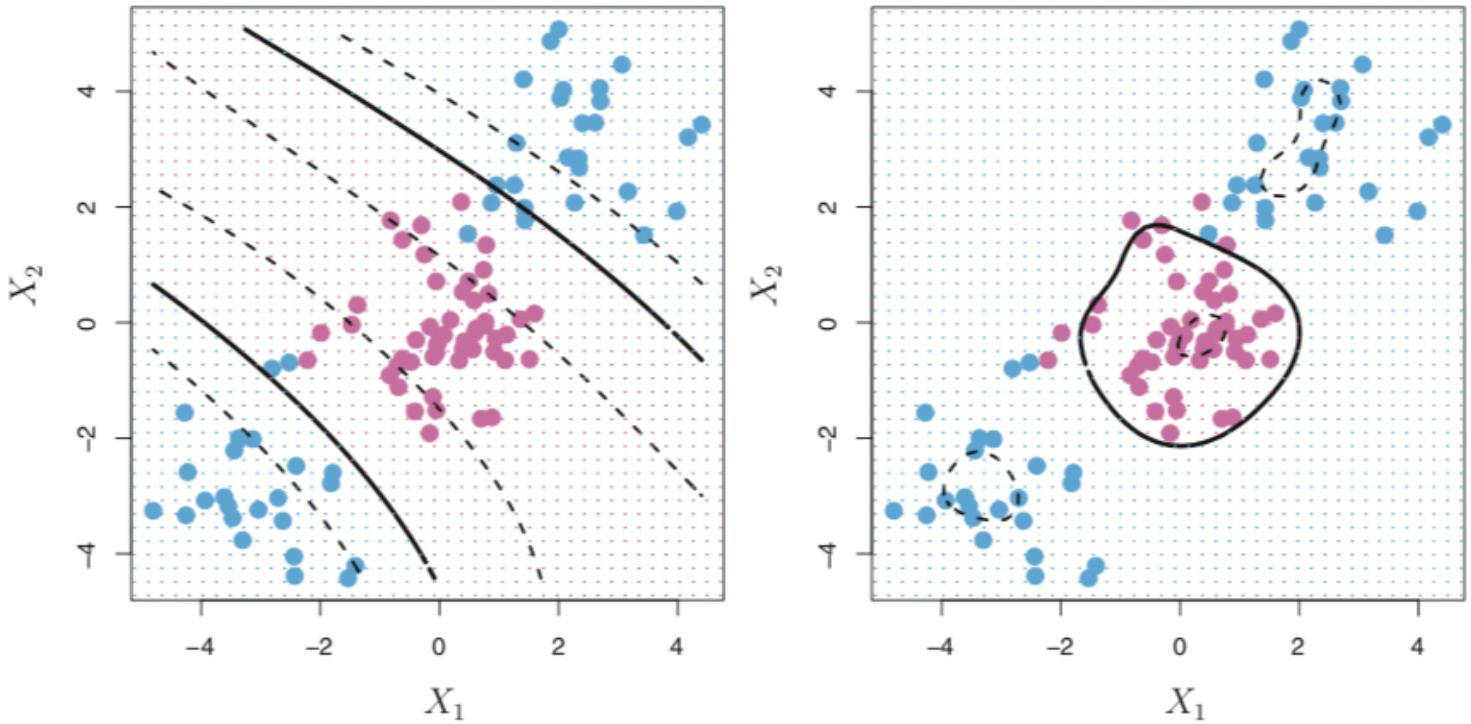


Support vector machines

- Working with kernels and SVMs still requires a decision of which kernel to use
- Different kernels can perform better or worse on a particular data set
- Flexible kernels such as the radial kernel are popular because they work in many settings
  - Assuming tuning parameters are properly chosen

Support vector machines

- Here is an example where the polynomial and radial kernels give somewhat different boundaries
  - Both seem appropriate for this data
  - Need to evaluate on test data to see which is better



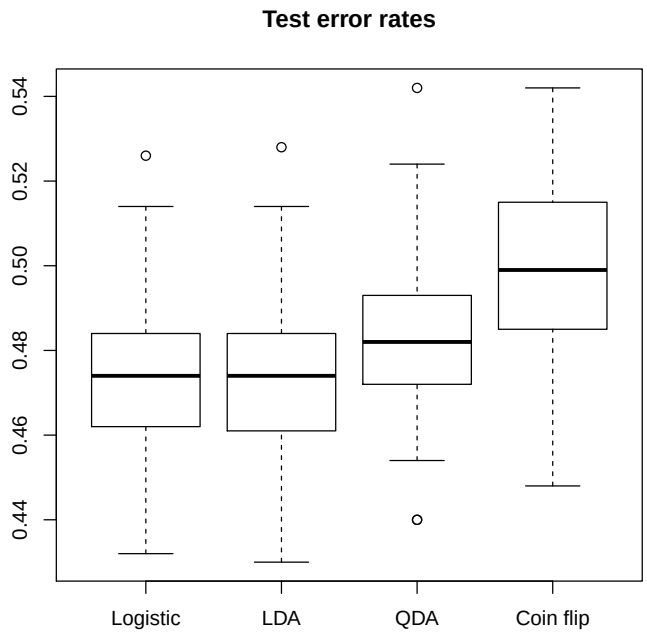
James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). An introduction to statistical learning. New York: springer.

Stock market data

- Let's try applying support vector machines to the stock market data from the previous lecture
- As a reminder we are trying to predict whether the market will go up or down on a given day
- Will use previous days returns, volume, and year as predictors to include in the SVM
- We have seen previously that predicting this outcome is hard!
  - Can barely beat random guessing

Stock market data

- We will again randomly generate 500 points to be testing points
- Will repeat this process a large number of times
- The previous estimators achieved the following performance

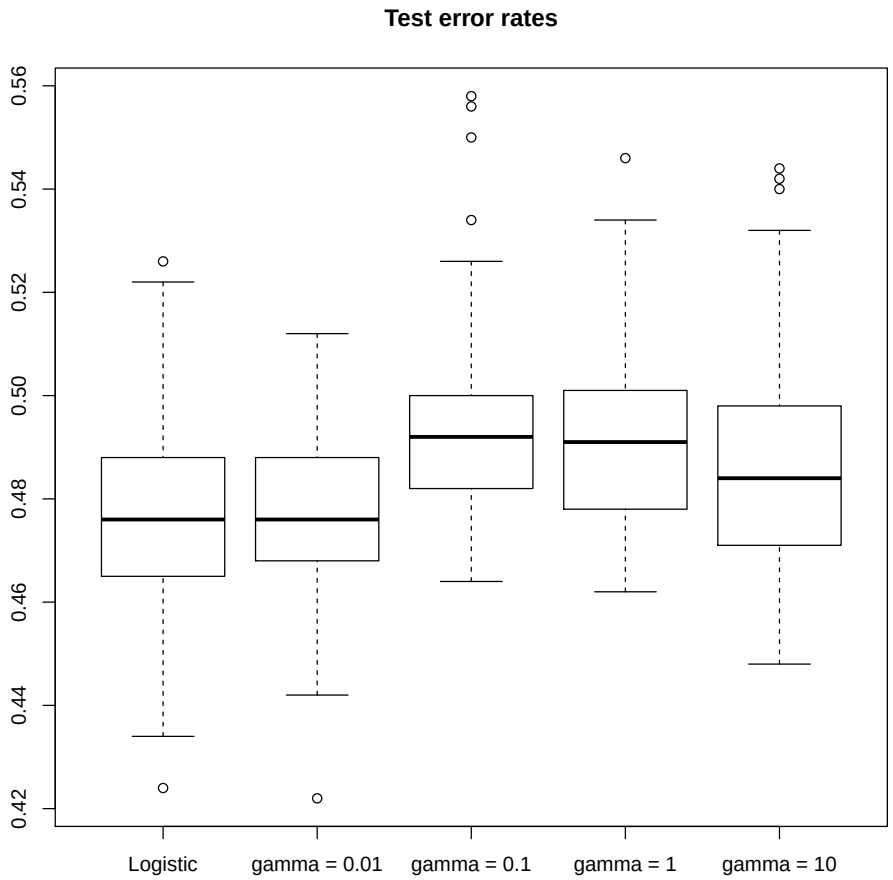


Stock market data

- Maybe we can do better with our new, fancy tools
- We will try to fit SVMs with both polynomial and radial kernels
- We will assess sensitivity to the tuning parameters
  - Degree of polynomial
  - γ for radial basis

Stock market data - radial SVM

- We tried $\gamma \in \{0.01, 0.1, 1, 10\}$
- We chose the other tuning parameter, C, by cross-validation
  - All SVMs have this tuning parameter



Stock market data - radial SVM

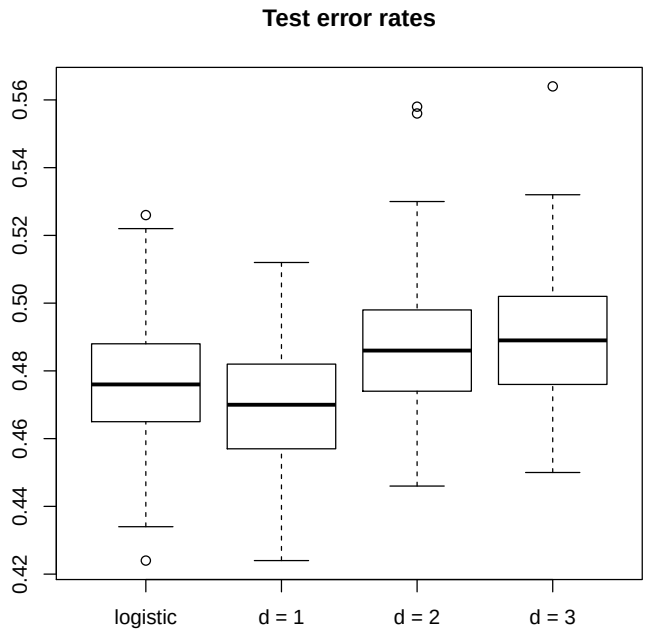
- Larger values of γ lead to worse performance
- Let's look at the average training vs. testing error rates for each γ

	testing	training
Logistic	0.477	0.444
SVM, $\gamma = 0.01$	0.477	0.424
SVM, $\gamma = 0.1$	0.494	0.366
SVM, $\gamma = 1$	0.492	0.289
SVM, $\gamma = 10$	0.487	0.124

- Massive amounts of overfitting with larger γ

Stock market data - polynomial SVM

- We tried $d \in \{1, 2, 3\}$
- Again chose C by cross-validation



Stock market data - polynomial SVM

- Smaller values of d lead to better performance
- Let's look at the average training vs. testing error rates for each d

	testing	training
Logistic	0.477	0.444
SVM, $d = 1$	0.470	0.452
SVM, $d = 2$	0.487	0.420
SVM, $d = 3$	0.490	0.394

- Some evidence of overfitting with larger d
  - Not nearly as bad as with radial kernel

Stock market data - Summary

- Overall, SVMs were not able to greatly improve on LDA or Logistic regression, which were the best performers previously
- The linear kernel slightly outperformed logistic regression
- Allowing for nonlinearity didn't help performance and only led to overfitting
- Results are sensitive to the tuning parameters
  - We will learn an approach to choosing them next week

Heart disease example

- The textbook goes through an example where the outcome is a binary indicator of heart disease status
- The goal is to classify patients based on 13 predictors
- Compare LDA to SVM with a radial kernel
  - Assess training and testing error rates
  - Sensitivity to tuning parameter selection
- Utilize ROC curves to assess predictive performance

Heart disease example

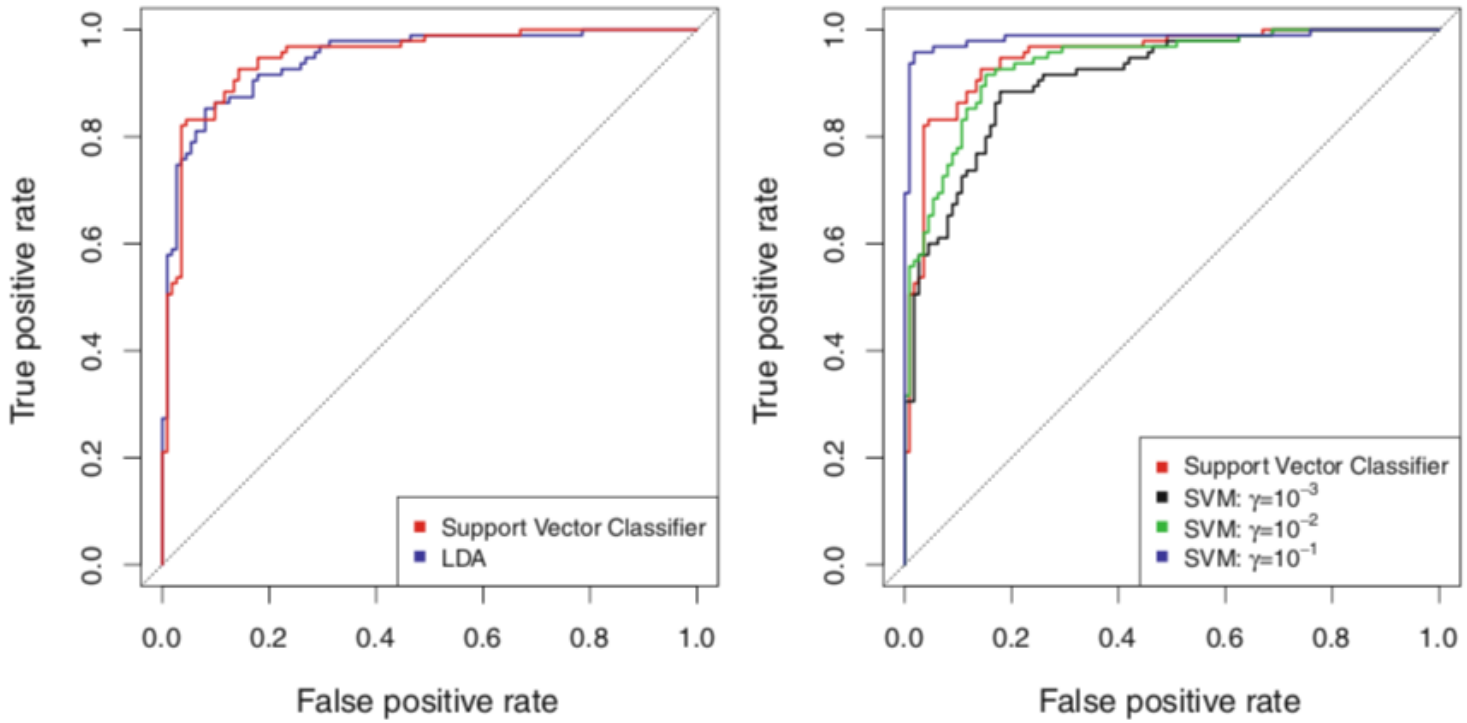
- ROC curves are commonly used to assess classification performance
- Suppose we have a classifier of the type

$$\widehat{Y} = \begin{cases} 1, & f(X) > t \\ -1, & f(X) \leq t \end{cases}$$

- An ROC curve plots true positive rate against false positive rate as a function of t
- An ideal ROC curve comes as close to the top left corner of the graph as possible

Heart disease example

- Here are ROC curves for the heart training data

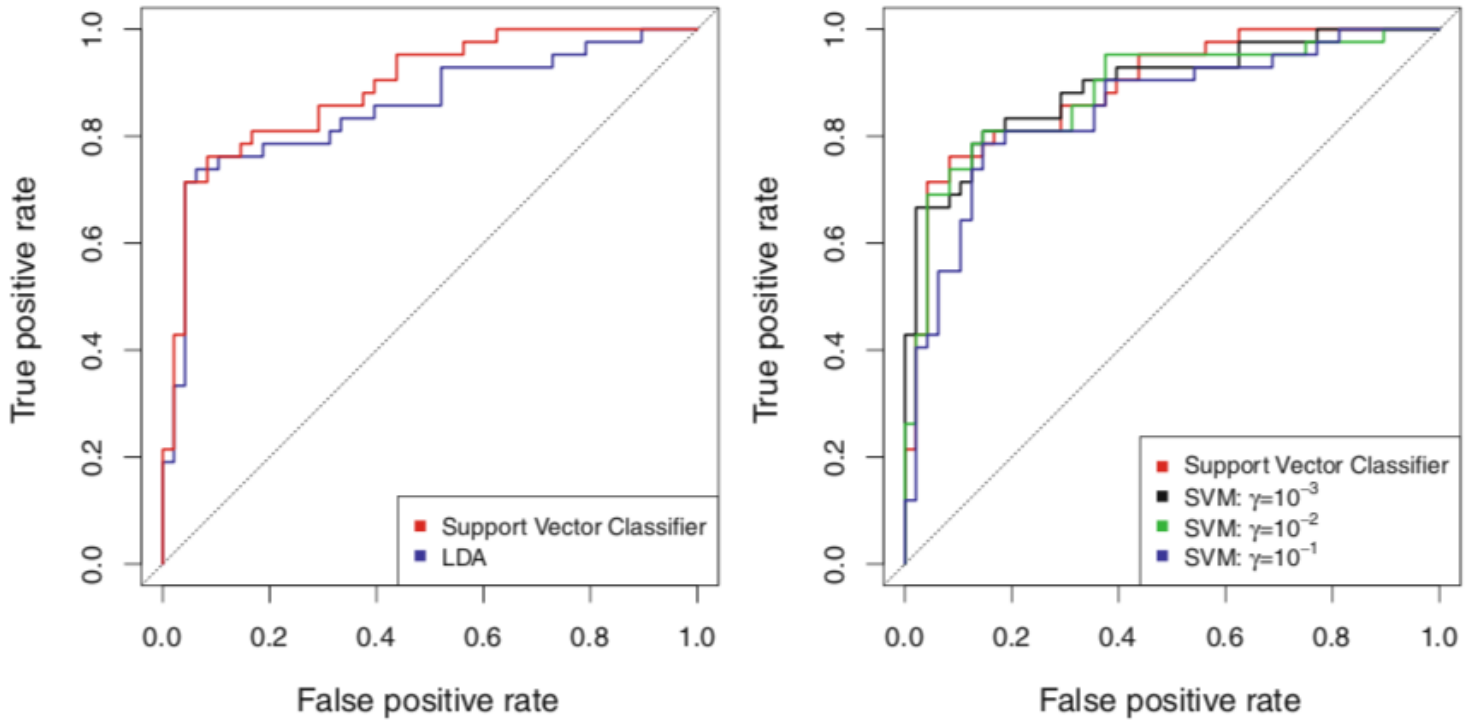


Heart disease example

- The linear SVM does slightly better than LDA
- We get extremely good performance of the radial SVM when we have large γ values
- It is possible that the true model is highly nonlinear and therefore we would expect large γ values to work best
- Nonetheless, we should be worried about overfitting

Heart disease example

- Now let's look at the same curves but on the testing data



Heart disease example

- The linear SVM again does better than LDA
  - Only minor difference
- Strong evidence of overfitting for large γ values
  - Bad testing performance compared to training performance
- Increasing γ does not improve the predictions on testing data
- Again we see how important it is to choose the tuning parameter correctly

SVM with more than two classes

- As we've seen, SVMs are a powerful tool for classification
- The main drawback is that SVMs are designed for binary classification problems
- What do we do if there are 3 or more outcome categories?
- It is not straightforward to extend the hyperplane idea to multiple classes

One versus one classification

- If there are K outcome classes, we can fit $\binom{K}{2}$ SVM classifiers
- For each classifier, keep track of which of the two classes a particular data point is assigned to
- Keep track of the number of times that each class is chosen by a classifier
- Use the class that has the largest tally at the end of this process

One versus all classification

- An alternative is to fit K SVMs
  - One for each class
- Each time classifying to either class k or to all classes except class k
- For each classifier we get a decision rule based on parameters $\beta_{0k}, \beta_{1k}, \ldots, \beta_{pk}$
- Choose the class with the largest value of

$$\beta_{0k} + \beta_{1k} X_1 + \cdots + \beta_{pk} X_k$$

One versus all classification

- These ideas are not unique to SVMs
- We can apply either the one versus one or one versus all principles to any binary classifier to create multiple category classifiers
- Not clear how well these will perform and they are not widely used
- This really shows the usefulness of LDA/QDA which automatically extend to more classes without problem