Machine Learning, Neural Networks and Deep Learning

Dr. B. Wilbertz[1]

[1]Trendiction S.A. / Talkwalker

1st October, 2021

Organization / InClass Competition

Slides and other material is available at

```
https://drive.google.com/drive/folders/1OG6vL_y8oxKwQOx6d_6W-ZtLQ81pgyRj
```

Kaggle InClass Competition

`https://www.kaggle.com/c/m2-proba-finance-2021`

- The ranking on this competition will contribute to your final grade for this course
- Invitation link for joining the competition:
  `https://www.kaggle.com/t/f1958ec06f7e440e983a69fd75a5ec52`
- **Important:** Choose your *name* or *student number* as team name! (Account name whatever you like).
- In case of problems:
  `https://www.kaggle.com/about/inclass/faqs`

Pre-Processing: Feature Scaling

Feature Scaling

For certain methods, scale of features vector $X_1, \ldots, X_p$ is important during training

- when using penalization, the coefficients of the classifier won't be penalized the same
- Lipschitz constant of the loss often depend on $\|X_j\|_2$: features with large scale slow down convergence

Scaling variants include:

- center, include an intercept, standardize
- min-max scaling
- binarize

Feature Scaling

Contiuous features

- Centering and standardization (or "whitening") of j-th feature

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}}$$

- Min-max scaling of j-th feature

$$\tilde{x}_{ij} = \frac{x_{ij} - x_{\min,j}}{x_{\max,j} - x_{\min,j}}$$

- Binning by replacing the variable with the center of the bins or dummies (quantiles)

Other forms of Feature Engineering

Categorial features

- Introduction of dummy variables for categorial features

```
> dummy <- model.matrix(Balance ~ Ethnicity, df)
> head(dummy)
     (Intercept)   EthnicityAsian   EthnicityCaucasian
1              1                0                    1
2              1                1                    0
3              1                1                    0
4              1                1                    0
5              1                0                    1
6              1                0                    1
```

Feature Engineering for text problems

Bag-of-word features

- Build a vocabulary $\mathcal{W}$ of your corpus
- Use one-hot encoding for every word (if a word is at position i_0 in $\mathcal{W}$, then it is represented by $(\delta_{ii_0})_i \in \mathbb{R}^{|\mathcal{W}|}$)
- Bag-of-word approach (optionally using TF/IDF as frequencies) sums up all one-hot encoded vectors of a text document

Embeddings

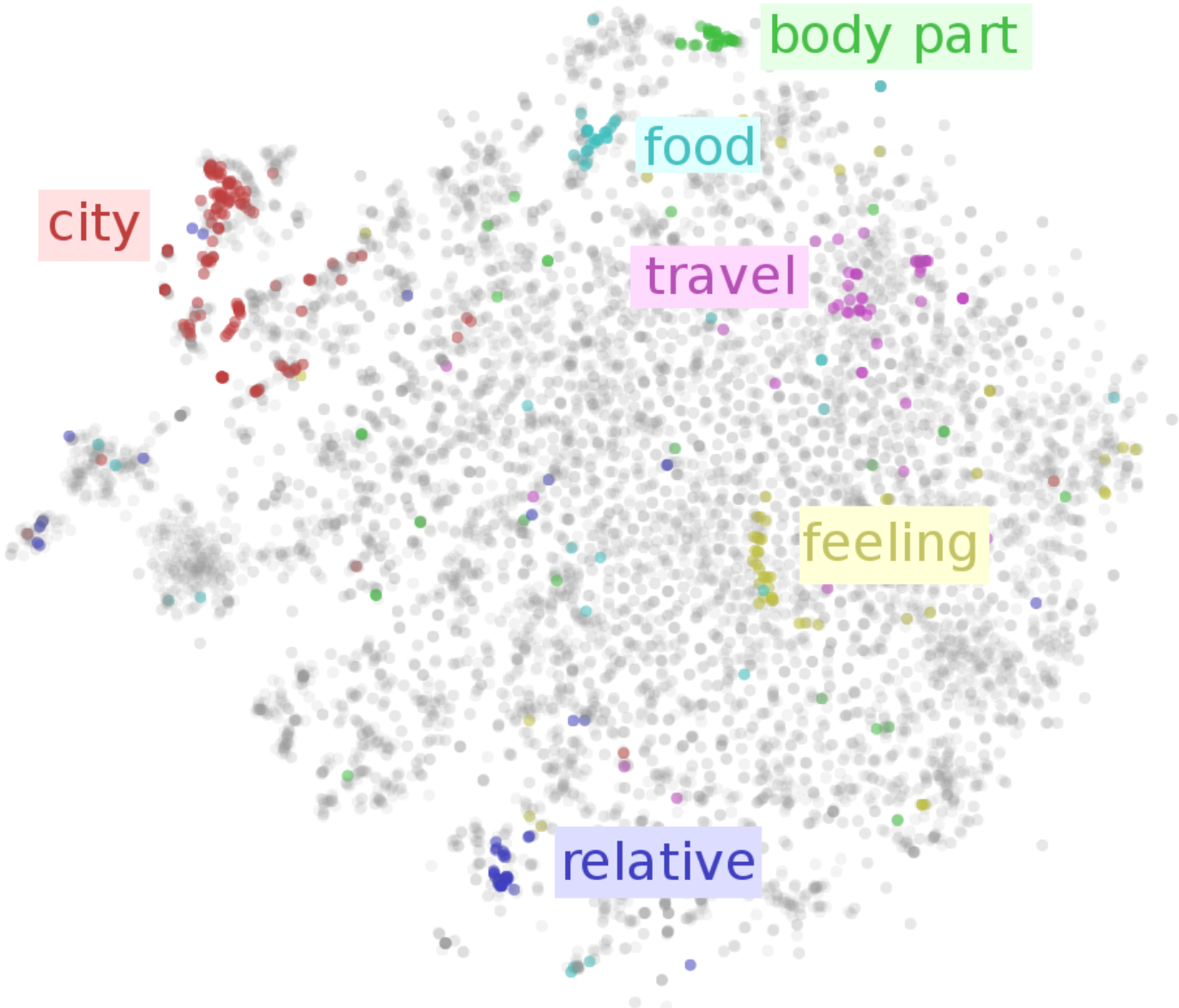
- Hashing of the words into a smaller space, i.e. $h : \mathcal{W} \hookrightarrow \mathbb{R}^d$, with $d \ll |\mathcal{W}|$ and a random projection h (hash function)
- Word embeddings
- Contextual embeddings

Word embeddings

Word embeddings: Idea

Word Embeddings (Word2Vec, GloVe, FastText, etc) are shallow neural networks, that are trained to reconstruct linguistic contexts of words: the network is shown a word, and must guess which words occurred in adjacent positions in an input text.
They build up a mapping $f_{\text{emb}} : \mathcal{W} \hookrightarrow \mathbb{R}^d$, where d typically has size 100 or 300.

Word embeddings

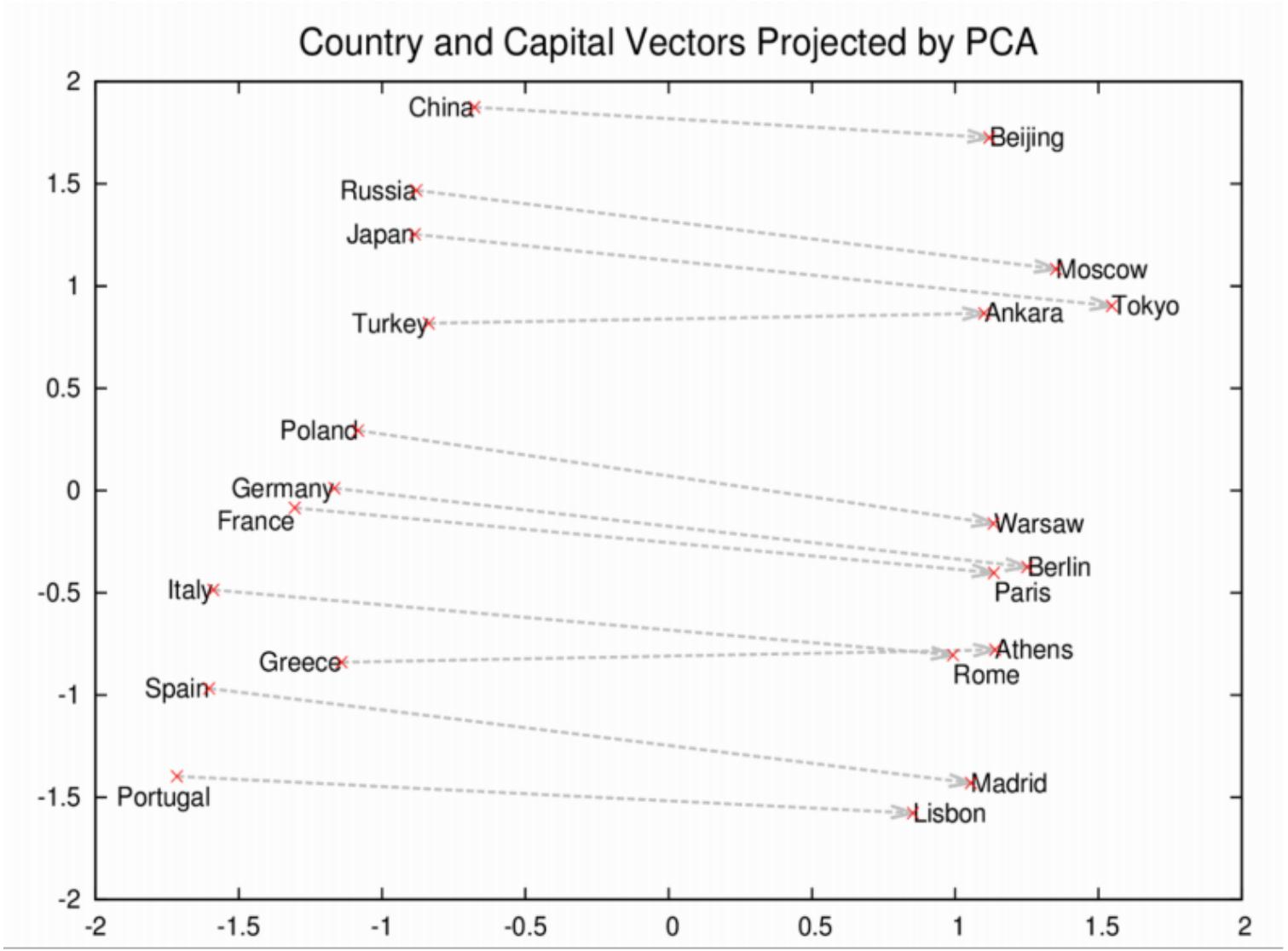


Word embeddings

One important chracteristic of this mapping is, that they map semantically close words into similar locations of the d-dimensional vector space.
This even allows some kind of arithmetic on words, i.e.

$$f_{\text{emb}}(\text{Berlin}) - f_{\text{emb}}(\text{Germany}) + f_{\text{emb}}(\text{Italy}) = f_{\text{emb}}(\text{Rome})$$

Word embeddings



Sentence and context-aware embeddings

Idea

- Train a deep neural network (Transformer architecure like BERT, roBERTa, SentenceBERT etc.) on predicting adjacent words or finding wrong words in a sentence.
- The embedding produced for the first input token turned out to represent very well the whole input text/sentence.
- Dimensionality of embeddings: 768 or 1024

Kaggle competition

Features `V1...V1024` in the dataset are those context-aware embeddings of the tweet text.

Linear Regression

Linear Regression

- Here our model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon,$$

- We estimate $\beta_0, \beta_1, \ldots, \beta_p$ as the values that minimize the sum of squared residuals

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_1 - \ldots - \beta_p x_p)^2 \end{aligned}$$

The values $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$ that minimize RSS are the multiple least squares regression coefficient estimates.

Baseball salary data (`hitter.csv`)

Example

Major League Baseball Data from the 1986 and 1987 seasons containing 320 observations and 20 variables. (Variables starting with `C` refer to the whole career)

- `AtBat` Number of times at bat in 1986
- `Hits` Number of hits in 1986
- `HmRun` Number of home runs in 1986
- `Runs` Number of runs in 1986
- `RBI` Number of runs batted in in 1986
- `Walks` Number of walks in 1986
- `Years` Number of years in the major leagues
- `League` A factor with levels A and N indicating player's league at the end of 1986
- `Division` A factor with levels E and W indicating player's division at the end of 1986
- `PutOuts` Number of put outs in 1986
- `Assists` Number of assists in 1986
- `Errors` Number of errors in 1986
- `Salary` 1987 annual salary on opening day in thousands of dollars
- `NewLeague` A factor with levels A and N indicating player's league at the beginning of 1987

Interpreting regression coefficients

Interpreting regression coefficients

- The ideal scenario is when the predictors are uncorrelated – a *balanced design*:
  - Each coefficient can be estimated and tested separately.
  - Interpretations such as "a unit change in X_j is associated with a β_j change in Y , while all the other variables stay fixed", are possible.
- Correlations amongst predictors cause problems:
  - The variance of all coefficients tends to increase, sometimes dramatically
  - Interpretations become hazardous – when X_j changes, everything else changes.
- *Claims of causality* should be avoided for observational data.

Why consider alternatives to least squares?

Shortcomings of ordinary least squares

- *Prediction Accuracy*: especially when $p > n$, to control the variance.
- *Model Interpretability*: By removing irrelevant features - that is, by setting the corresponding coefficient estimates to zero - we can obtain a model that is more easily interpreted. We will present some approaches for automatically performing *feature selection*.

Three classes of upgrades to least squares

Three classes of upgrades to least squares

- *Subset Selection.* We identify a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.
- *Shrinkage.* We fit a model involving all p predictors, but the estimated coefficients are shrunken towards zero relative to the least squares estimates. This shrinkage (also known as *regularization*) has the effect of reducing variance and can also perform variable selection.
- *Dimension Reduction.* We project the p predictors into a M-dimensional subspace, where $M < p$. This is achieved by computing M different *linear combinations*, or *projections*, of the variables. Then these M projections are used as predictors to fit a linear regression model by least squares.

Subset Selection

Best Subset Selection

1. Let $\mathcal{M}_0$ denote the *null model*, which contains no predictors. This model simply predicts the sample mean for each observation.
2. For $k = 1, 2, \ldots p$:
   a. Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   b. Pick the best among these $\binom{p}{k}$ models, and call it M_k . Here *best* is defined as having the smallest RSS, or largest R^2.
3. Select a single best model from among $M_0, \ldots, M_p$ using cross-validated prediction error, C_p(AIC), BIC, or adjusted R^2.

Stepwise Selection

Stepwise Selection

- For computational reasons, best subset selection cannot be applied with very large p.
- Best subset selection may also suffer from statistical problems when p is large: the larger the search space, the higher the chance of finding models that look good on the training data, even though they might not have any predictive power on future data.
- Thus an enormous search space can lead to *overfitting* and high variance of the coefficient estimates.
- For both of these reasons, stepwise methods, which explore a far more restricted set of models, are attractive alternatives to best subset selection.

Forward Stepwise Selection

Forward Stepwise Selection

1. Let $\mathcal{M}_0$ denote the *null* model, which contains no predictors.
2. For $k = 0, \ldots, p-1$:
   1. Consider all $p-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   2. Choose the *best* among these $p-k$ models, and call it $\mathcal{M}_{k+1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p(AIC), BIC, or adjusted R^2.

Credit data example

# Variables	Best subset	Forward stepwise
One	rating	rating
Two	rating, income	rating, income
Three	rating, income, student	rating, income, student
Four	cards, income	rating, income,
	student, limit	student, limit

The first four selected models for best subset selection and forward stepwise selection. The first three models are identical but the fourth models differ.

Backward Stepwise Selection

Backward Stepwise Selection

1. Let $\mathcal{M}_p$ denote the *full* model, which contains all p predictors.
2. For $k = p, p-1, \ldots, 1$:
   1. Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   2. Choose the *best* among these k models, and call it $\mathcal{M}_{k-1}$. Here *best* is defined as having smallest RSS or highest R^2.
3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validated prediction error, C_p(AIC), BIC, or adjusted R^2.

Choosing the Optimal Model

Choosing the Optimal Model

- The model containing all of the predictors will always have the smallest RSS and the largest R^2, since these quantities are related to the training error.
- We wish to choose a model with low test error, not a model with low training error.
- Therefore, RSS and R^2 are not suitable for selecting the best model among a collection of models with different numbers of predictors.

Estimating Test Error: Two Approaches

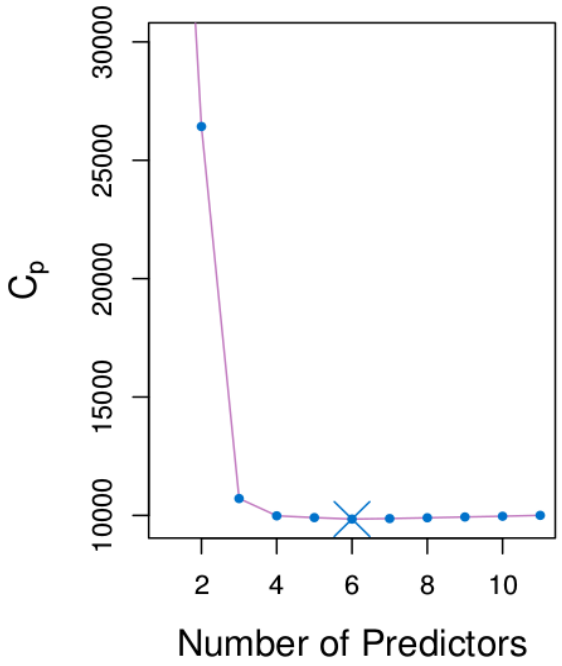
- We can *indirectly* estimate test error by making an *adjustment* to the training error to account for the bias due to overfitting.
- We can *directly* estimate the test error, using either a validation set approach or a cross-validation approach, as discussed in previous lectures.

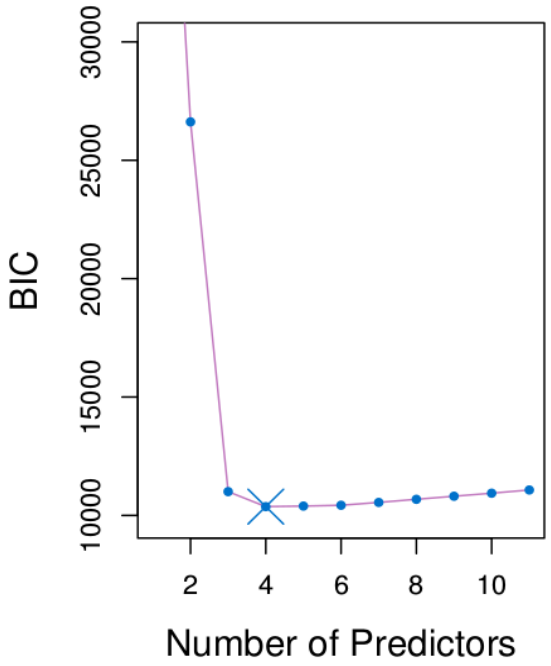
C_p, AIC, BIC, and Adjusted R^2

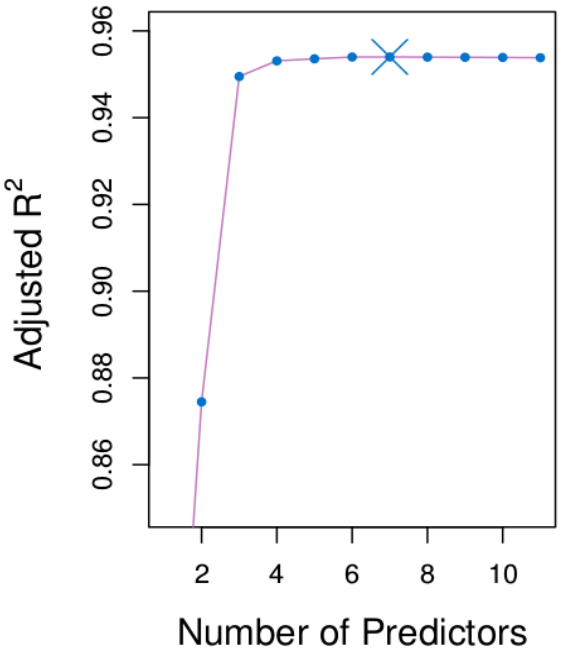
C_p, AIC, BIC, and Adjusted R^2

- These techniques adjust the training error for the model size, and can be used to select among a set of models with different numbers of variables.
- The next figure displays C_p, BIC, and adjusted R^2 for the best model of each size produced by best subset selection on the *Credit* data set.

Credit data example







Details on the adjustments

- *Mallow's* C_p :

$$C_p = \frac{1}{n}(\text{RSS} + 2d\hat{\sigma}^2)$$

where d is the total # of parameters used and $\hat{\sigma}^2$ is an estimate of the variance of the error associated with each response measurement.

- The AIC criterion is defined for a large class of models fit by maximum likelihood:

$$\text{AIC} = -2\log L + 2 \cdot d$$

where L is the maximized value of the likelihood function for the estimated model.

- In the case of the linear model with Gaussian errors, maximum likelihood and least squares are the same thing, and C_p and AIC are equivalent.

Details on BIC

Details on BIC

$$\text{BIC} = \frac{1}{n}(\text{RSS} + \log(n) d \hat{\sigma}^2)$$

- Like C_p, the BIC will tend to take on a small value for a model with a low test error, and so generally we select the model that has the lowest BIC value.
- Notice that BIC replaces the $2d\hat{\sigma}^2$ used by C_p with a $\log(n)d\hat{\sigma}^2$ term, where n is the number of observations.
- Since $log(n) > 2$ for any $n > 7$, the BIC statistic generally places a heavier penalty on models with many variables, and hence results in the selection of smaller models than C_p. See Figure on previous slides.

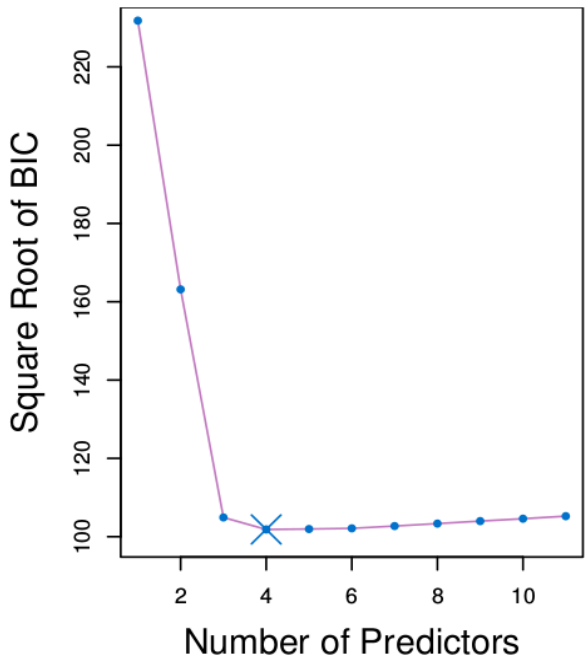
Adjusted R^2

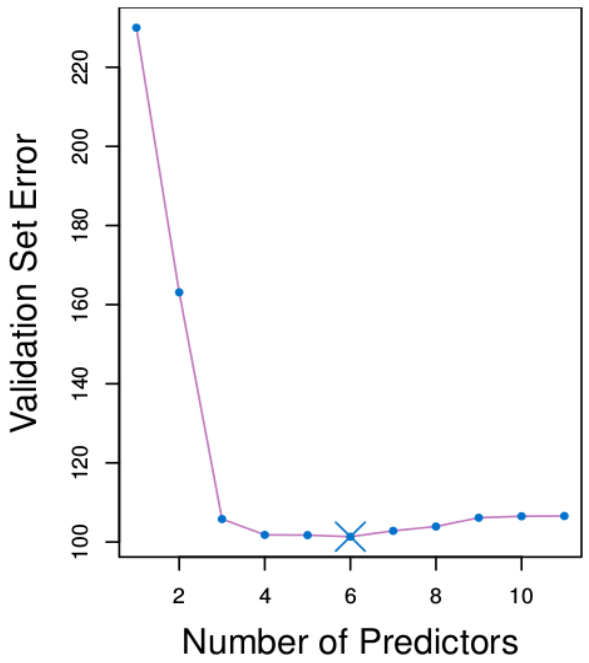
- For a least squares model with d variables, the adjusted R^2 statistic is calculated as

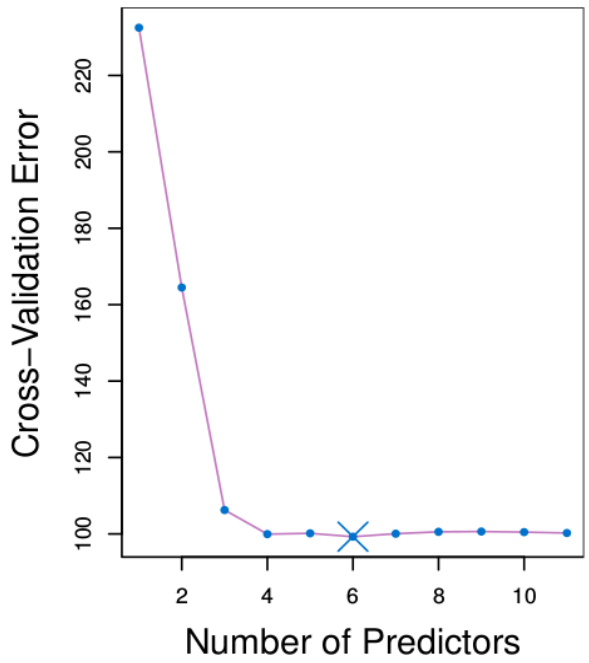
$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

  where TSS is the total sum of squares.
- Unlike C_p, AIC, and BIC, for which a *small* value indicates a model with a low test error, a *large* value of adjusted R^2 indicates a model with a small test error.
- Maximizing the adjusted R^2 is equivalent to minimizing $\frac{\text{RSS}}{n-d-1}$. While RSS always decreases as the number of variables in the model increases, $\frac{\text{RSS}}{n-d-1}$ may increase or decrease, due to the presence of d in the denominator.
- Unlike the R^2 statistic, the adjusted R^2 statistic *pays a price* for the inclusion of unnecessary variables in the model. See also Figure on previous' slides.

Credit data example







Details of Previous Figure

- The validation errors were calculated by randomly selecting three-quarters of the observations as the training set, and the remainder as the validation set.
- The cross-validation errors were computed using $k = 10$ folds. In this case, the validation and cross-validation methods both result in a six-variable model.
- However, all three approaches suggest that the four-, five-, and six-variable models are roughly equivalent in terms of their test errors.
- In this setting, we can select a model using the *one-standard-error rule*. We first calculate the standard error of the estimated test MSE for each model size, and then select the smallest model for which the estimated test error is within one standard error of the lowest point on the curve.

Shrinkage Methods

Ridge regression and *Lasso*

- The subset selection methods use least squares to fit a linear model that contains a subset of the predictors.
- As an alternative, we can fit a model containing all p predictors using a technique that *constrains* or *regularizes* the coefficient estimates, or equivalently, that *shrinks* the coefficient estimates towards zero.
- It may not be immediately obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can significantly reduce their variance.

Ridge regression

- Recall that the least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimize

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

- In contrast, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

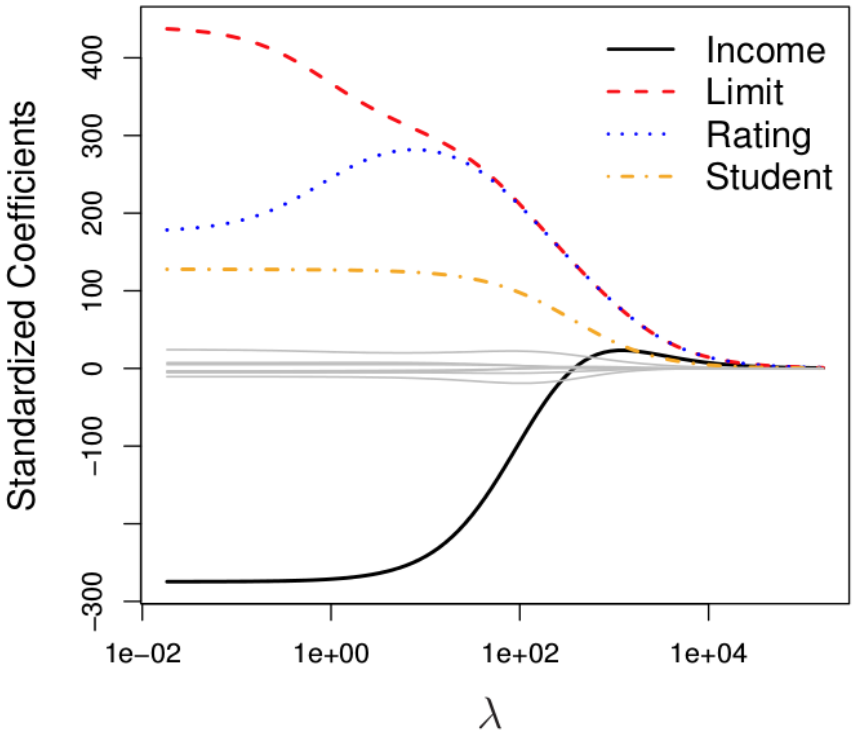
$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

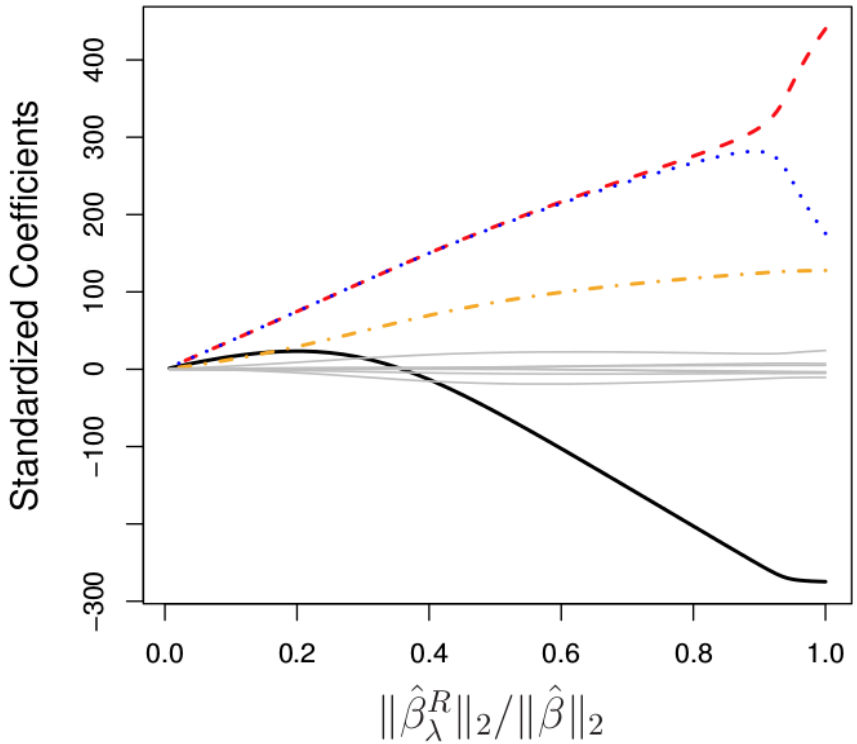
where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.

Ridge regression: continued

- As with least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small.
- However, the second term, $\lambda \sum_j \beta_j^2$, called a *shrinkage penalty*, is small when $\beta_1, \ldots, \beta_p$ are close to zero, and so it has the effect of *shrinking* the estimates of β_j towards zero.
- The tuning parameter λ serves to control the relative impact of these two terms on the regression coefficient estimates.
- Selecting a good value for λ is critical; cross-validation is used for this.

Credit data example





Details of Previous Figure

Details of Previous Figure

- In the left-hand panel, each curve corresponds to the ridge regression coefficient estimate for one of the ten variables, plotted as a function of λ.
- The right-hand panel displays the same ridge coefficient estimates as the left-hand panel, but instead of displaying λ on the x-axis, we now display $\|\hat{\beta}_\lambda^R\|_2/\|\hat{\beta}\|_2$, where $\hat{\beta}$ denotes the vector of least squares coefficient estimates.

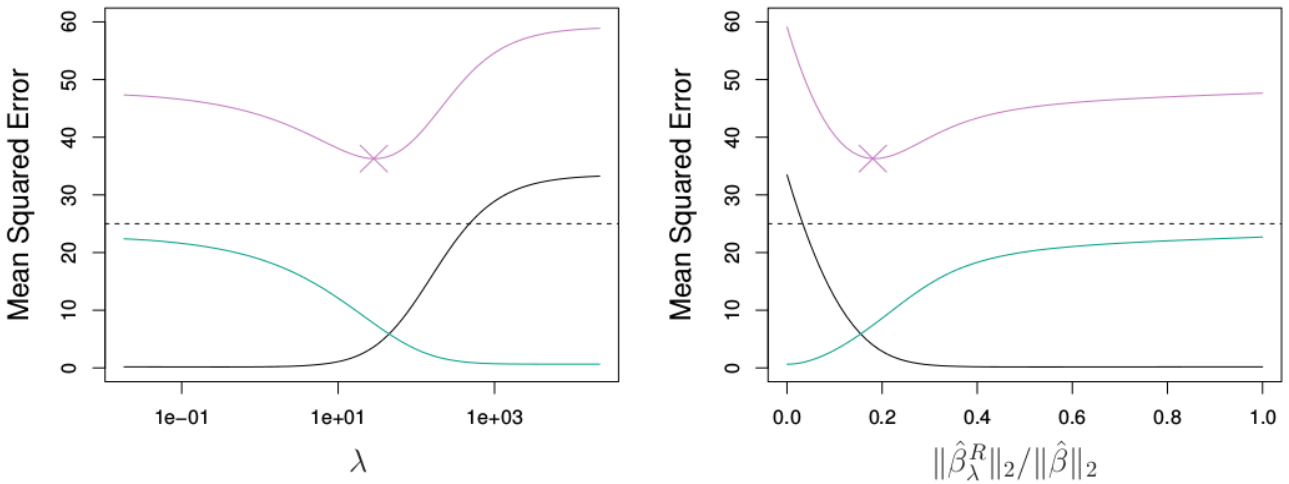
Ridge regression: scaling of predictors

- The standard least squares coefficient estimates are *scale equivariant*: multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$. In other words, regardless of how the jth predictor is scaled, $X_j\hat{\beta}_j$ will remain the same.
- In contrast, the ridge regression coefficient estimates can change *substantially* when multiplying a given predictor by a constant, due to the sum of squared coefficients term in the penalty part of the ridge regression objective function.
- Therefore, it is best to apply ridge regression after *standardizing the predictors*, using the formula

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \bar{x}_j)^2}}$$

Why Does Ridge Regression Improve Over Least Squares?

The Bias-Variance tradeoff



Simulated data with $n = 50$ observations, $p = 45$ predictors, all having nonzero coefficients. Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on a simulated data set, as a function of λ and $\|\hat{\beta}^R_\lambda\|_2/\|\hat{\beta}\|_2$. The horizontal dashed lines indicate the minimum possible MSE. The purple crosses indicate the ridge regression models for which the MSE is smallest.

The Lasso

The Lasso

- Ridge regression does have one obvious disadvantage: unlike subset selection, which will generally select models that involve just a subset of the variables, ridge regression will include all p predictors in the final model
- The Lasso is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, $\hat{\beta}_\lambda^L$, minimize the quantity

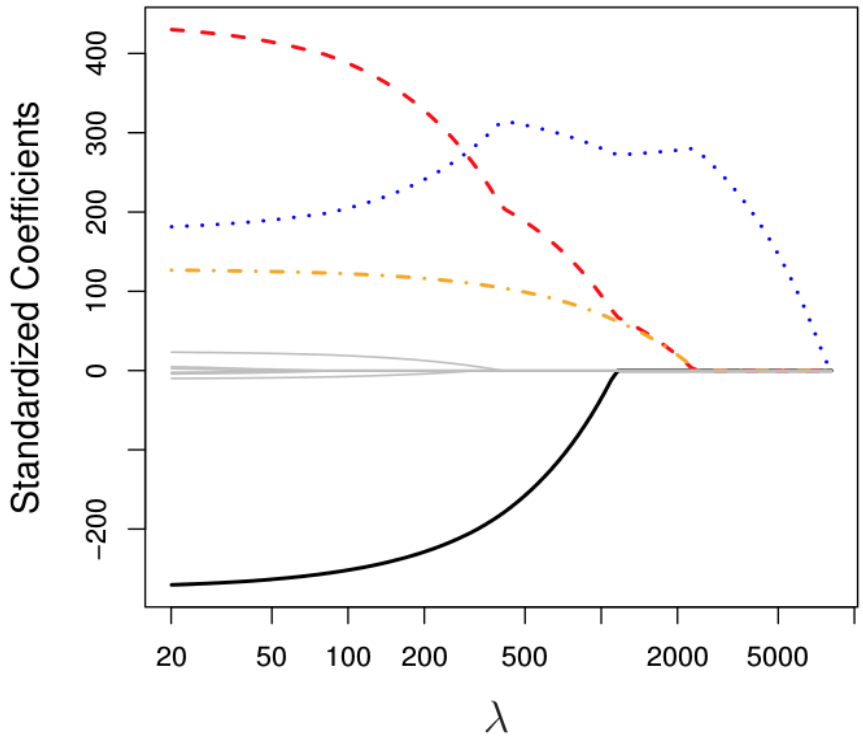
$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$$

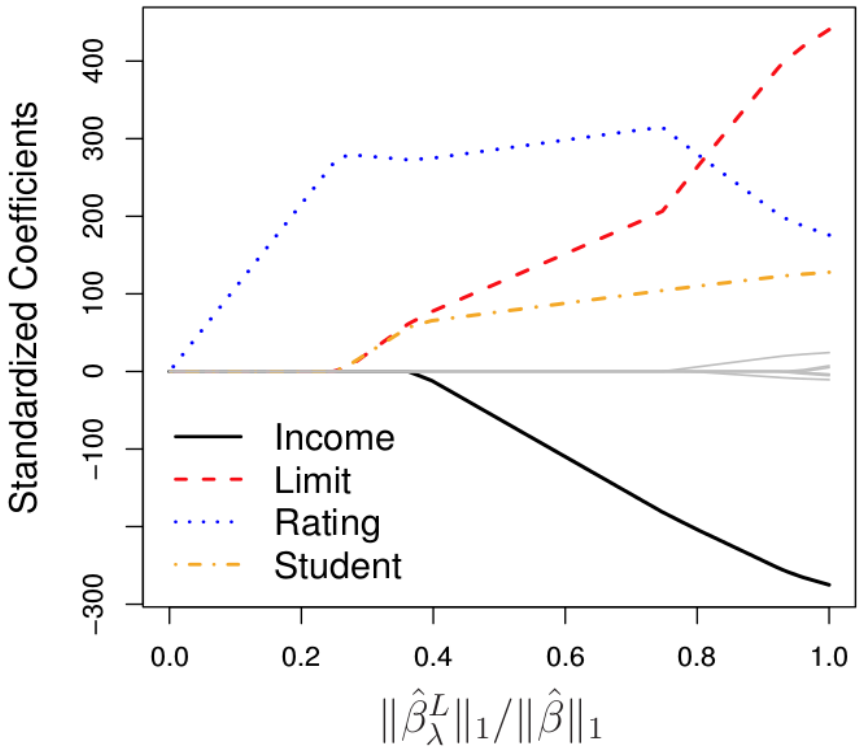
- This means the lasso uses an l_1 penalty instead of an l_2 penalty.

The Lasso: continued

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the l_1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter λ is sufficiently large.
- Hence, much like best subset selection, the lasso performs *variable selection*.
- We say that the lasso yields *sparse models* – that is, models that involve only a subset of the variables.
- As in ridge regression, selecting a good value of λ for the lasso is critical; cross-validation is again the method of choice.

Example: Credit dataset





The Variable Selection Property of the Lasso

Why is it, that the lasso, unlike ridge regression, results in coefficient estimates that are exactly equal to zero? One can show that the lasso and ridge regression coefficient estimates solve the problems

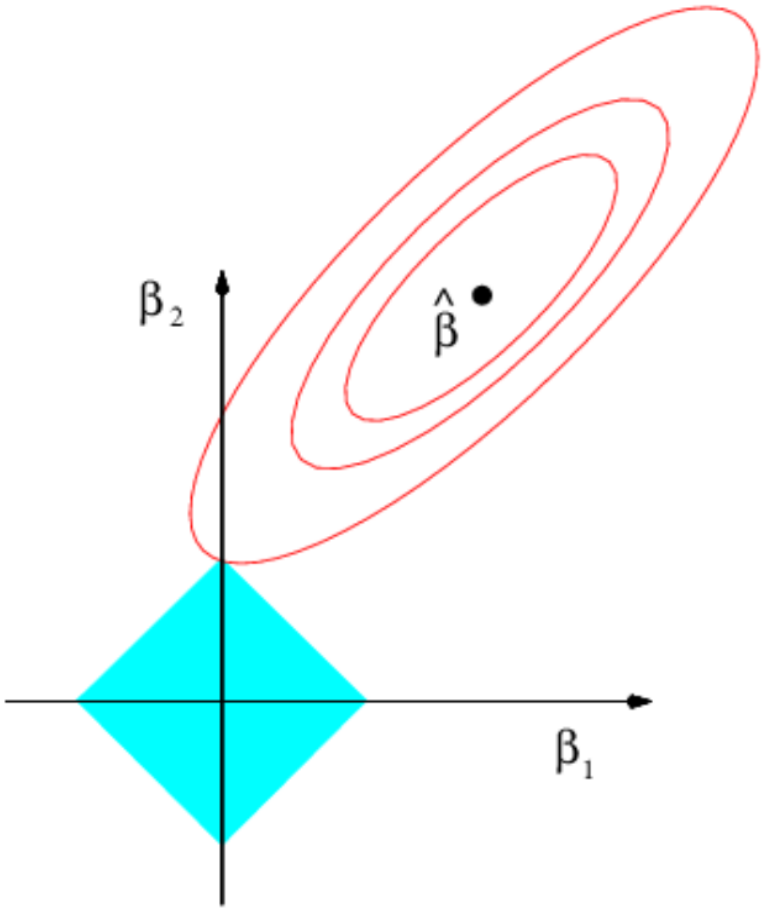
$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

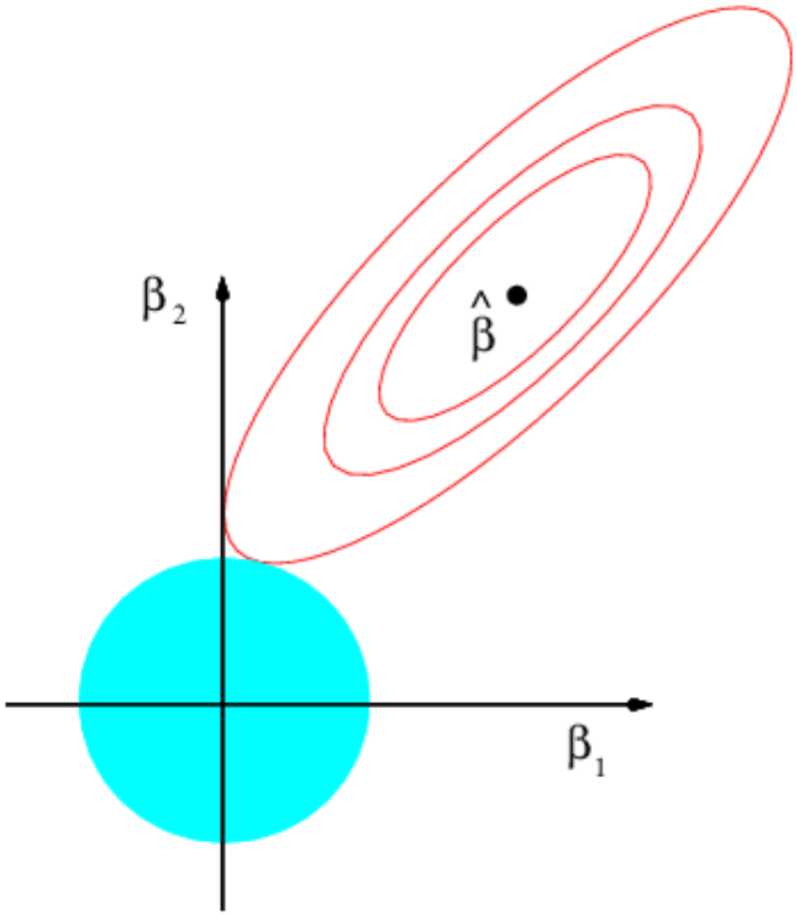
and

$$\min_{\beta} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s$$

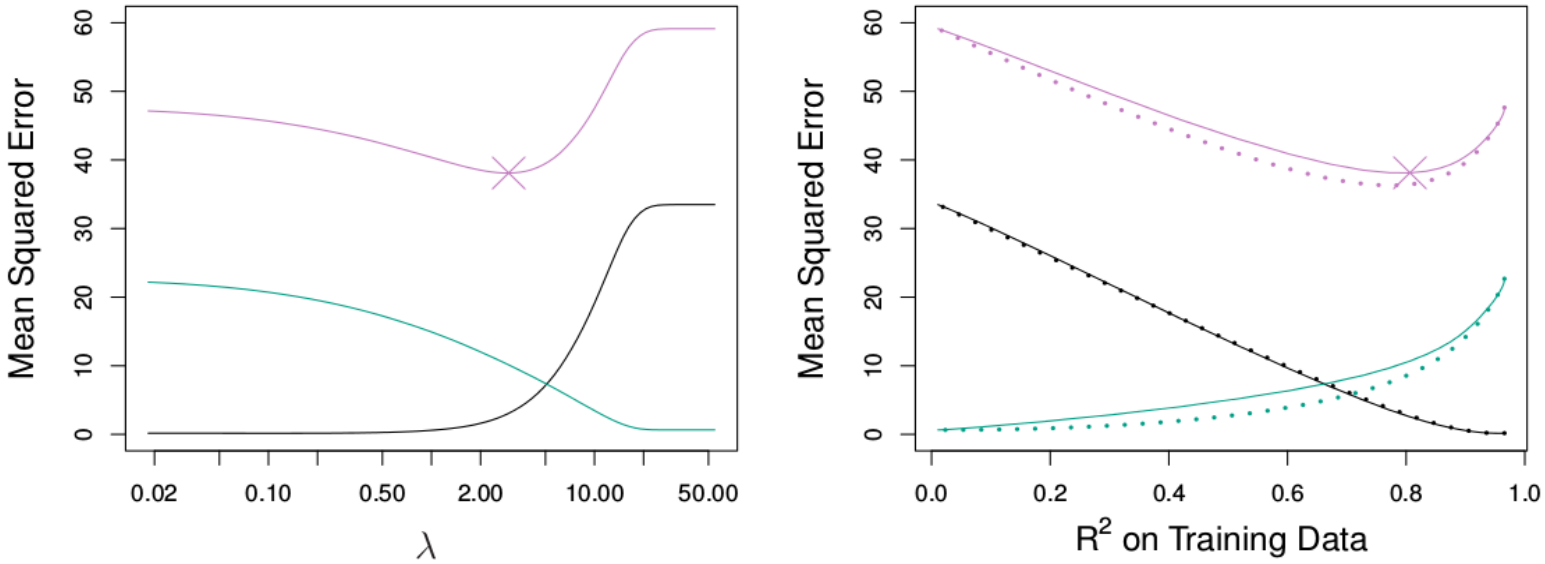
respectively.

The Lasso Picture



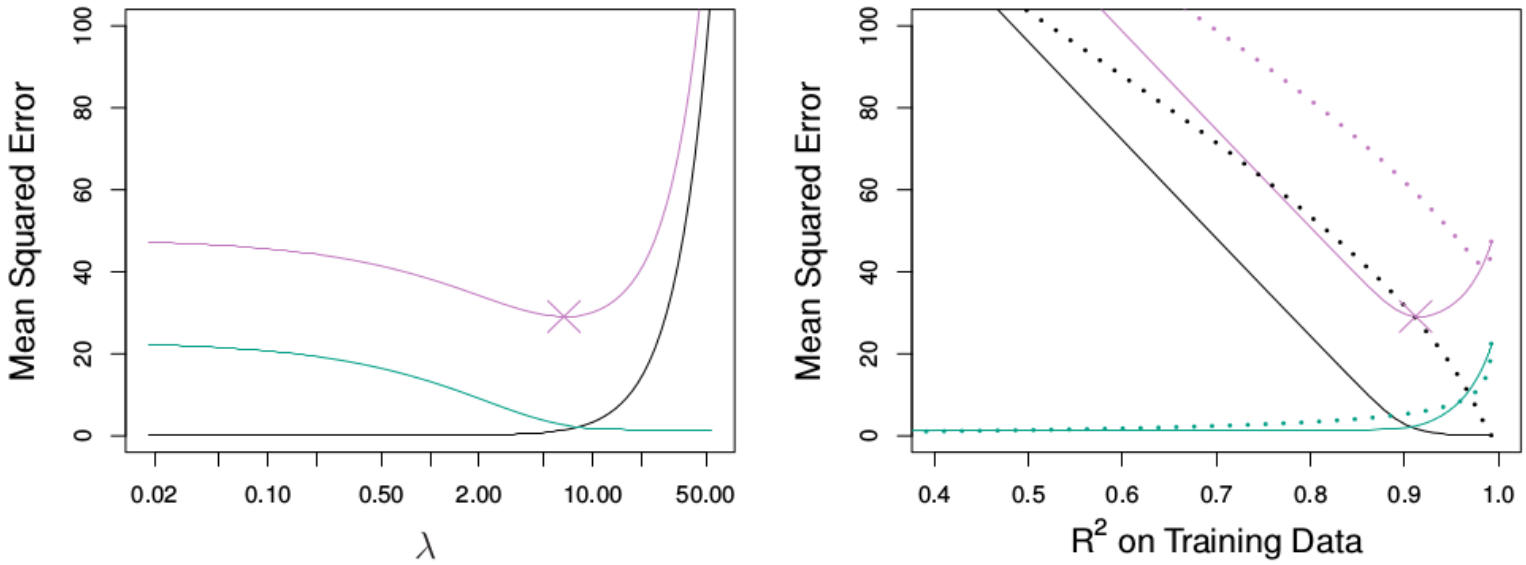


Comparing the Lasso and Ridge Regression



Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso on simulated data set of slide 36. *Right*: Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their R^2 on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

Comparing the Lasso and Ridge Regression: continued



Left: Plots of squared bias (black), variance (green), and test MSE (purple) for the lasso. The simulated data is similar to that on previous slide, except that now only two predictors are related to the response. Right: Comparison of squared bias, variance and test MSE between lasso (solid) and ridge (dashed). Both are plotted against their R^2 on the training data, as a common form of indexing. The crosses in both plots indicate the lasso model for which the MSE is smallest.

Conclusions

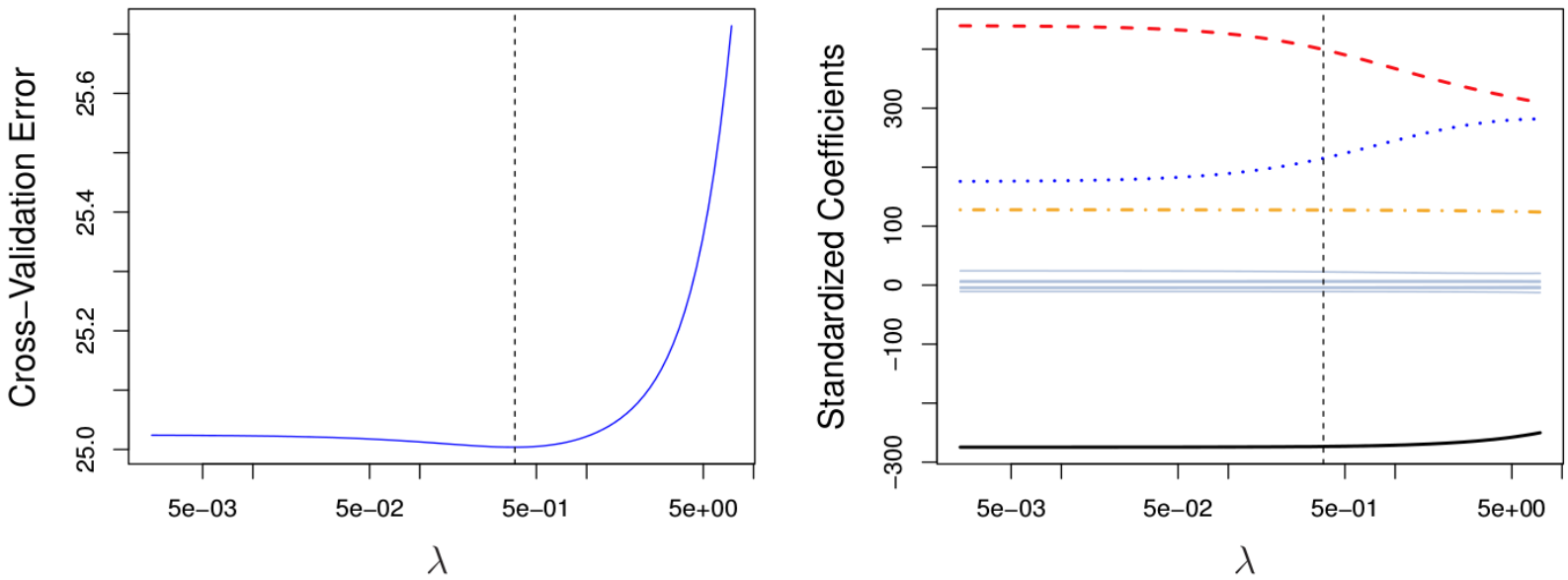
Conclusions

- These two examples illustrate that neither ridge regression nor the lasso will universally dominate the other.
- In general, one might expect the lasso to perform better when the response is a function of only a relatively small number of predictors.
- However, the number of predictors that is related to the response is never known a priori for real data sets.
- A technique such as cross-validation can be used in order to determine which approach is better on a particular data set.

Selecting the Tuning Parameter for Ridge Regression and Lasso

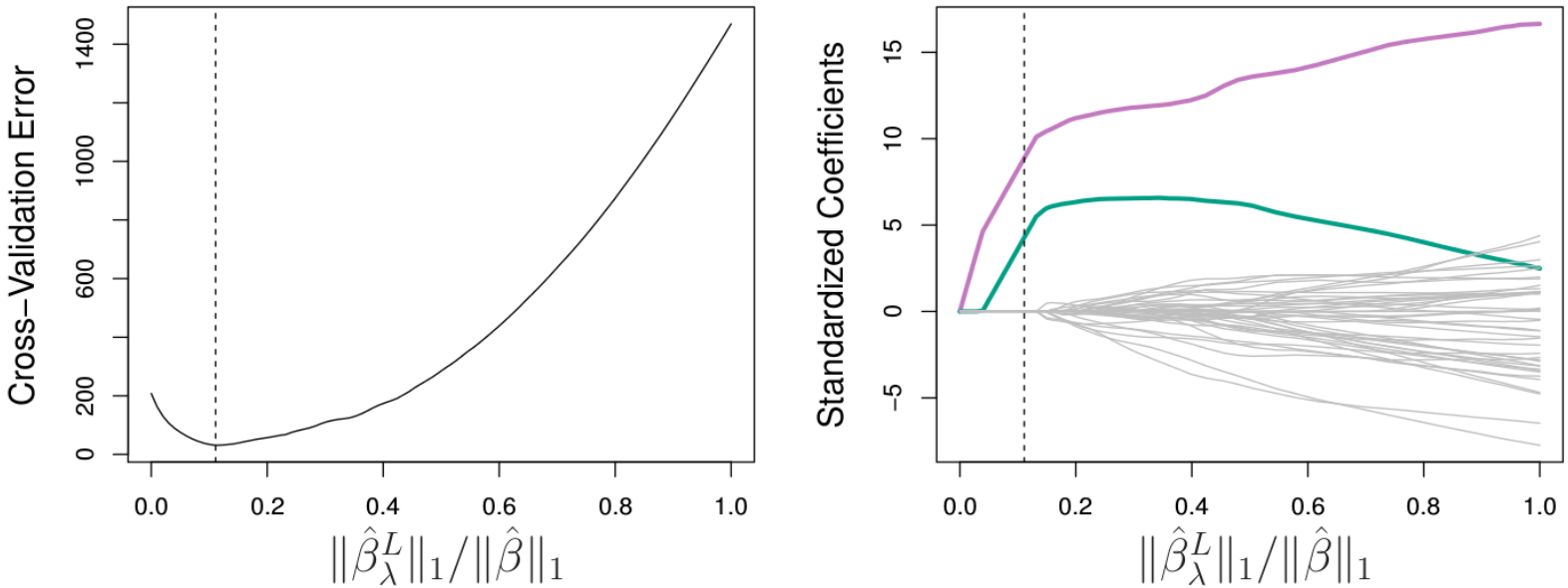
- As for subset selection, for ridge regression and lasso we require a method to determine which of the models under consideration is best.
- That is, we require a method selecting a value for the tuning parameter λ or equivalently, the value of the constraint s.
- Cross-validation provides a simple way to tackle this problem. We choose a grid of λ values, and compute the cross-validation error rate for each value of λ.
- We then select the tuning parameter value for which the cross-validation error is smallest.
- Finally, the model is re-fit using all of the available observations and the selected value of the tuning parameter.

Credit data example



Left: Cross-validation errors that result from applying ridge regression to the `Credit` data set with various values of λ. *Right*: The coefficient estimates as a function of λ. The vertical dashed lines indicates the value of λ selected by cross-validation.

Simulated data example



Left: Ten-fold cross-validation MSE for the lasso, applied to the sparse simulated data set from slide 43. *Right*: The corresponding lasso coefficient estimates are displayed. The vertical dashed lines indicate the lasso fit for which the cross-validation error is smallest.

Dimension Reduction Methods

Dimension Reduction Methods

- The methods that we have discussed so far in this lecture have involved fitting linear regression models, via least squares or a shrunken approach, using the original predictors, $X_1, X_2, \ldots, X_p$.
- We now explore a class of approaches that *transform* the predictors and then fit a least squares model using the transformed variables. We will refer to these techniques as *dimension reduction* methods.

Dimension Reduction Methods: details

- Let $Z_1, Z_2, \ldots, Z_M$ represent $M < p$ linear combinations of our original p predictors. That is,

$$Z_m = \sum_{j=1}^{p} \phi_{mj} X_j \tag{1}$$

  for some constants $\phi_{m1}, \ldots, \phi_{mp}$
- We can then fit the linear regression model,

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i, \qquad i = 1, \ldots, n, \tag{2}$$

  using ordinary least squares.
- Note that in model (2), the regression coefficients are given by $\theta_0, \theta_1, \ldots, \theta_M$. If the constants $\phi_{m1}, \ldots, \phi_{mp}$ are chosen wisely, then such dimension reduction approaches can often outperform OLS regression.

Dimension Reduction Methods: details

- Notice that from definition (1),

$$\sum_{m=1}^{M} \theta_m z_{im} = \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \phi_{mj} x_{ij} = \sum_{m=1}^{M} \sum_{j=1}^{p} \theta_m \phi_{mj} x_{ij} = \sum_{j=1}^{p} \beta_j x_{ij}$$

  where

$$\beta_j = \sum_{m=1}^{M} \theta_m \phi_{mj} \tag{3}$$

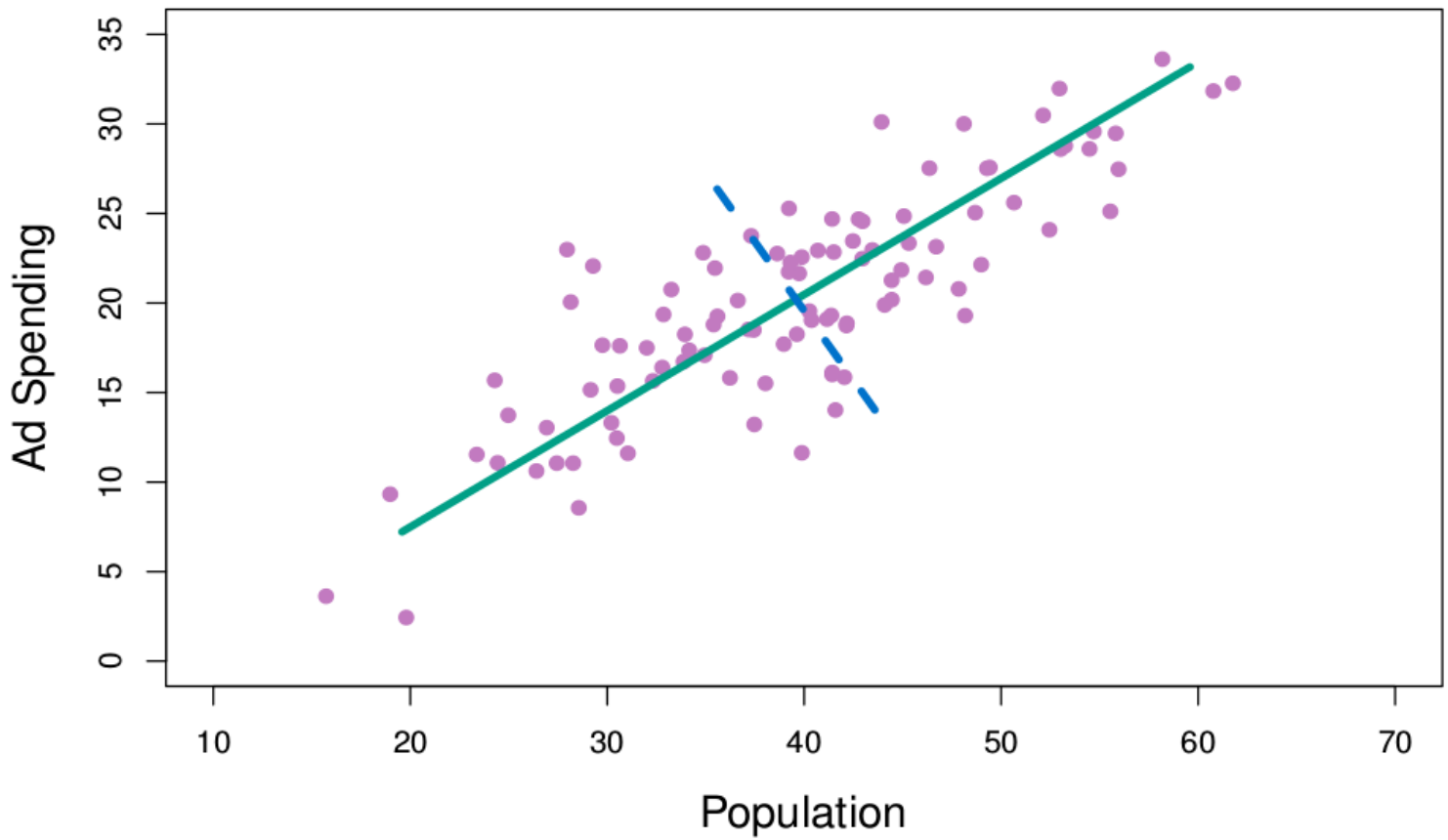
- Hence model (2) can be thought of as a special case of the original linear regression model.
- Dimension reduction serves to constrain the estimated β_j coefficients, since now they must take the form (3).
- Can win in the bias-variance tradeoff.

Principal Components Analysis

Principal Components Analysis

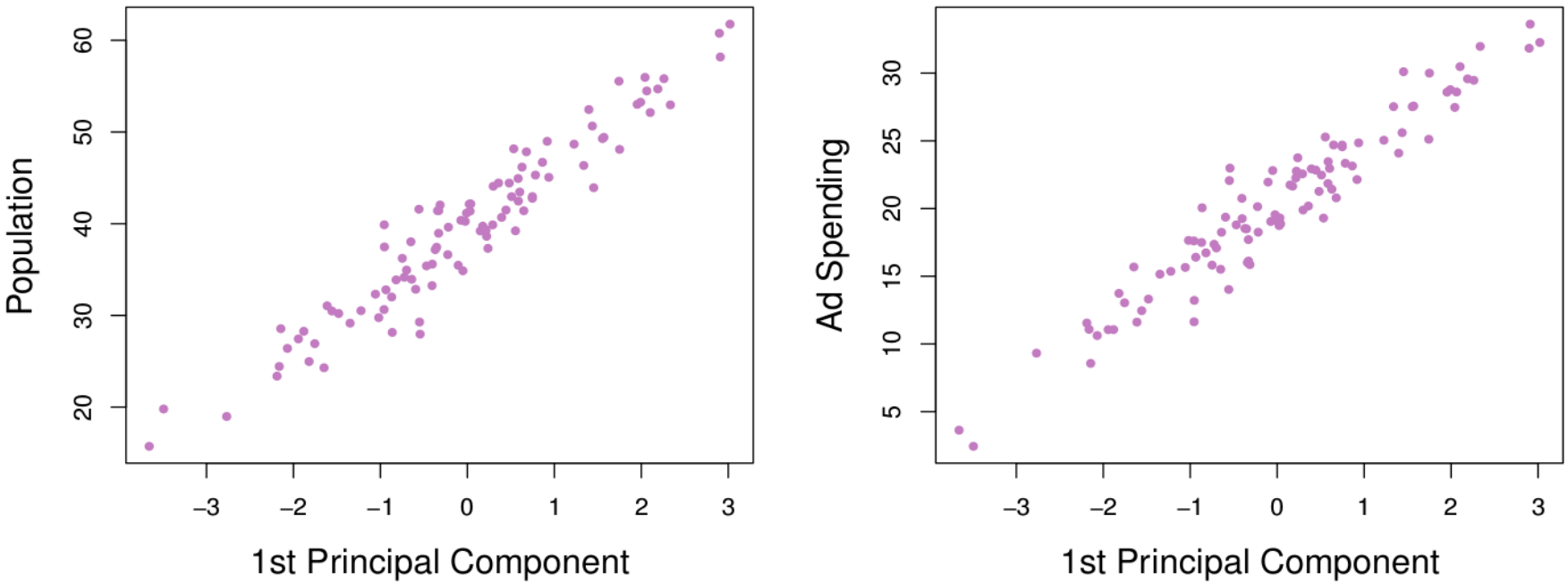
- Here we apply principal components analysis (PCA) to define the linear combinations of the predictors, for use in our regression.
- The first principal component is that (normalized) linear combination of the variables with the largest variance.
- The second principal component has largest variance, subject to being uncorrelated with the first.
- And so on.
- Hence with many correlated original variables, we replace them with a small set of principal components that capture their joint variation.

Pictures of PCA



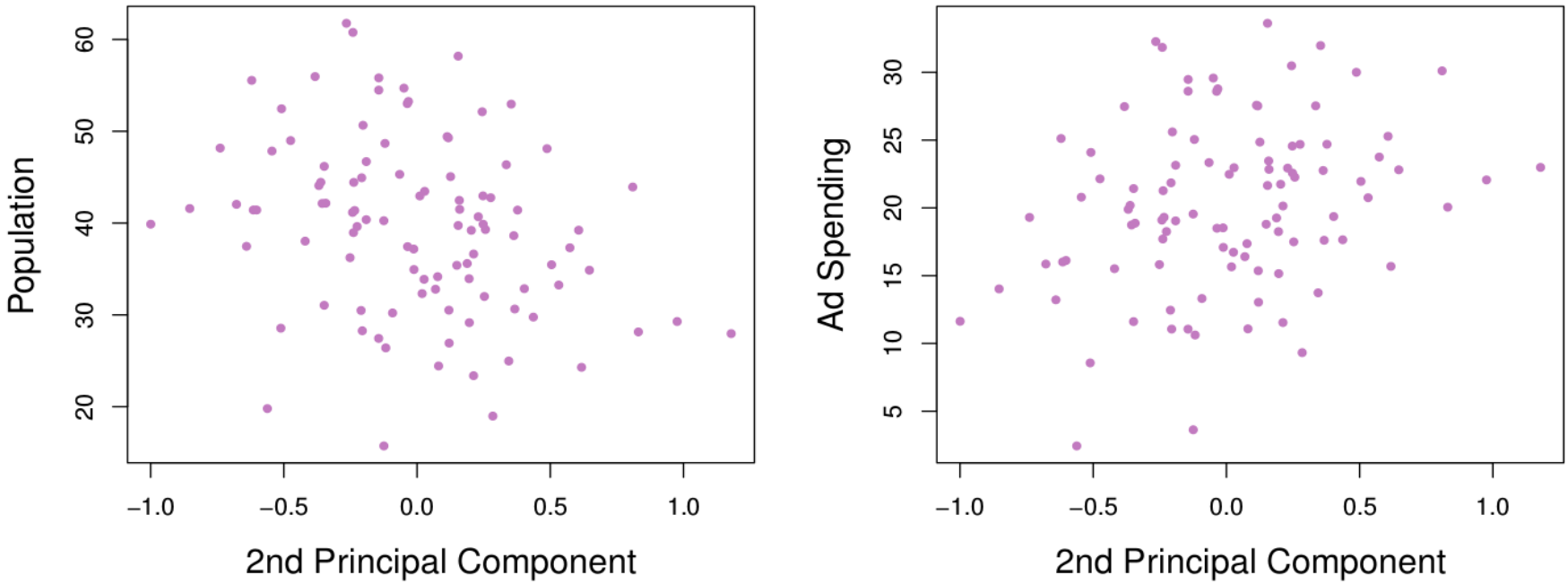
The population size (pop) and ad spending (ad) for 100 different cities are shown as purple circles. The green solid line indicates the first principal component, and the blue dashed line indicates the second principal component.

Pictures of PCA: continued



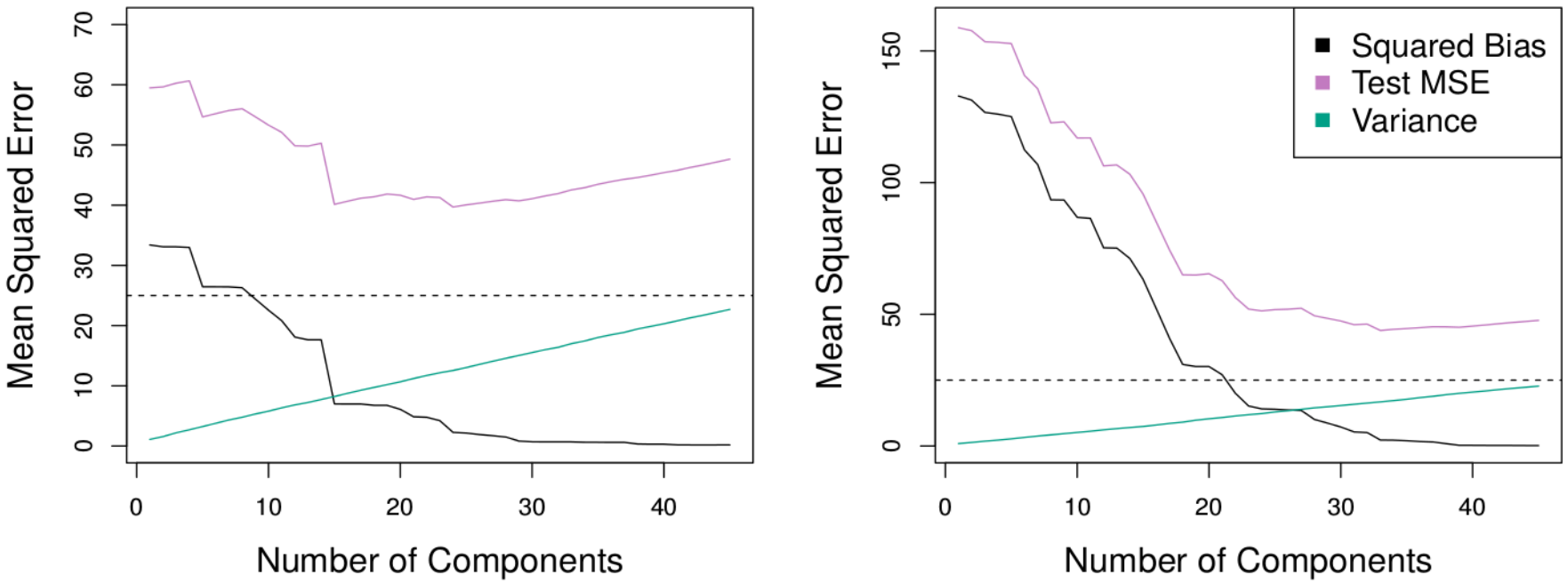
Plots of the first principal component scores z_{i1} versus *pop* and *ad*. The relationships are strong.

Pictures of PCA: continued



Plots of the second principal component scores z_{i2} versus *pop* and *ad*. The relationships are weak.

Application to Principal Components Regression



PCR was applied to two simulated data sets. The black, green, and purple lines correspond to squared bias, variance, and test mean squared error, respectively. *Left*: Simulated data from slide 36. *Right*: Simulated data from slide 43.

Support Vector Machines

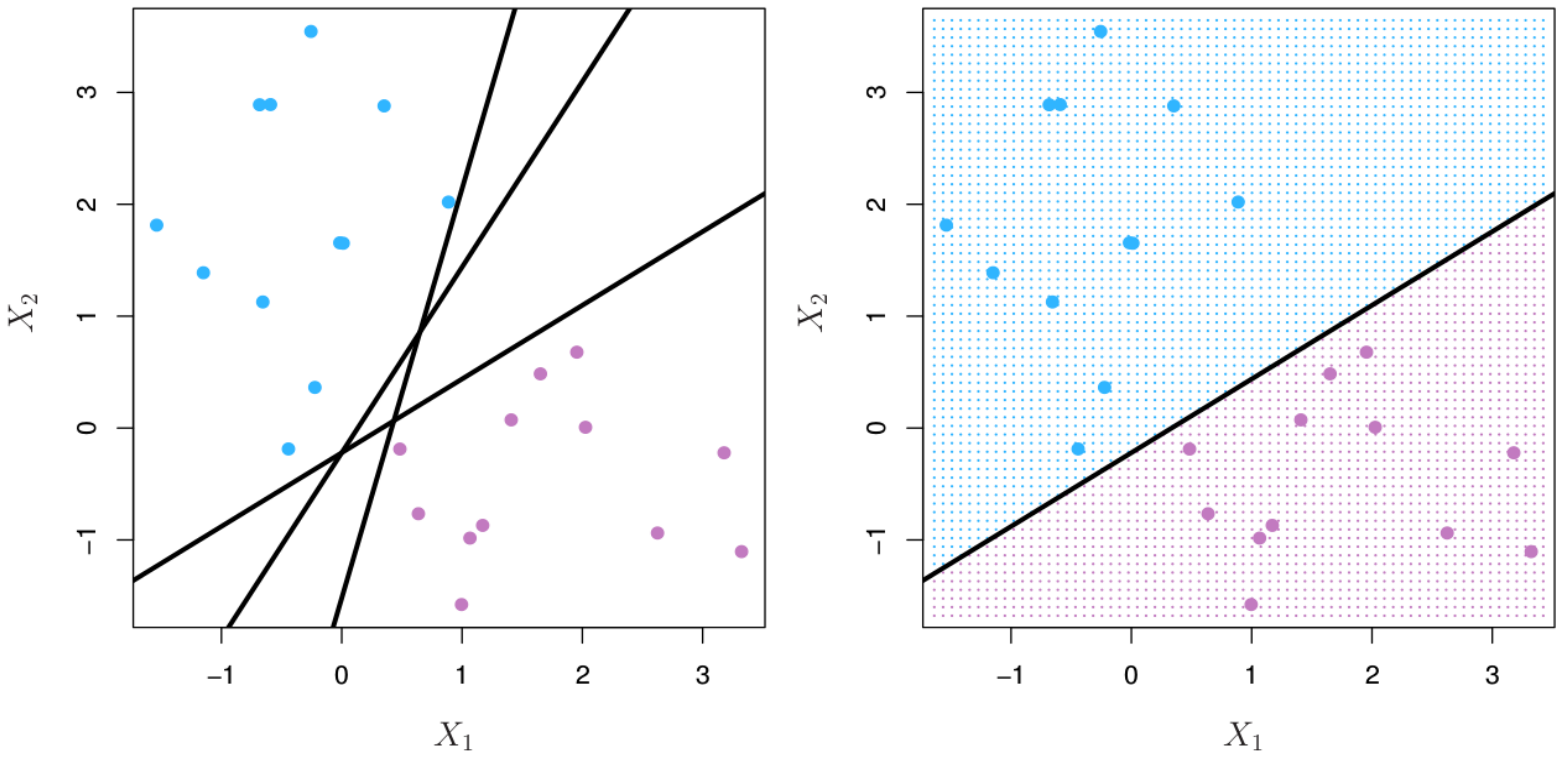
Support Vector Machines

Here we approach the two-class classification problem in a direct way:
We try and find a plane that separates the classes in feature space.

If we cannot, we get creative in two ways:

- We soften what we mean by "separates", and
- We enrich and enlarge the feature space so that separation is possible.

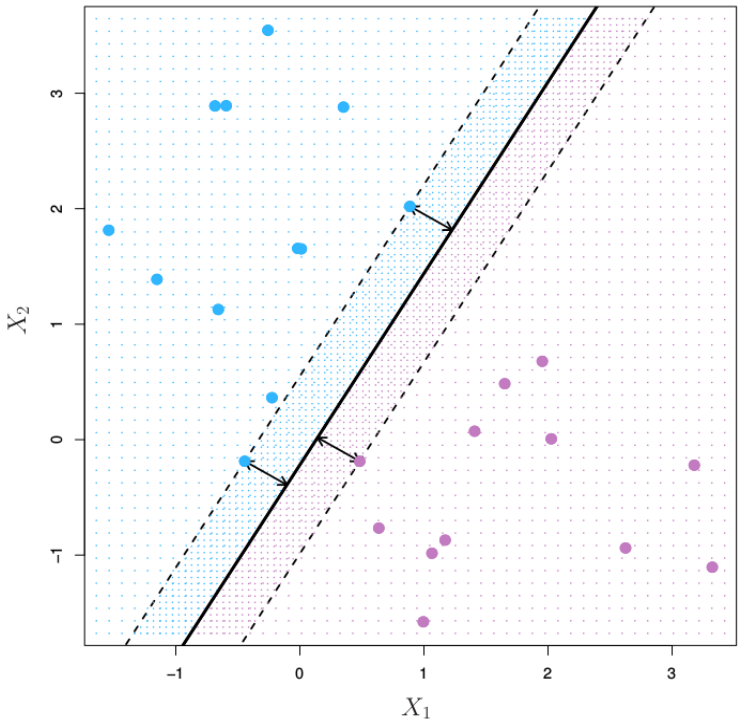
Separating Hyperplanes



- If $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$, then $f(X) > 0$ for points on one side of the hyperplane, and $f(X) < 0$ for points on the other.
- If we code the colored points as $Y_i = +1$ for blue, say, and $Y_i = -1$ for mauve, then if $Y_i \cdot f(X_i) > 0$ for all i, $f(X) = 0$ defines a *separating hyperplane*.

Maximal Margin Classifier

Among all separating hyperplanes, find the one that makes the biggest gap or margin between the two classes.



Constrained optimization problem for $y \in \{-1, 1\}$

$$\max_{\beta_0, \beta_1, \ldots, \beta_p} M$$

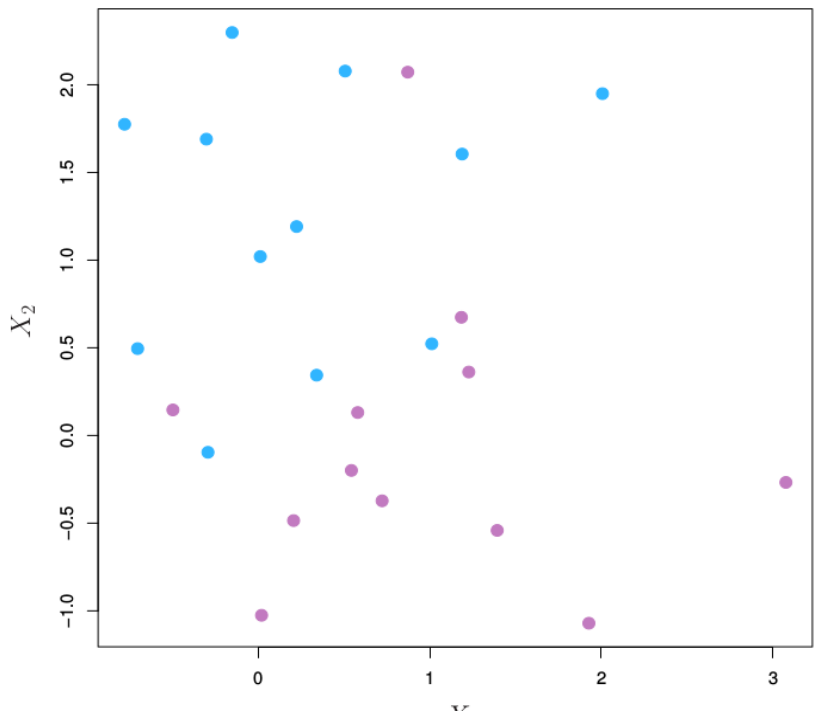
subject to $\sum_{j=1}^{p} \beta_j^2 = 1$,

$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M$$

for all $i = 1, \ldots, N$.

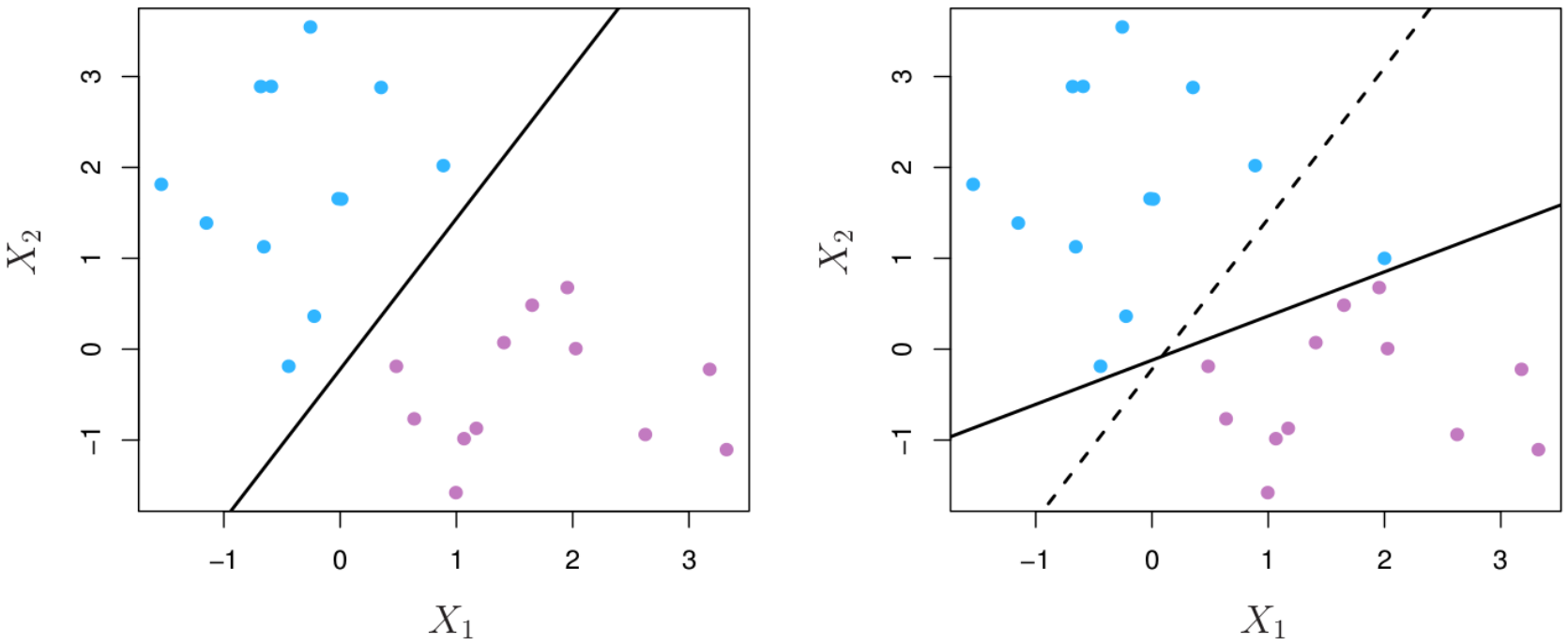
This can be rephrased as a convex quadratic program and solved efficiently.

Non-separable Data



The data on the left are not separable by a linear boundary. This is often the case, unless $N < p$.

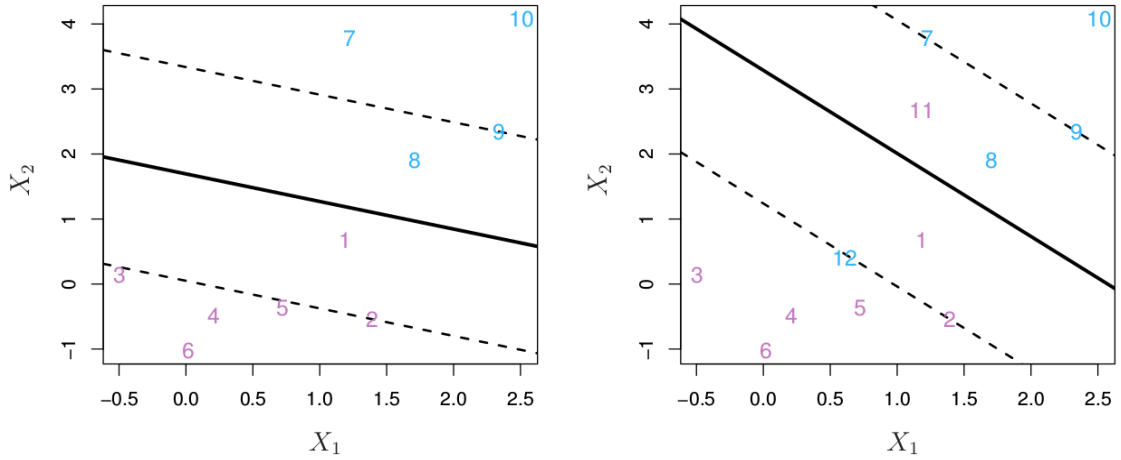
Noisy Data



Sometimes the data are separable, but noisy. This can lead to a poor solution for the maximal-margin classifier.

The *support vector classifier* maximizes a *soft* margin.

Support Vector Classifier

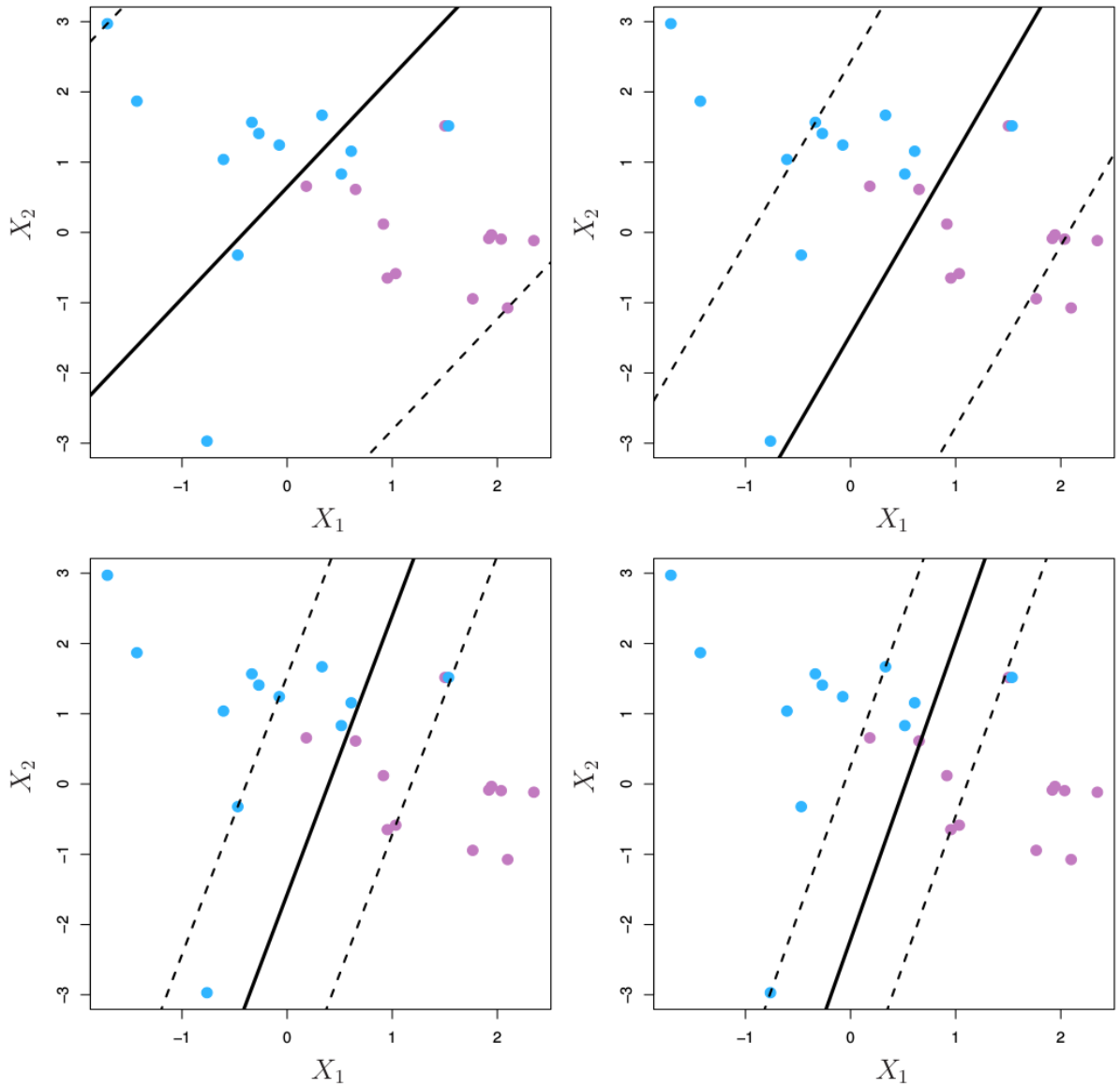


$$\max_{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots\epsilon_n} M \qquad \text{subject to} \qquad \sum_{j=1}^{p} \beta_j^2 = 1,$$

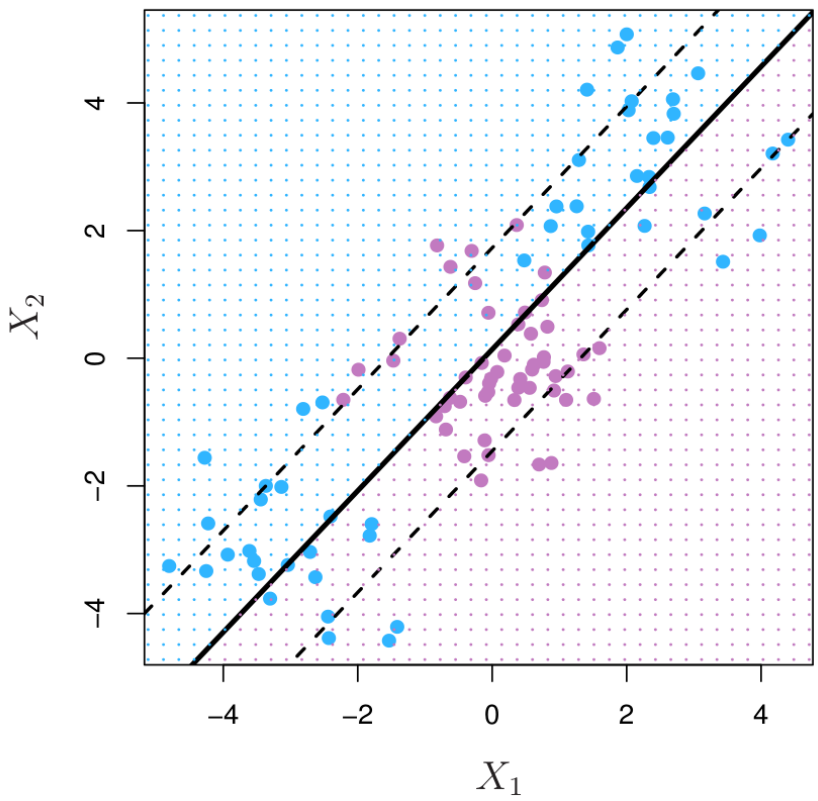
$$y_i(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

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

C is a regularization parameter



Linear boundary can fail



Sometime a linear boundary simply won't work, no matter what value of C.

The example on the left is such a case.

What to do?

Feature Expansion

Feature Expansion

- Enlarge the space of features by including transformations; e.g. $X_1^2, X_1^3, X_1X_2, X_1X_2^2, \ldots$. Hence go from a p-dimensional space to a $M > p$ dimensional space.
- Fit a support-vector classifier in the enlarged space.
- This results in non-linear decision boundaries in the original space.

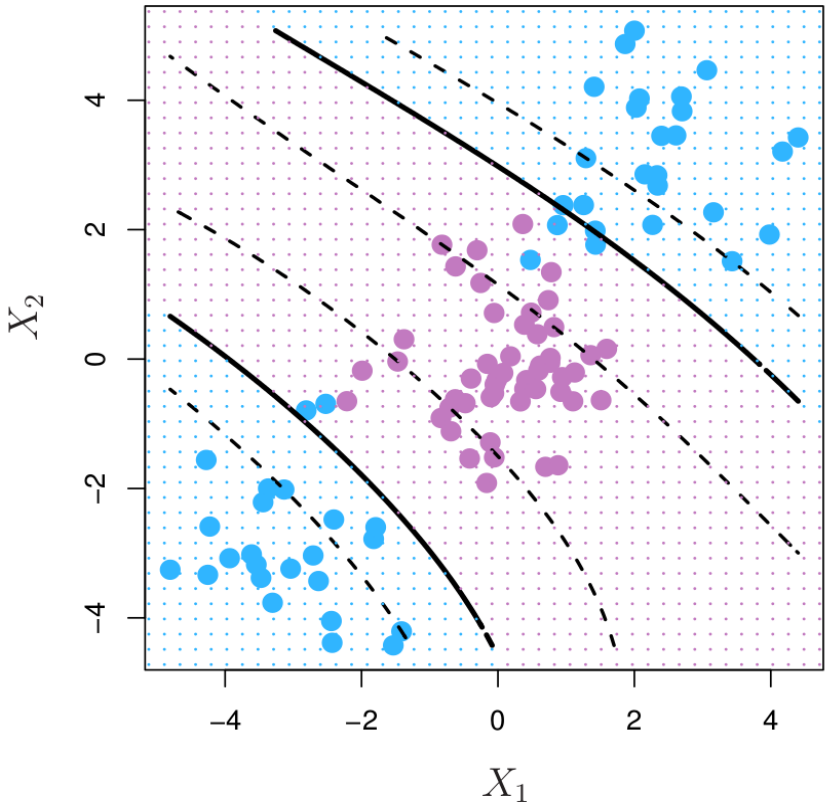
Example: Suppose we use $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$ instead of just (X_1, X_2). Then the decision boundary would be of the form

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

This leads to nonlinear decision boundaries in the original space (quadratic conic sections).

Cubic Polynomials

Here we use a basis expansion of cubic polynomials
The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space



$$\beta_0+\beta_1X_1+\beta_2X_2+\beta_3+X_1^2+\beta_4X_2^2+\beta_5X_1X_2+\beta_6X_1^3+\beta_7X_2^3+\beta_8X_1X_2^2+\beta_9X_1^2X_2=0$$

Nonlinearities and Kernels

Nonlinearities and Kernels

- Polynomials (especially high-dimensional ones) get wild rather fast.
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers – through the use of *kernels*.
- Before we discuss these, we must understand the role of *inner products* in support-vector classifiers.

Inner products and support vectors

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j} \qquad \text{– inner product between vectors}$$

- The linear support vector classifier can be represented as

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \qquad \text{– n parameters}$$

- To estimate the parameters $\alpha_1, \ldots, \alpha_n$ and β_0 , all we need are the inner products $\langle x_i, x_{i'} \rangle$ between all pairs of training observations

It turns out that most of the $\hat{\alpha}_i$ are zero and we have

$$f(x) = \hat{\beta}_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i \langle x, x_i \rangle,$$

where $\mathcal{S}$ is the set of indices, which lie within or on the soft margin.

Kernels and Support Vector Machines

- Hence, as soon as we can compute inner-products between observations, we can fit a SV classifier. This can be generalized to non-linear transformations.
- Some special *kernel functions* can do this for us. E.g.

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

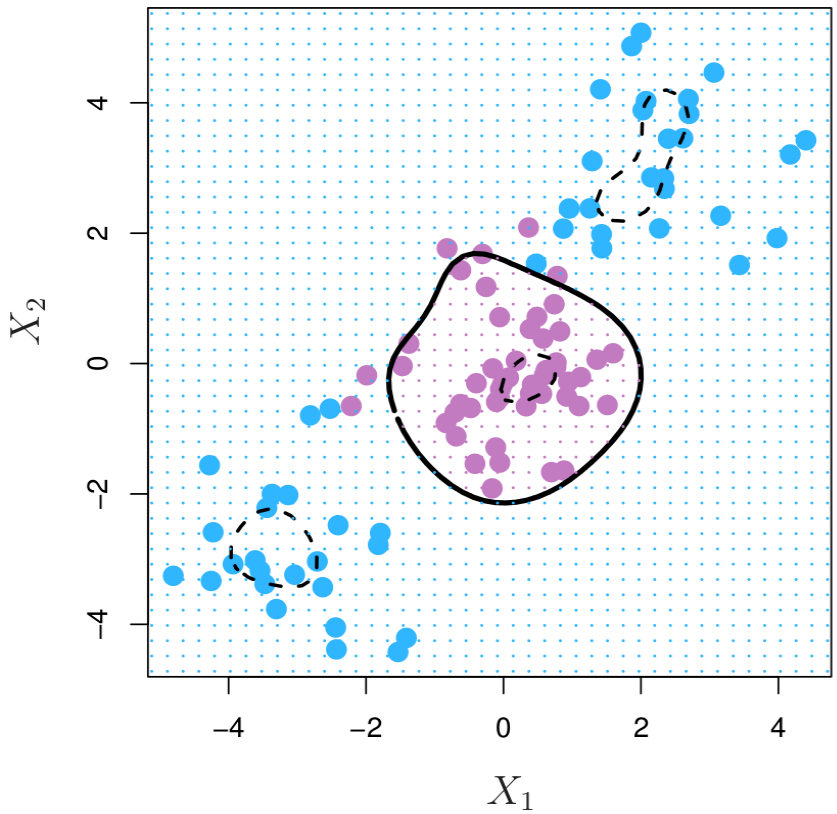
computes the inner-products needed for d dimensional polynomials – $\binom{p+d}{d}$ basis functions!

- The solution has the form

$$f(x) = \hat{\beta}_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

Radial Kernel

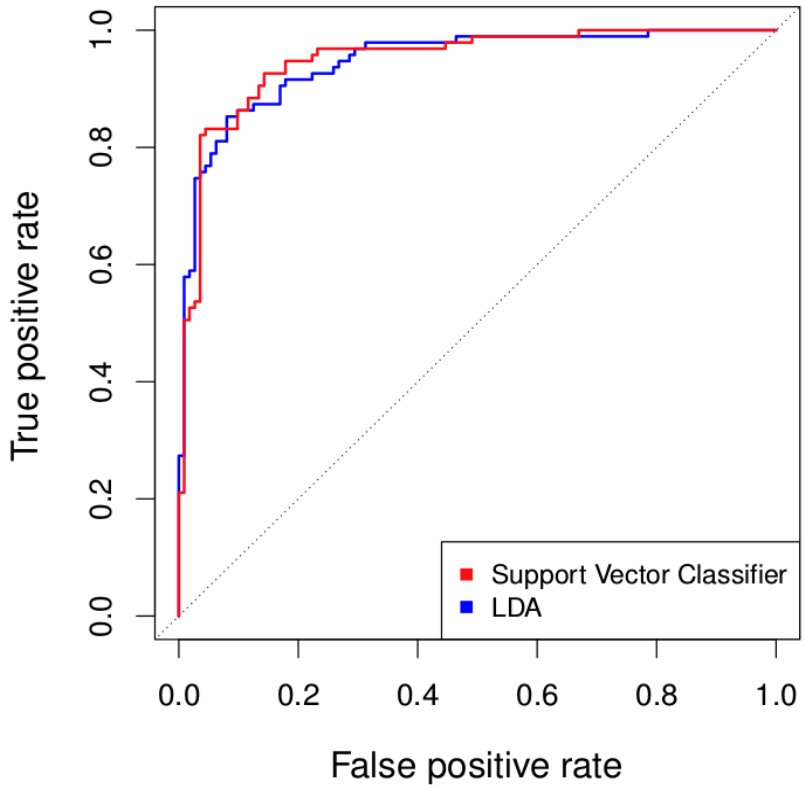
$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

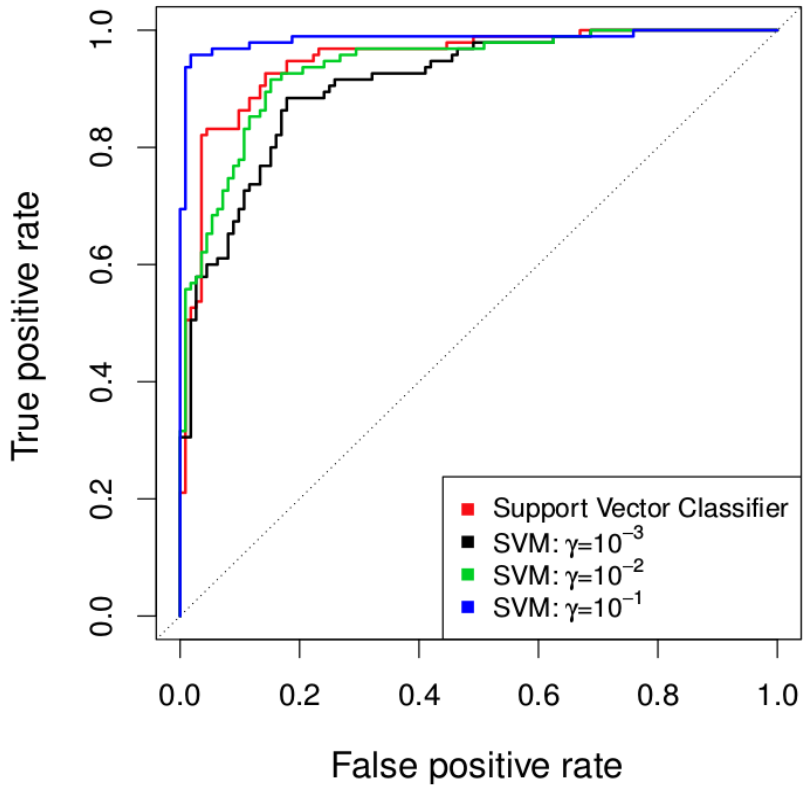


$$f(x) = \hat{\beta}_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

Implicit feature space; infinte dimensional.
Controls variance by squashing down most dimensions severely

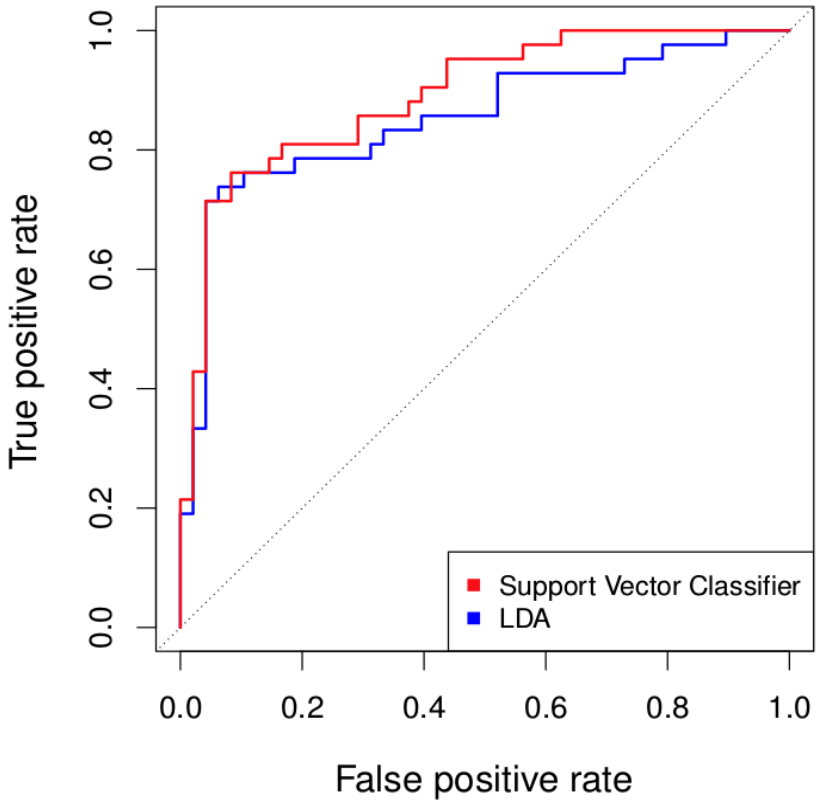
Example: Heart Data

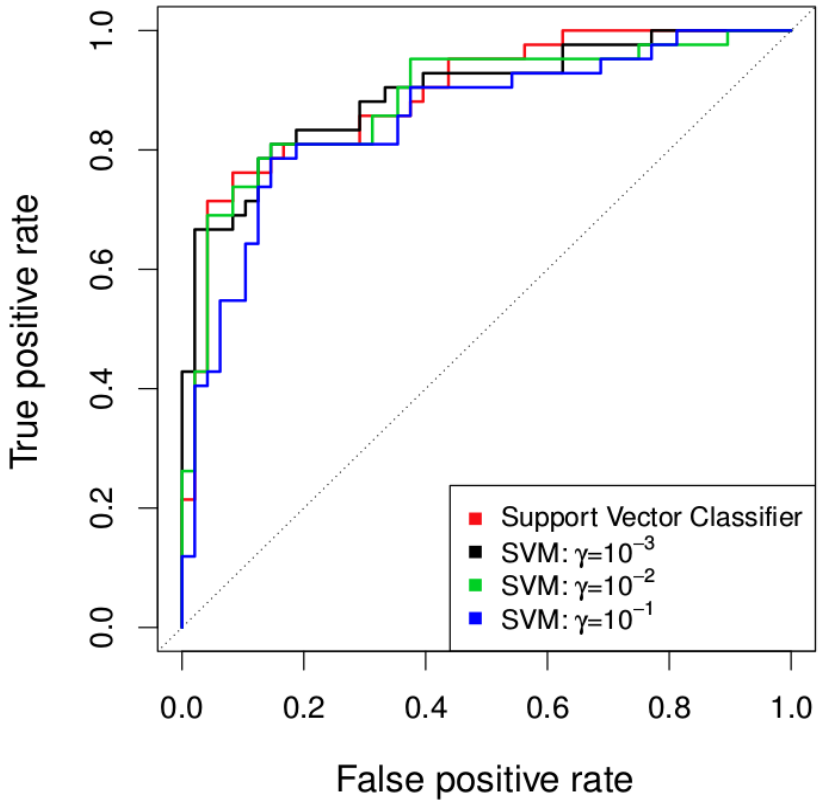




ROC curve is obtained by changing the threshold 0 to threshold t in $\hat{f}(X) > t$, and recording *false positive* and *true positive* rates as t varies. Here we see ROC curves on training data.

Example continued: Heart Test Data





SVMs: more than 2 classes?

SVMs: more than 2 classes?

The SVM as defined works for $K = 2$ classes. What do we do if we have $K > 2$ classes?

OVA One versus All. Fit K different 2-class SVM classifiers $\hat{f}_k(x), k = 1, \ldots, K$; each class versus the rest. Classify x^* to the class for which $\hat{f}_k(x^*)$ is largest.

OVO One versus One. Fit all $\binom{K}{2}$ pairwise classifiers $\hat{f}_{kl}(x)$. Classify x^* to the class that wins the most pairwise competitions.

Which to choose? If K is not too large, use OVO.