Final Overview

Introduction to ML

Marek Petrik

4/25/2017

This Course: Introduction to Machine Learning

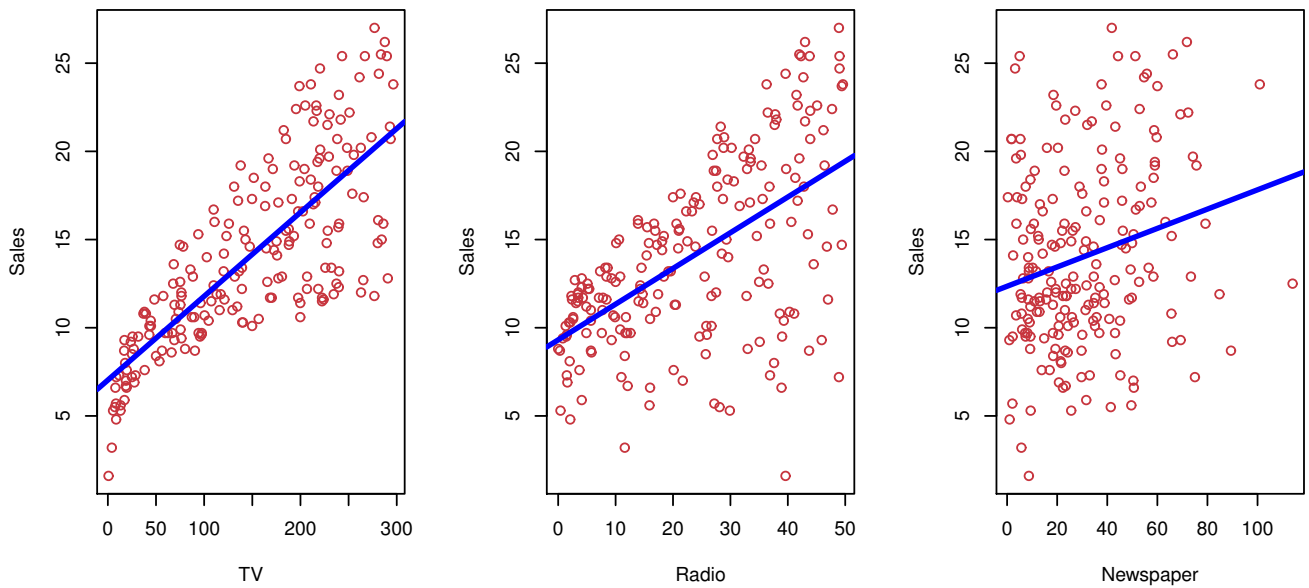
- Build a foundation for practice and research in ML
- Basic machine learning concepts: max likelihood, cross validation
- Fundamental machine learning techniques: regression, model-selection, deep learning
- Educational goals:
  1. How to apply basic methods
  2. Reveal what happens inside
  3. What are the pitfalls
  4. Expand understanding of linear algebra, statistics, and optimization

What is Machine Learning

- Discover unknown function f:

$$Y = f(X)$$

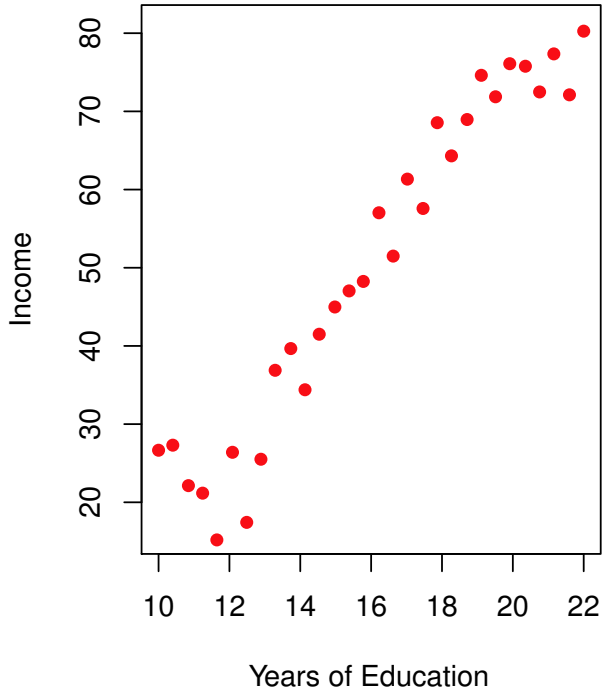
- X = set of features, or inputs
- Y = target, or response

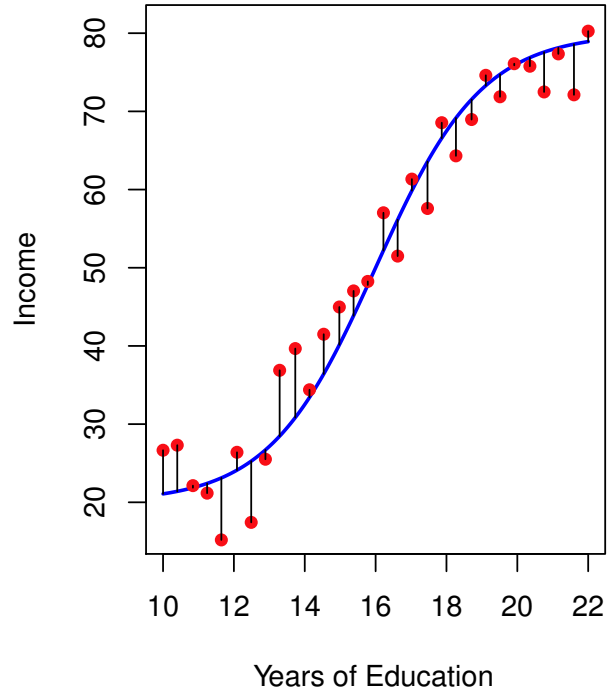


$$\text{Sales} = f(\text{TV}, \text{Radio}, \text{Newspaper})$$

Statistical View of Machine Learning

- Probability space Ω: Set of all adults
- Random variable: $X(\omega) = \mathbb{R}$: Years of education
- Random variable: $Y(\omega) = \mathbb{R}$: Salary





How Good are Predictions?

- Learned function $\hat{f}$
- Test data: $(x_1, y_1), (x_2, y_2), \ldots$
- **Mean Squared Error (MSE)**:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2$$

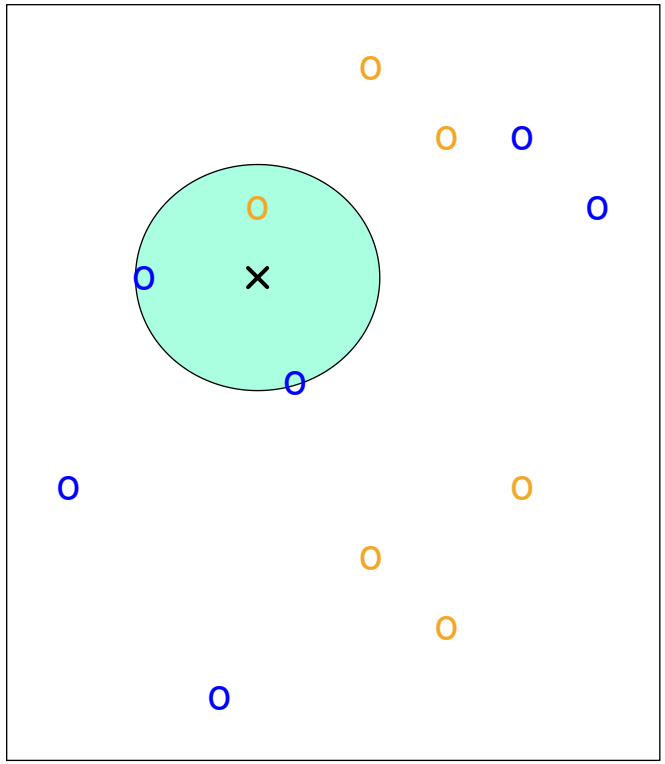
- This is the estimate of:

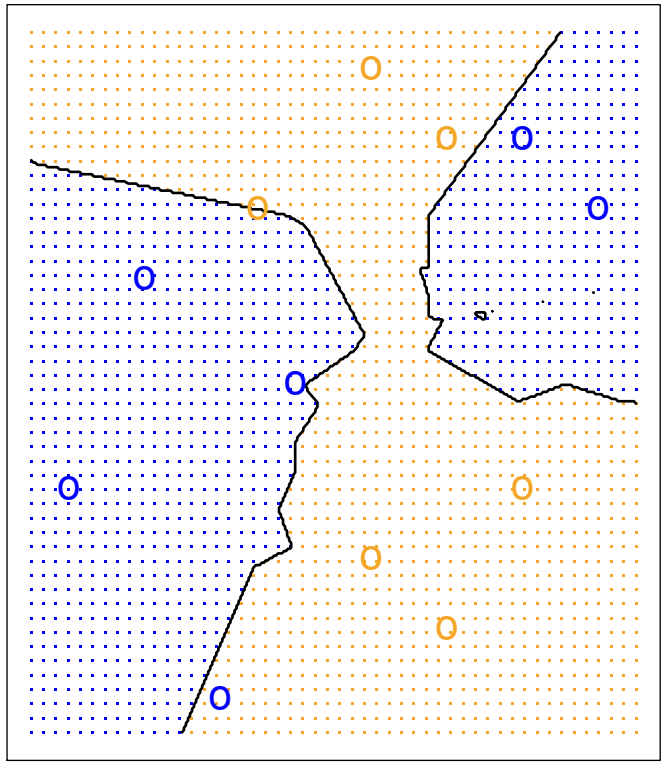
$$\text{MSE} = \mathbb{E}[(Y - \hat{f}(X))^2] = \frac{1}{|\Omega|} \sum_{\omega \in \Omega} (Y(\omega) - \hat{f}(X(\omega)))^2$$

- Important: Samples x_i are i.i.d.

KNN: K-Nearest Neighbors

- Idea: Use similar training points when making predictions





- Non-parametric method (unlike regression)

Bias-Variance Decomposition

$$Y = f(X) + \epsilon$$

Mean Squared Error can be decomposed as:

$$\text{MSE} = \mathbb{E}(Y - \hat{f}(X))^2 = \underbrace{\text{Var}(\hat{f}(X))}_{\text{Variance}} + \underbrace{(\mathbb{E}(\hat{f}(X)))^2}_{\text{Bias}} + \text{Var}(\epsilon)$$

- **Bias**: How well would method work with infinite data
- **Variance**: How much does output change with different data sets

R^2 Statistic

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

- RSS - residual sum of squares, TSS - total sum of squares
- R^2 measures the goodness of the fit as a proportion
- Proportion of data variance explained by the model
- Extreme values:

  0: Model does not explain data

  1: Model explains data perfectly

Correlation Coefficient

- Measures dependence between two random variables X and Y

$$r = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}}$$

- Correlation coefficient r is between $[-1, 1]$
  - 0: Variables are not related
  - 1: Variables are perfectly related (same)
  - -1: Variables are negatively related (different)

Correlation Coefficient

- Measures dependence between two random variables X and Y

$$r = \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)}\sqrt{\mathrm{Var}(Y)}}$$

- Correlation coefficient r is between $[-1, 1]$
  - 0: Variables are not related
  - 1: Variables are perfectly related (same)
  - -1: Variables are negatively related (different)
- $R^2 = r^2$

Qualitative Features: Many Values The Right Way

- Predict salary as a function of state
- Feature $\text{state}_i \in \{\text{MA}, \text{NH}, \text{ME}\}$

Qualitative Features: Many Values The Right Way

- Predict salary as a function of state
- Feature $\text{state}_i \in \{\text{MA}, \text{NH}, \text{ME}\}$
- Introduce 2 **indicator variables** x_i, z_i:

$$x_i = \begin{cases} 0 & \text{if } \text{state}_i = \text{MA} \\ 1 & \text{if } \text{state}_i \neq \text{MA} \end{cases} \qquad z_i = \begin{cases} 0 & \text{if } \text{state}_i = \text{NH} \\ 1 & \text{if } \text{state}_i \neq \text{NH} \end{cases}$$

- Predict salary as:

$$\text{salary} = \beta_0 + \beta_1 \times x_i + \beta_2 \times z_i = \begin{cases} \beta_0 + \beta_1 & \text{if } \text{state}_i = \text{MA} \\ \beta_0 + \beta_2 & \text{if } \text{state}_i = \text{NH} \\ \beta_0 & \text{if } \text{state}_i = \text{ME} \end{cases}$$

Qualitative Features: Many Values The Right Way

- Predict salary as a function of state
- Feature $\text{state}_i \in \{\text{MA}, \text{NH}, \text{ME}\}$
- Introduce 2 **indicator variables** x_i, z_i:

$$x_i = \begin{cases} 0 & \text{if } \text{state}_i = \text{MA} \\ 1 & \text{if } \text{state}_i \neq \text{MA} \end{cases} \qquad z_i = \begin{cases} 0 & \text{if } \text{state}_i = \text{NH} \\ 1 & \text{if } \text{state}_i \neq \text{NH} \end{cases}$$

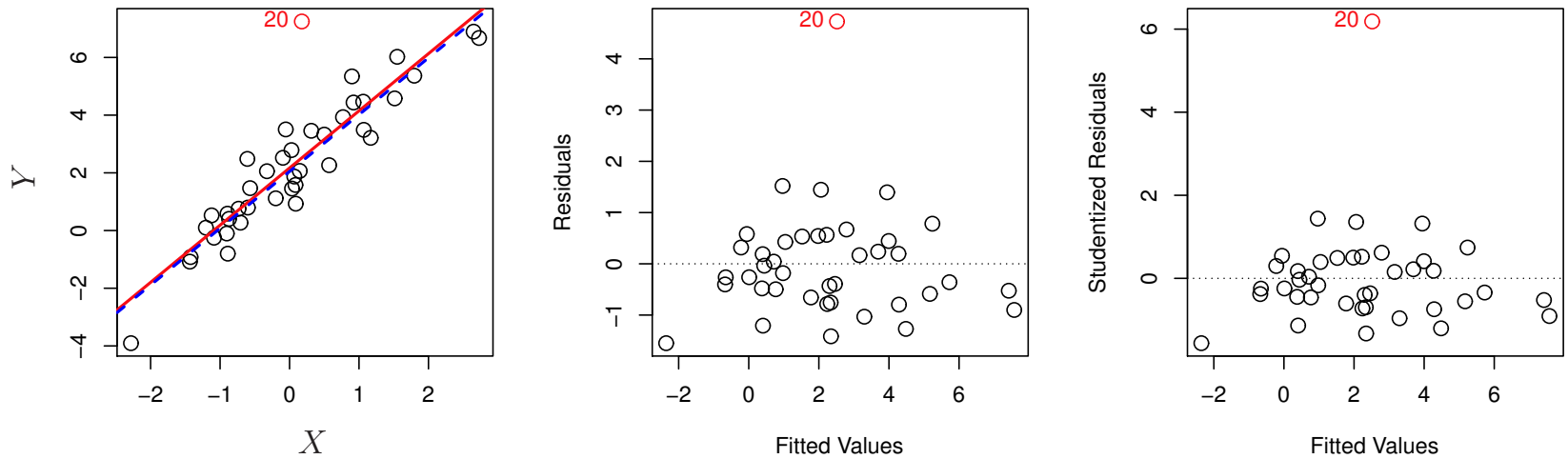
- Predict salary as:

$$\text{salary} = \beta_0 + \beta_1 \times x_i + \beta_2 \times z_i = \begin{cases} \beta_0 + \beta_1 & \text{if } \text{state}_i = \text{MA} \\ \beta_0 + \beta_2 & \text{if } \text{state}_i = \text{NH} \\ \beta_0 & \text{if } \text{state}_i = \text{ME} \end{cases}$$

- Need an indicator variable for ME? Why? hint: linear independence

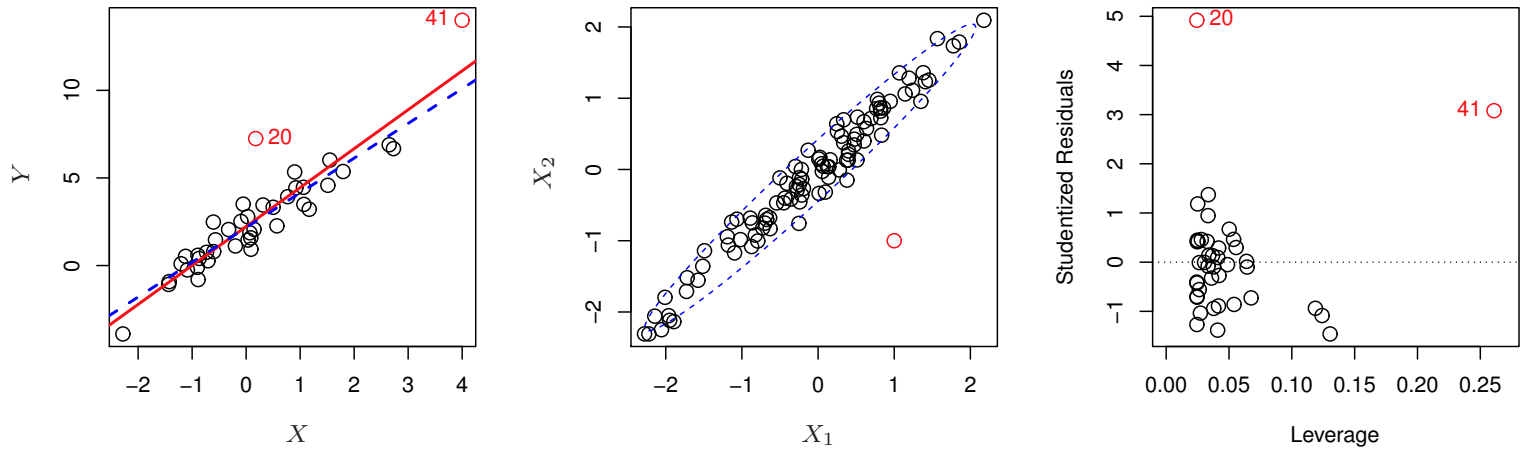
Outlier Data Points

- Data point that is far away from others
- Measurement failure, sensor fails, missing data point
- Can seriously influence prediction quality



Points with High Leverage

- Points with unusual value of x_i
- Single data point can have significant impact on prediction
- R and other packages can compute leverages of data points



- Good to remove points with high leverage and residual

Best Subset Selection

- Want to find a subset of p features
- The subset should be small and predict well
- Example: credit $\sim$ rating $+$ income $+$ student $+$ limit

$\mathcal{M}_0 \leftarrow$ *null model* (no features);
for $k = 1, 2, \ldots, p$ **do**
  Fit all $\binom{p}{k}$ models that contain k features ;
  $\mathcal{M}_k \leftarrow$ best of $\binom{p}{k}$ models according to a metric (CV error, R^2, etc)
end
return Best of $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_p$ according to metric above

Algorithm 1: Best Subset Selection

Regularization

- **Ridge regression** (parameter λ), ℓ_2 penalty

$$\min_{\beta} \mathrm{RSS}(\beta) + \lambda \sum_j \beta_j^2 =$$

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

- **Lasso** (parameter λ), ℓ_1 penalty

$$\min_{\beta} \mathrm{RSS}(\beta) + \lambda \sum_j |\beta_j| =$$

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

- Approximations to the ℓ_0 solution

Logistic Regression

- Predict **probability** of a class: $p(X)$
- Example: $p(\text{balance})$ probability of default for person with balance
- **Linear regression**:

$$p(X) = \beta_0 + \beta_1$$

- **logistic regression**:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

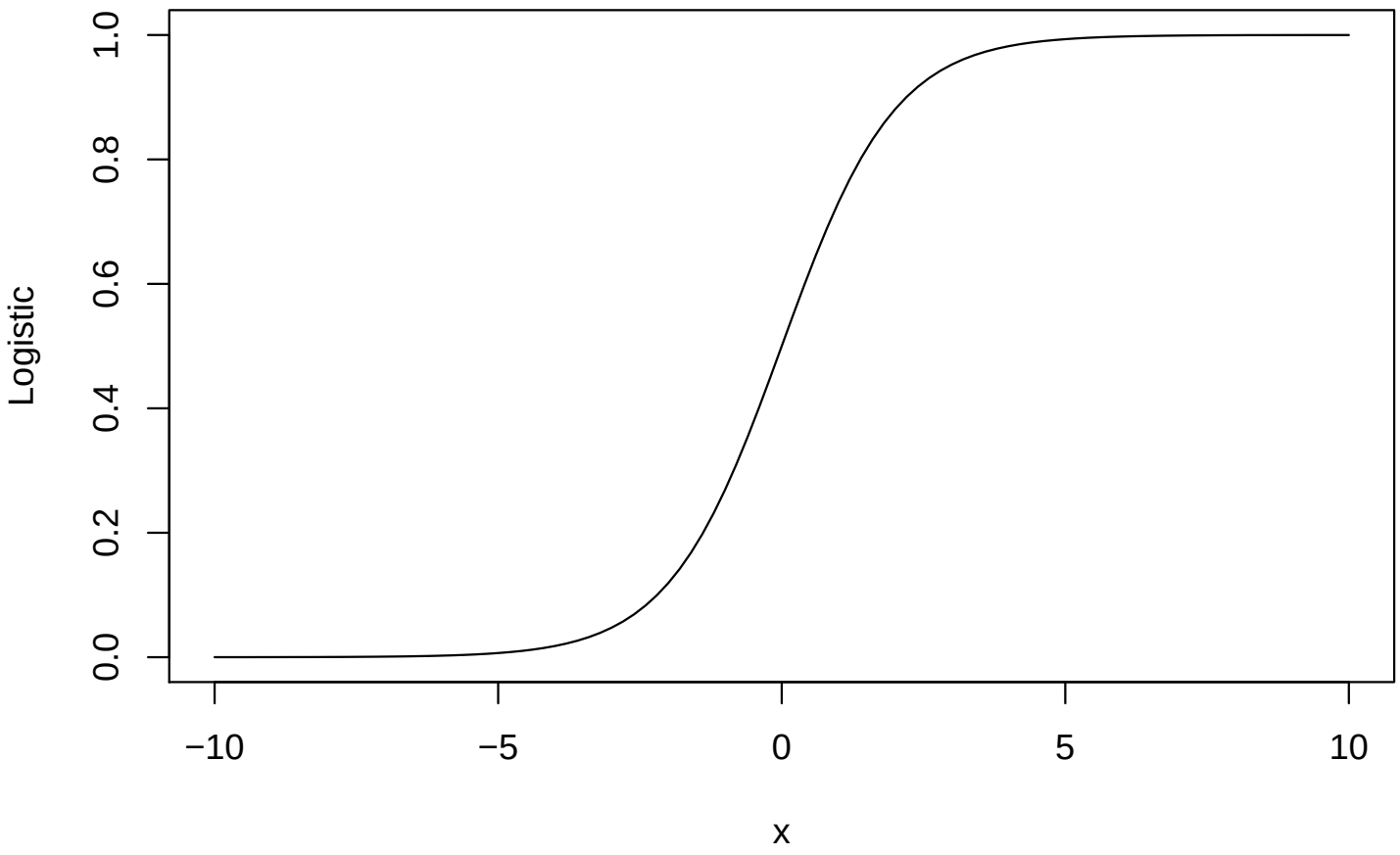
- the same as:

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

- Odds: $p(X)/1-p(X)$

Logistic Function

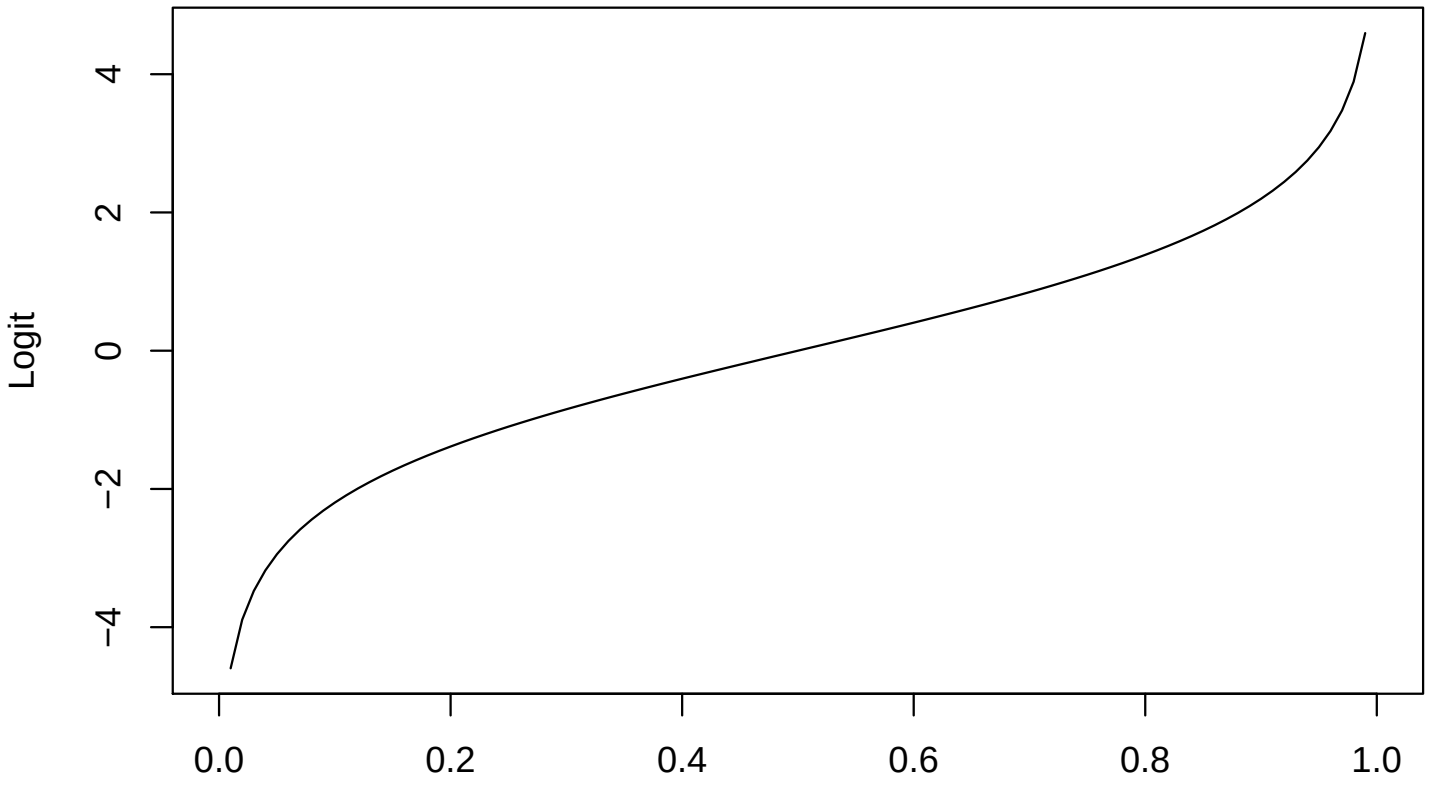
$$y = \frac{e^x}{1 + e^x}$$



$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

Logit Function

$$\log\left(\frac{p(X)}{1-p(X)}\right)$$



$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

Estimating Coefficients: Maximum Likelihood

- **Likelihood**: Probability that data is generated from a model

$$\ell(\text{model}) = \Pr[\text{data} \mid \text{model}]$$

- Find the most likely model:

$$\max_{\text{model}} \ell(\text{model}) = \max_{\text{model}} \Pr[\text{data} \mid \text{model}]$$

- Likelihood function is difficult to maximize
- Transform it using $\log$ (strictly increasing)

$$\max_{\text{model}} \log \ell(\text{model})$$

- Strictly increasing transformation does not change maximum

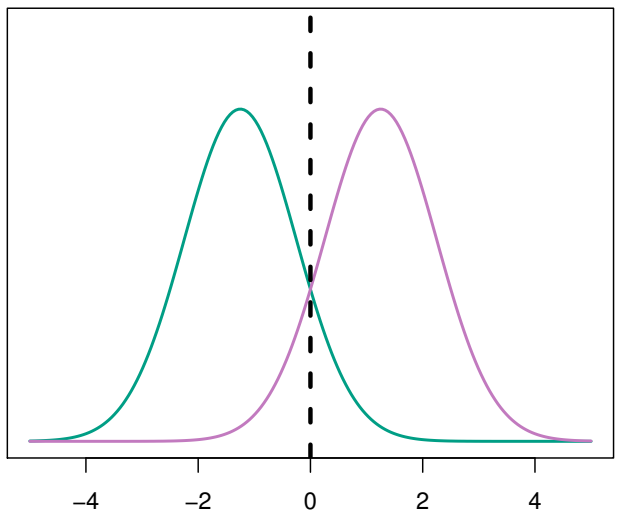
Discriminative vs Generative Models

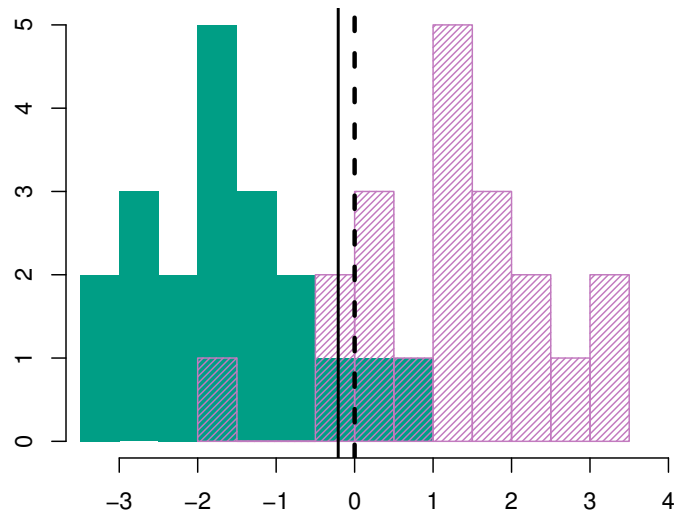
- **Discriminative models**
  - Estimate conditional models $\Pr[Y \mid X]$
  - Linear regression
  - Logistic regression

- **Generative models**
  - Estimate joint probability $\Pr[Y, X] = \Pr[Y \mid X] \Pr[X]$
  - Estimates not only probability of labels but also the features
  - Once model is fit, can be used to generate data
  - LDA, QDA, Naive Bayes

LDA: Linear Discriminant Analysis

- **Generative model**: capture probability of predictors for each label

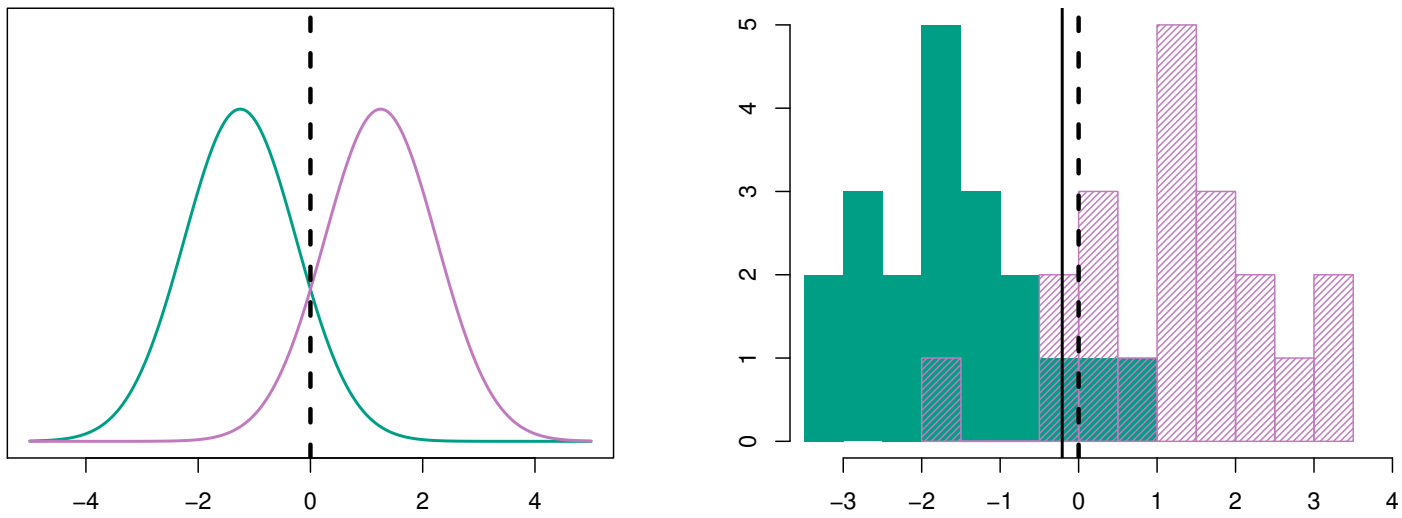




- Predict:

LDA: Linear Discriminant Analysis

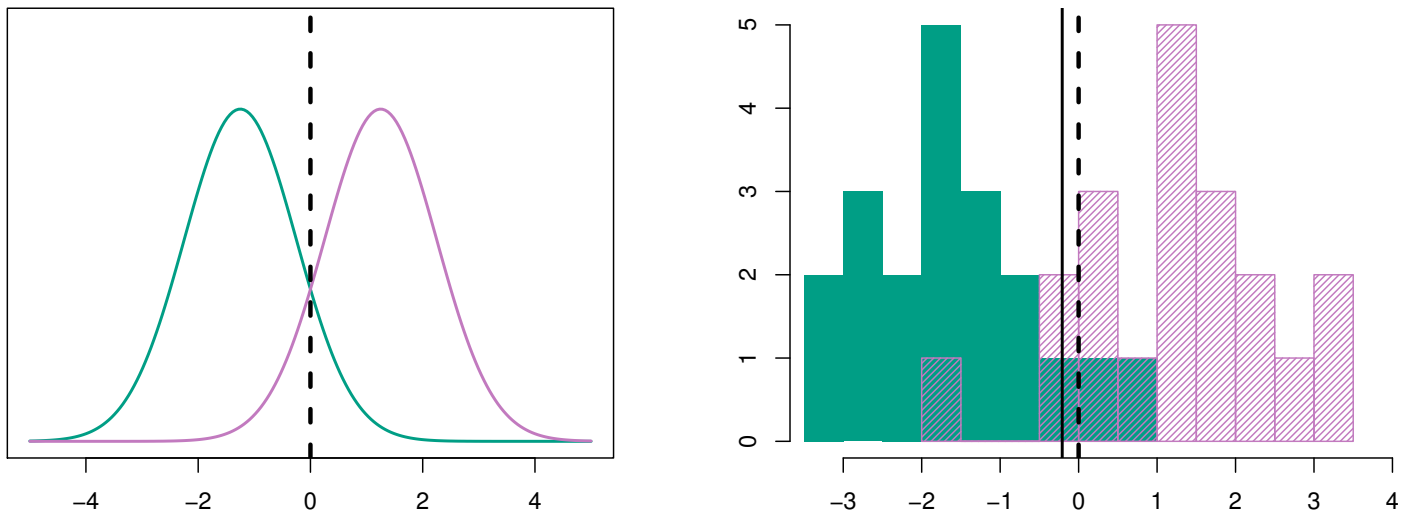
- **Generative model**: capture probability of predictors for each label



- Predict:
  1. $\Pr[\text{balance} \mid \text{default} = \text{yes}]$ and $\Pr[\text{default} = \text{yes}]$

LDA: Linear Discriminant Analysis

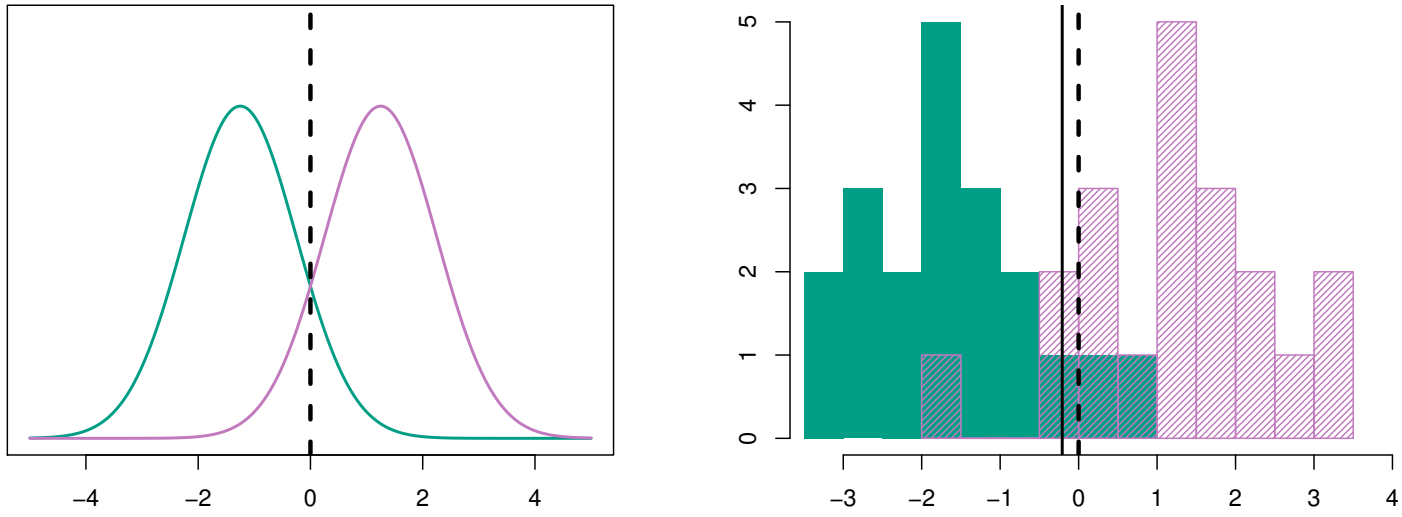
- **Generative model**: capture probability of predictors for each label



- Predict:
  1. $\Pr[\text{balance} \mid \text{default} = \text{yes}]$ and $\Pr[\text{default} = \text{yes}]$
  2. $\Pr[\text{balance} \mid \text{default} = \text{no}]$ and $\Pr[\text{default} = \text{no}]$

LDA: Linear Discriminant Analysis

- **Generative model**: capture probability of predictors for each label



- Predict:
  1. $\Pr[\text{balance} \mid \text{default} = \text{yes}]$ and $\Pr[\text{default} = \text{yes}]$
  2. $\Pr[\text{balance} \mid \text{default} = \text{no}]$ and $\Pr[\text{default} = \text{no}]$
- Classes are normal: $\Pr[\text{balance} \mid \text{default} = \text{yes}]$

QDA: Quadratic Discriminant Analysis

- Generalizes LDA

- **LDA**: Class variances $\Sigma_k = \Sigma$ are the same
- **QDA**: Class variances Σ_k can differ

QDA: Quadratic Discriminant Analysis

- Generalizes LDA

- **LDA**: Class variances $\Sigma_k = \Sigma$ are the same
- **QDA**: Class variances Σ_k can differ

- LDA or QDA has smaller training error on the same data?

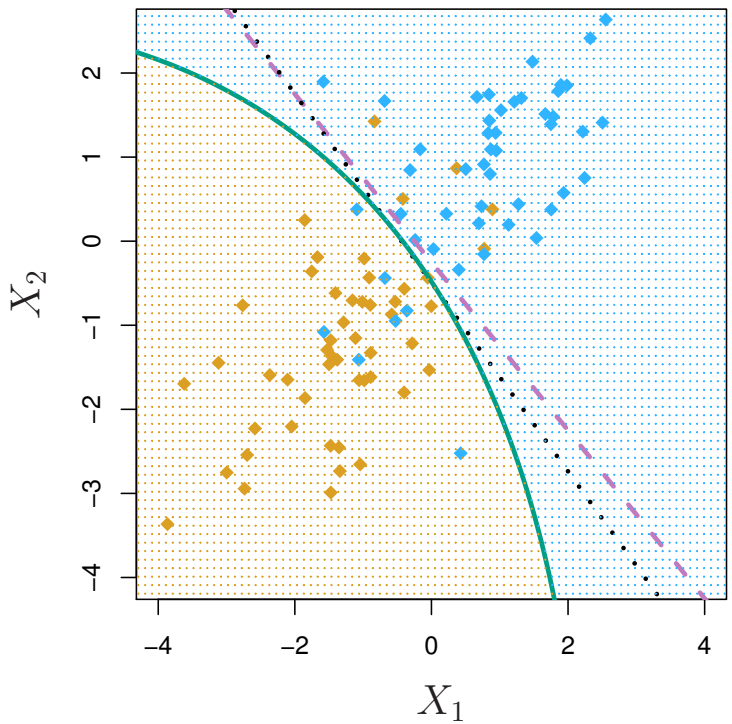
QDA: Quadratic Discriminant Analysis

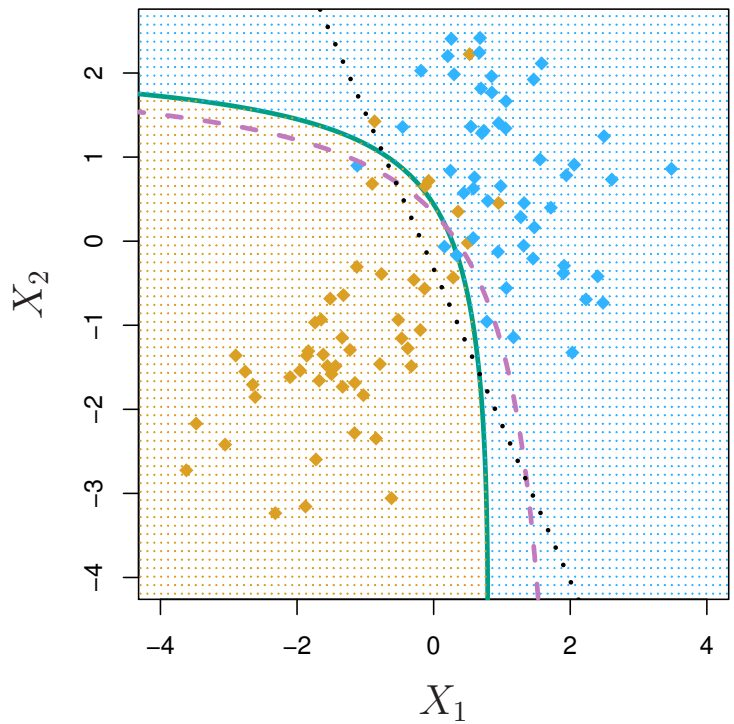
- Generalizes LDA

- **LDA**: Class variances $\Sigma_k = \Sigma$ are the same
- **QDA**: Class variances Σ_k can differ

- LDA or QDA has smaller training error on the same data?
- What about the test error?

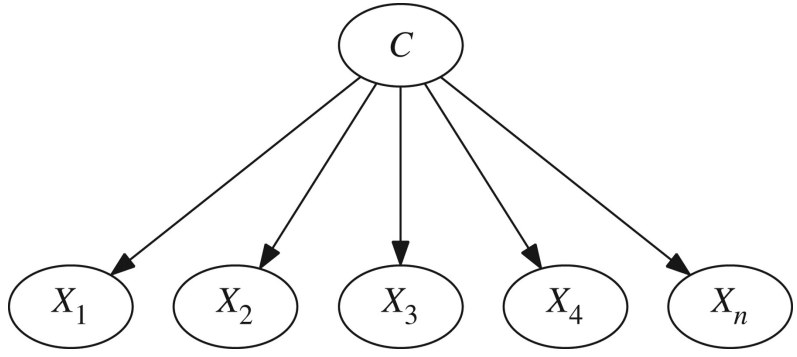
QDA: Quadratic Discriminant Analysis





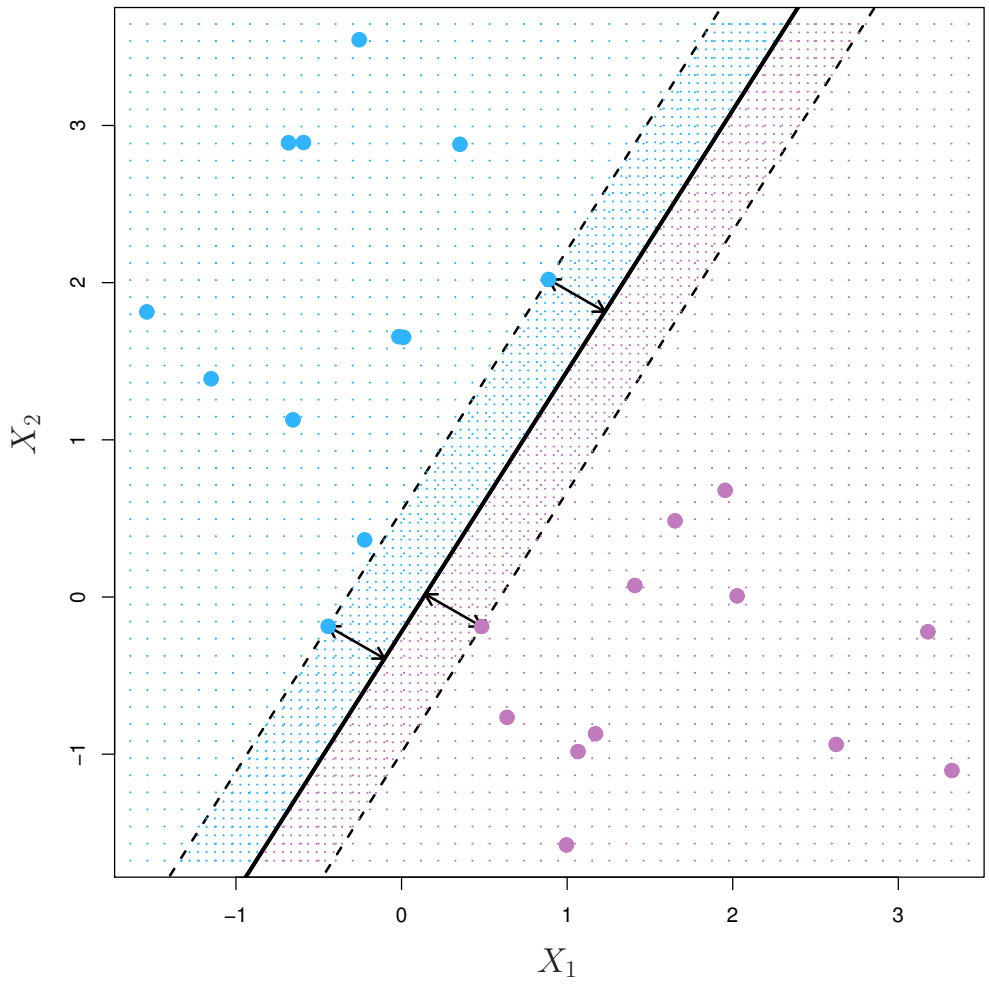
Naive Bayes

- Simple Bayes net classification



- With normal distribution over features $X_1, \ldots, X_k$ special case of QDA with diagonal Σ
- Generalizes to non-Normal distributions and discrete variables
- More on it later ...

Maximum Margin Hyperplane



Introducing Slack Variables

- **Maximum margin classifier**

$$\begin{aligned} \max_{\beta, M} \quad & M \\ \text{s.t.} \quad & y_i(\beta^\top x_i) \geq M \\ & \|\beta\|_2 = 1 \end{aligned}$$

- **Support Vector Classifier** a.k.a Linear SVM

$$\begin{aligned} \max_{\beta, M, \epsilon \geq 0} \quad & M \\ \text{s.t.} \quad & y_i(\beta^\top x_i) \geq (M - \epsilon_i) \\ & \|\beta\|_2 = 1 \\ & \|\epsilon\|_1 \leq C \end{aligned}$$

- Slack variables: ϵ
- Parameter: C

Introducing Slack Variables

- **Maximum margin classifier**

$$\begin{aligned} \max_{\beta, M} \quad & M \\ \text{s.t.} \quad & y_i(\beta^\top x_i) \geq M \\ & \|\beta\|_2 = 1 \end{aligned}$$

- **Support Vector Classifier** a.k.a Linear SVM

$$\begin{aligned} \max_{\beta, M, \epsilon \geq 0} \quad & M \\ \text{s.t.} \quad & y_i(\beta^\top x_i) \geq (M - \epsilon_i) \\ & \|\beta\|_2 = 1 \\ & \|\epsilon\|_1 \leq C \end{aligned}$$

- Slack variables: ϵ
- Parameter: C What if $C = 0$?

Kernelized SVM

- **Dual Quadratic Program** (usually max-min, not here)

$$\max_{\alpha \geq 0} \quad \sum_{l=1}^{M} \alpha_l - \frac{1}{2} \sum_{j,k=1}^{M} \alpha_j \alpha_k y_j y_k k(x_j, x_k)$$

$$\text{s.t.} \quad \sum_{l=1}^{M} \alpha_l y_l = 0$$

- **Representer theorem**: (classification test):

$$f(z) = \sum_{l=1}^{M} \alpha_l y_l k(z, x_l) > 0$$

Kernels

- Polynomial kernel

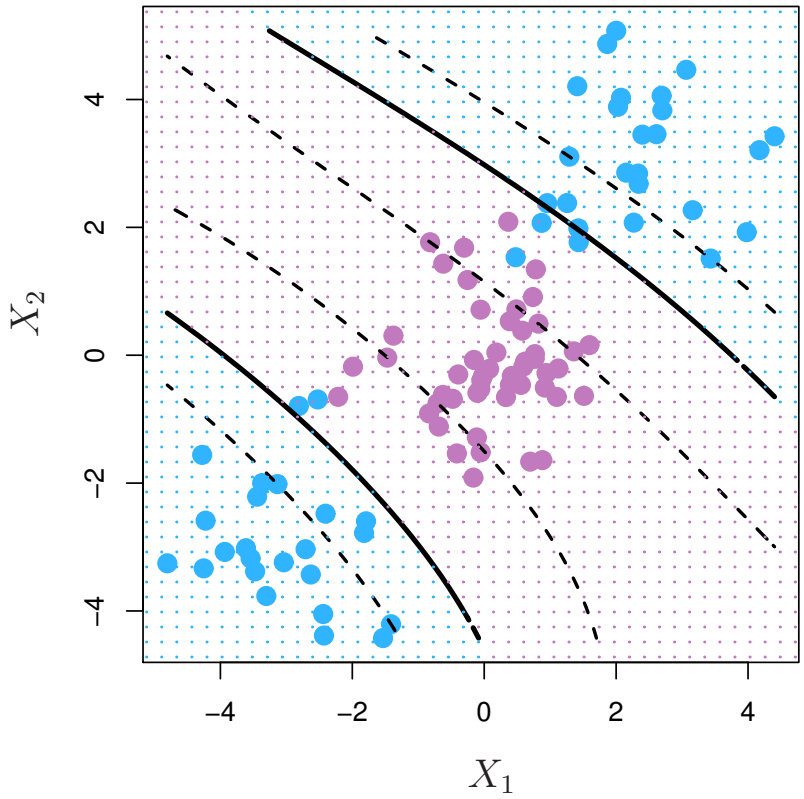
$$k(x_1, x_2) = \left(1 + x_1^\top x_2\right)^d$$

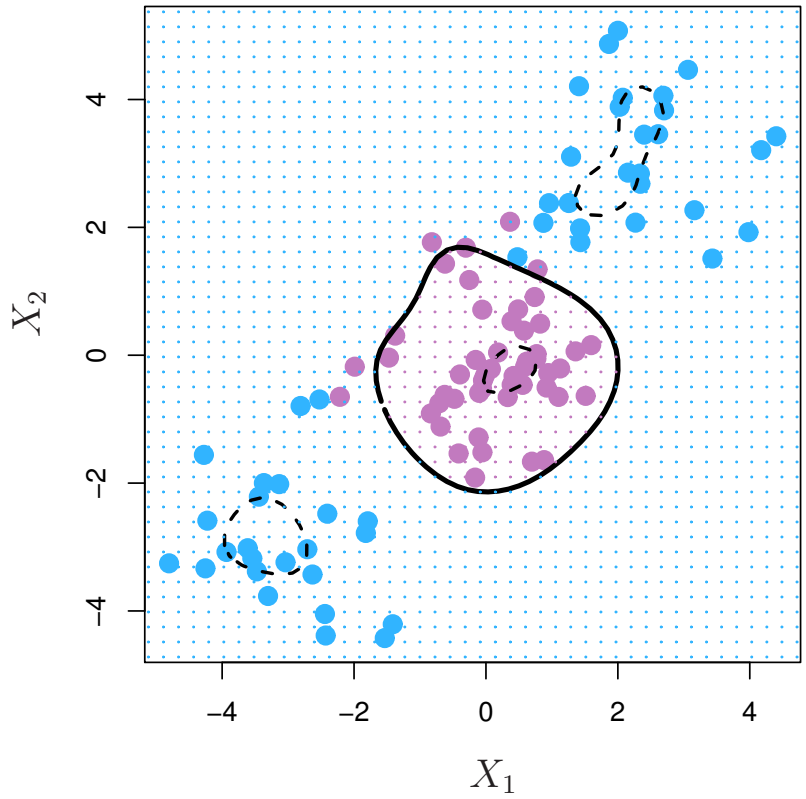
- Radial kernel

$$k(x_1, x_2) = \exp\left(-\gamma \|x_1 - x_2\|_2^2\right)$$

- Many many more

Polynomial and Radial Kernels





Regression Trees

- Predict Baseball Salary based on Years played and Hits
- Example:

CART: Recursive Binary Splitting

- Greedy top-to-bottom approach
- Recursively divide regions to minimize RSS

$$\sum_{x_i \in R_1} (y_i - \bar{y}_1)^2 + \sum_{x_i \in R_2} (y_i - \bar{y}_2)^2$$

- Prune tree

Trees vs. KNN

- Trees do not require a distance metric
- Trees work well with categorical predictors
- Trees work well in large dimensions
- KNN are better in low-dimensional problems with complex decision boundaries

Bagging

- Stands for "Bootstrap Aggregating"
- Construct multiple bootstrapped training sets:

$$T_1, T_2, \ldots, T_B$$

- Fit a tree to each one:

$$\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_B$$

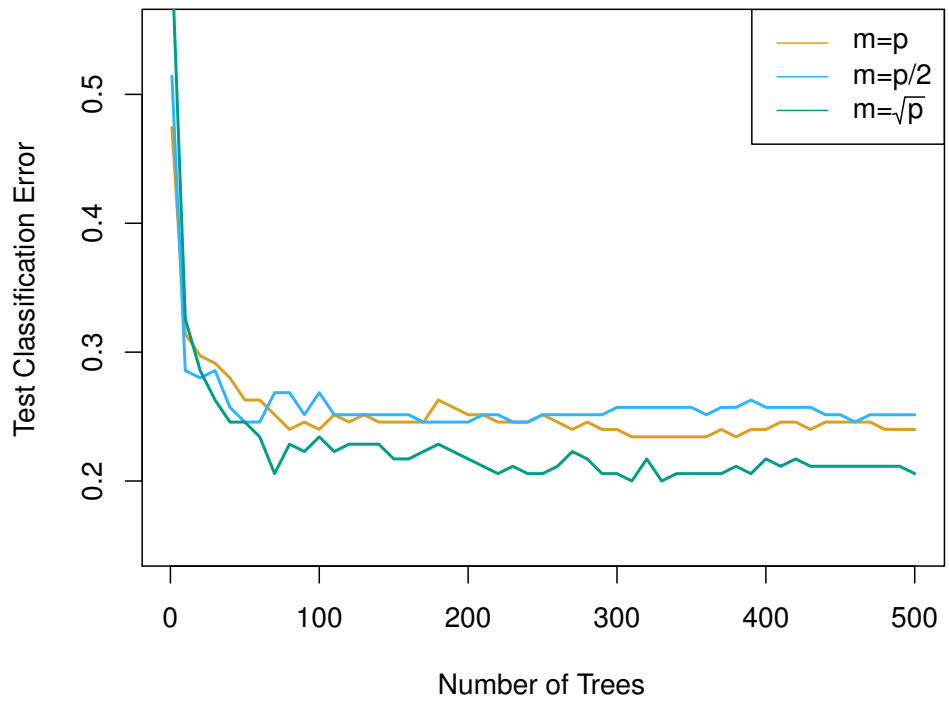
- Make predictions by averaging individual tree predictions

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x)$$

- Large values of B are not likely to overfit, $B \approx 100$ is a good choice

Random Forests

- Many trees in bagging will be similar
- Algorithms choose the same features to split on
- Random forests help to address similarity:
  - At each split, choose only from m randomly sampled features
- Good empirical choice is $m = \sqrt{p}$



Gradient Boosting (Regression)

- Boosting uses all of data, not a random subset (usually)
- Also builds trees $\hat{f}_1, \hat{f}_2, \ldots$
- and weights $\lambda_1, \lambda_2, \ldots$
- Combined prediction:

$$\hat{f}(x) = \sum_i \lambda_i \hat{f}_i(x)$$

- Assume we have $1 \ldots m$ trees and weights, next best tree?

Gradient Boosting (Regression)

- Just use gradient descent
- **Objective** is to minimize RSS (1/2):

$$\frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

- **Objective** with the new tree $m+1$:

$$\frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{m}\hat{f}_j(x_i) - \hat{f}_{m+1}(x_i)\right)^2$$

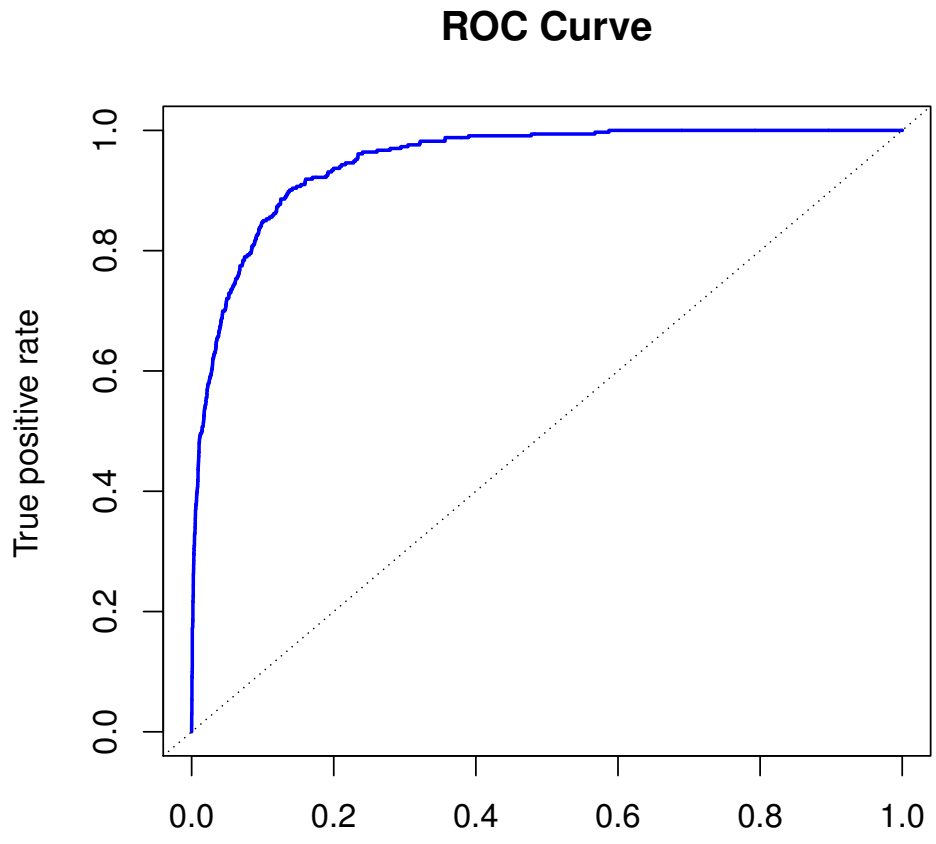
- Greatest reduction in RSS: gradient

$$y_i - \sum_{j=1}^{m}\hat{f}_j(x_i) \approx \hat{f}_{m+1}(x_i)$$

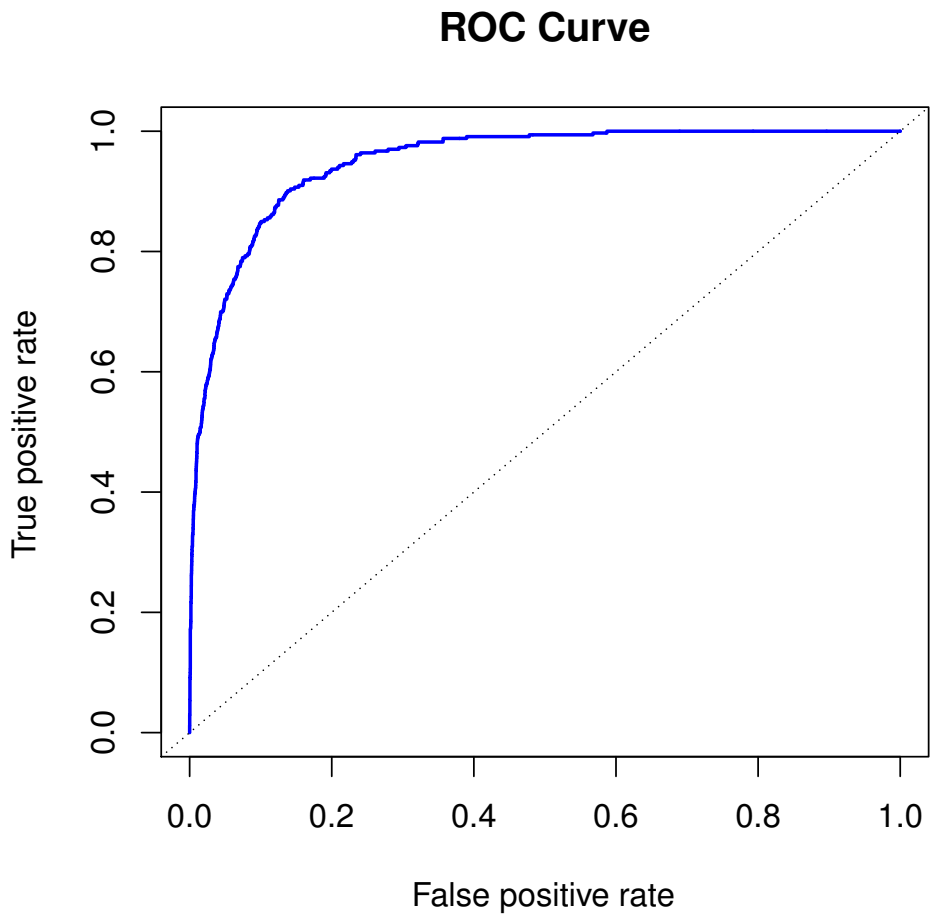
ROC Curve

Confusion matrix

		Reality	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative



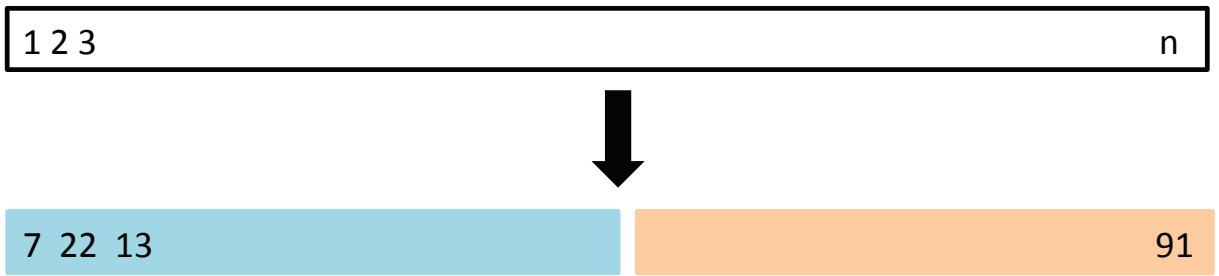
Area Under ROC Curve



- Larger area is better
- Many other ways to measure classifier performance, like F_1

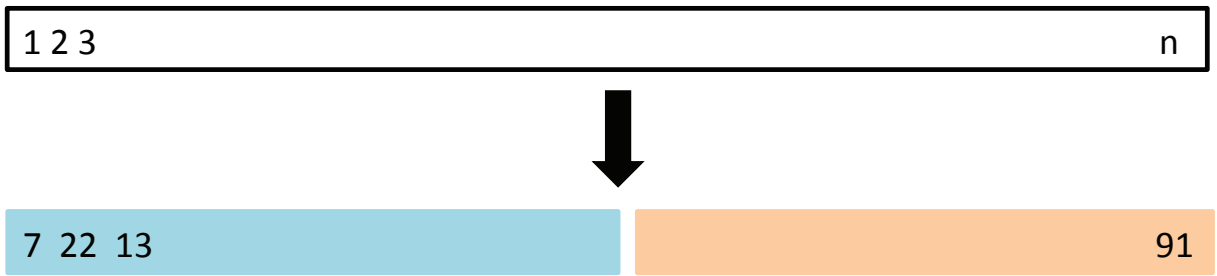
Evaluation Method 1: Validation Set

- Just evaluate how well the method works on the test set
- **Randomly** split data to:
  1. Training set: about half of all data
  2. Validation set (AKA hold-out set): remaining half



Evaluation Method 1: Validation Set

- Just evaluate how well the method works on the test set
- **Randomly** split data to:
  1. Training set: about half of all data
  2. Validation set (AKA hold-out set): remaining half



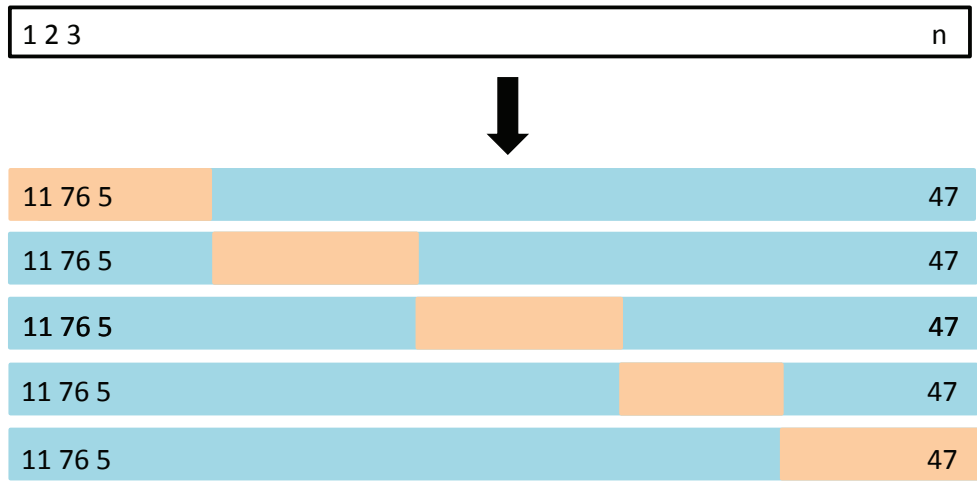
- Choose the number of features/representation based on minimizing error on **validation set**

Evaluation Method 2: Leave-one-out

- Addresses problems with validation set
- Split the data set into 2 parts:
  1. Training: Size $n - 1$
  2. Validation: Size 1
- Repeat n times: to get n learning problems

Evaluation Method 3: k-fold Cross-validation

- Hybrid between validation set and LOO
- Split training set into k subsets
  1. Training set: $n - {}^{n}/_{k}$
  2. Test set: ${}^{n}/_{k}$
- k learning problems



- Cross-validation error:

$$\mathrm{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \mathrm{MSE}_i$$

Bootstrap

- **Goal**: Understand the confidence in learned parameters
- Most useful in inference
- How confident are we in learned values of β:

$$\text{mpg} = \beta_0 + \beta_1 \text{ power}$$

Bootstrap

- **Goal**: Understand the confidence in learned parameters
- Most useful in inference
- How confident are we in learned values of β:

$$\text{mpg} = \beta_0 + \beta_1 \, \text{power}$$

- **Approach**: Run learning algorithm multiple times with different data sets:

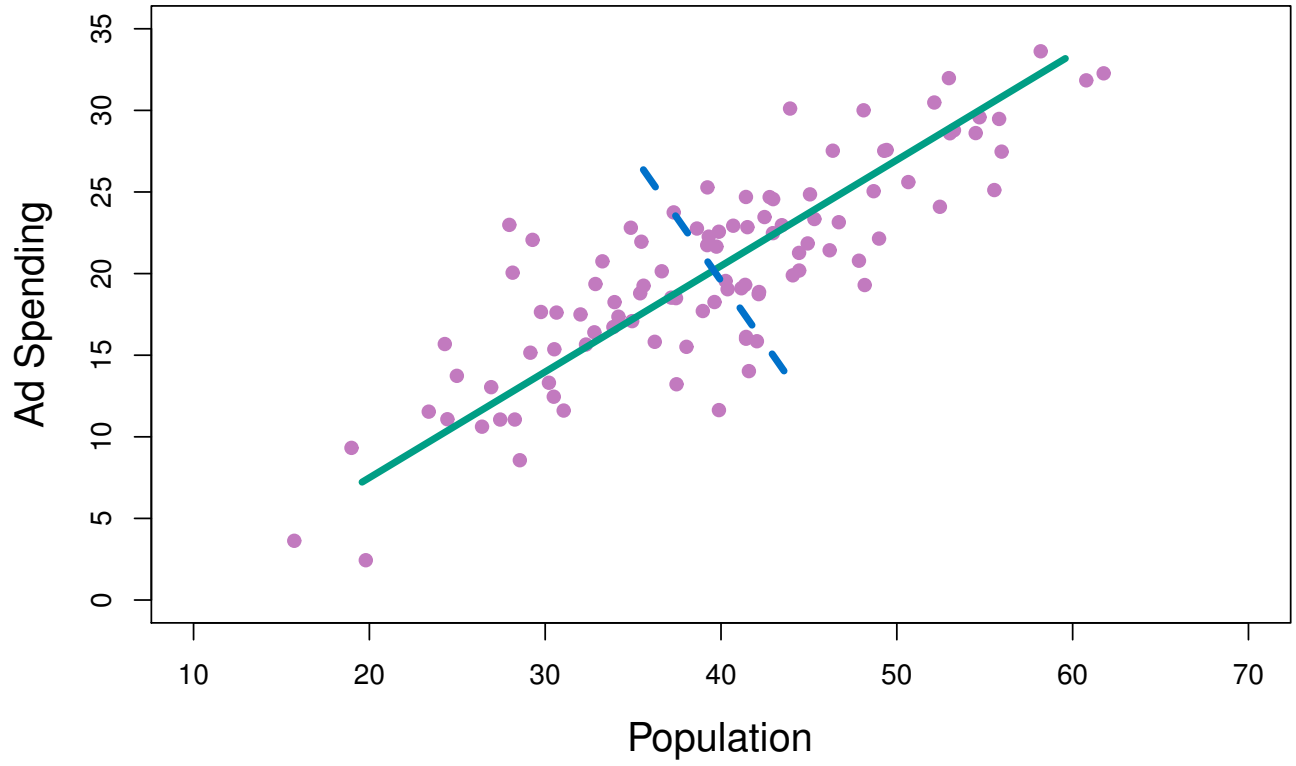
Bootstrap

- **Goal**: Understand the confidence in learned parameters
- Most useful in inference
- How confident are we in learned values of β:

$$\text{mpg} = \beta_0 + \beta_1 \, \text{power}$$

- **Approach**: Run learning algorithm multiple times with different data sets:
- Create a new data-set by sampling **with replacement** from the original one

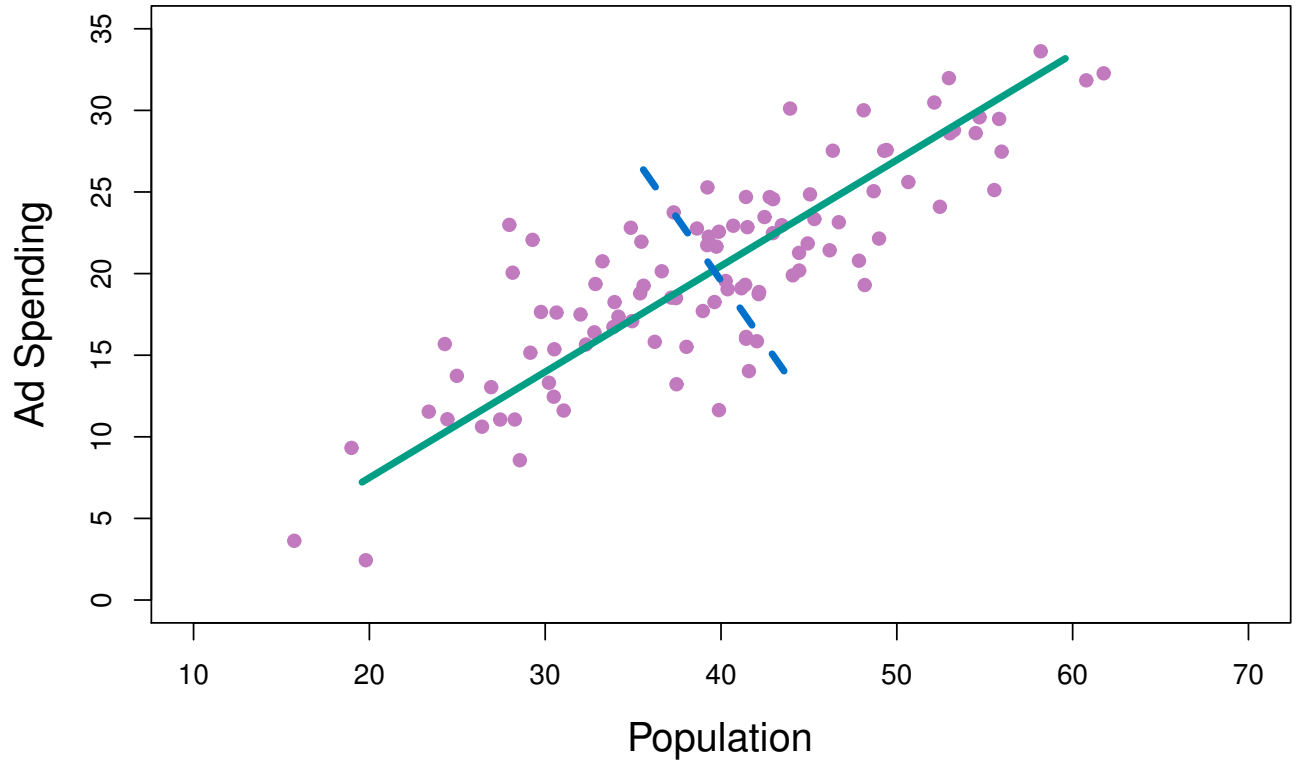
Principal Component Analysis



- **1st Principal Component**: Direction with the largest variance

$$Z_1 = 0.839 \times (\text{pop} - \overline{\text{pop}}) + 0.544 \times (\text{ad} - \overline{\text{ad}})$$

Principal Component Analysis

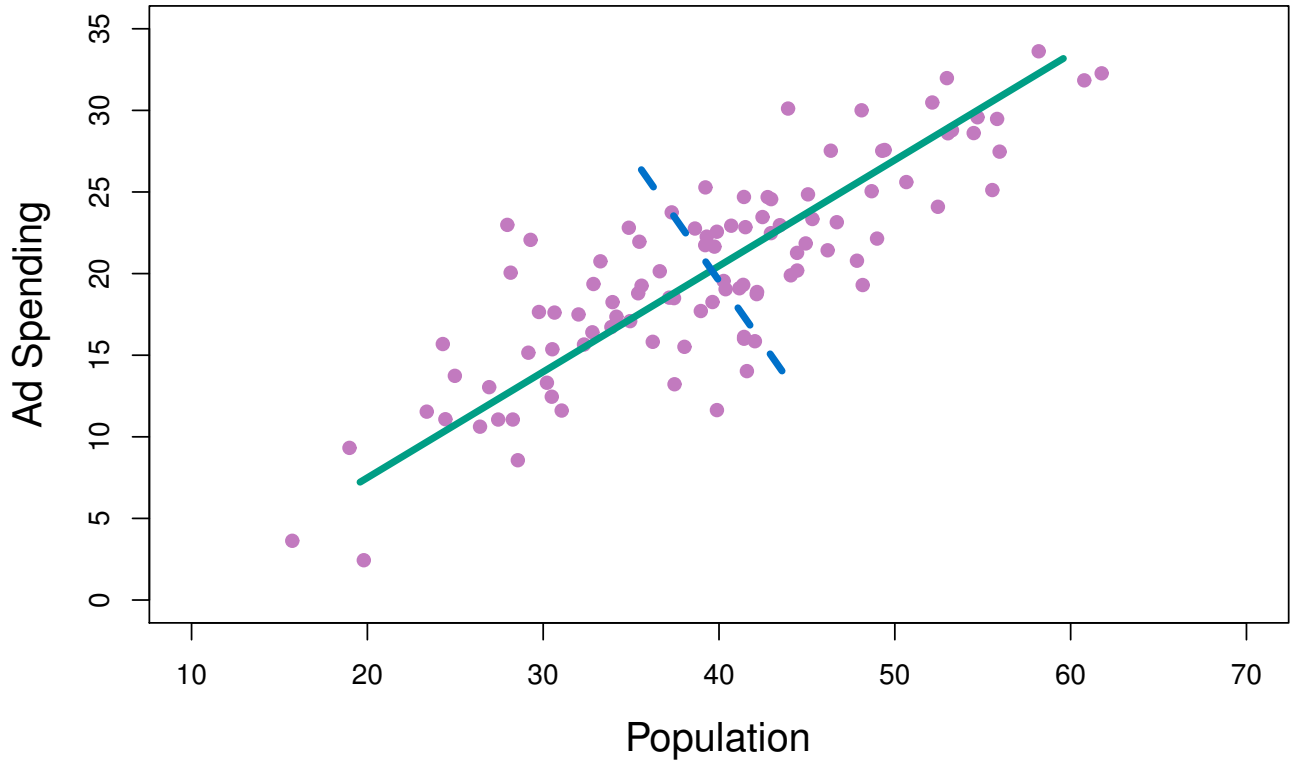


- **1st Principal Component**: Direction with the largest variance

$$Z_1 = 0.839 \times (\text{pop} - \overline{\text{pop}}) + 0.544 \times (\text{ad} - \overline{\text{ad}})$$

- Is this linear?

Principal Component Analysis



- **1st Principal Component**: Direction with the largest variance

$$Z_1 = 0.839 \times (\text{pop} - \overline{\text{pop}}) + 0.544 \times (\text{ad} - \overline{\text{ad}})$$

- Is this linear? Yes, after *mean centering*.

K-Means Algorithm

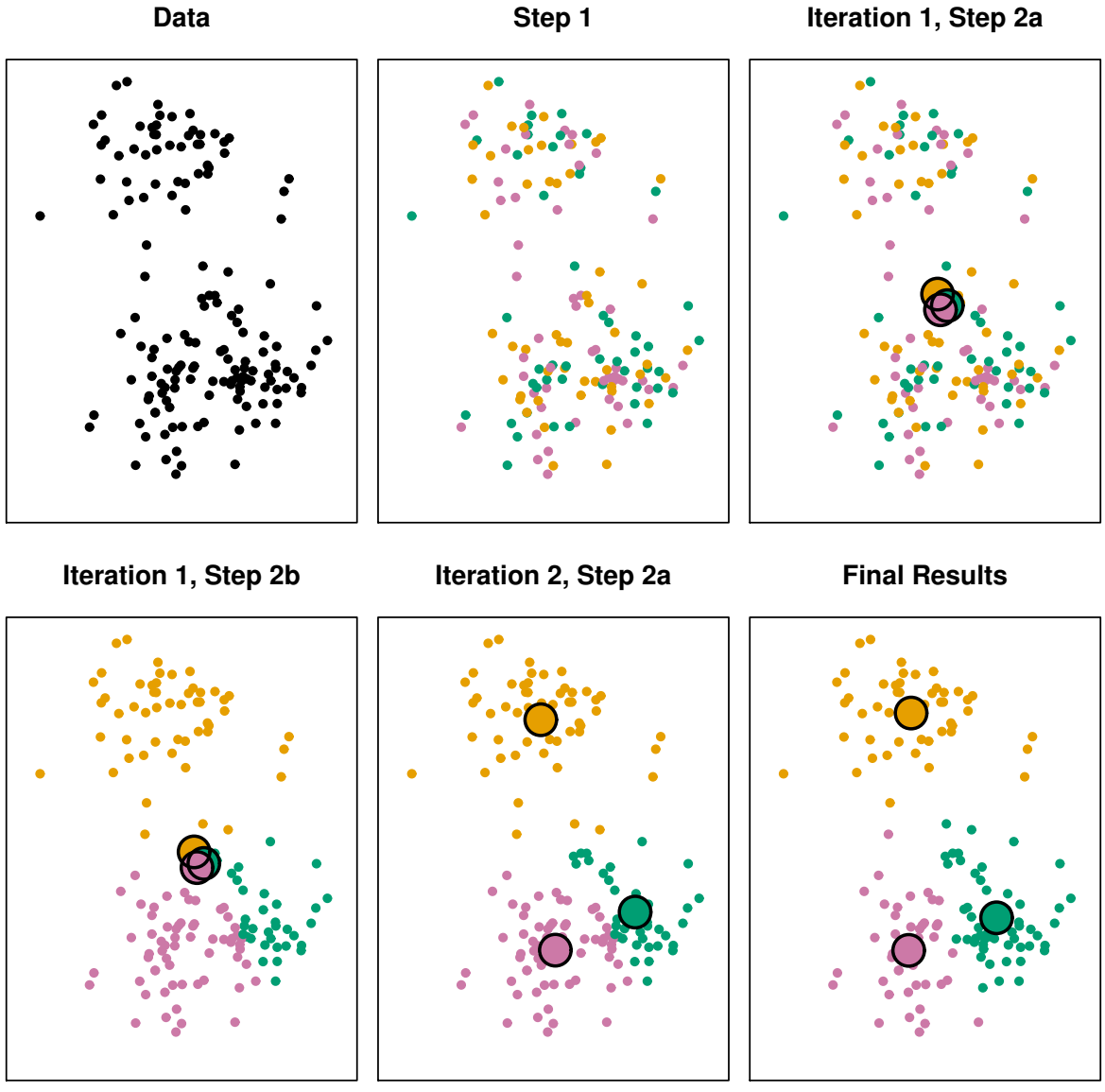
Heuristic solution to the minimization problem

1. Randomly assign cluster numbers to observations
2. Iterate while clusters change
   - 2.1 For each cluster, compute the centroid
   - 2.2 Assign each observation to the closest cluster

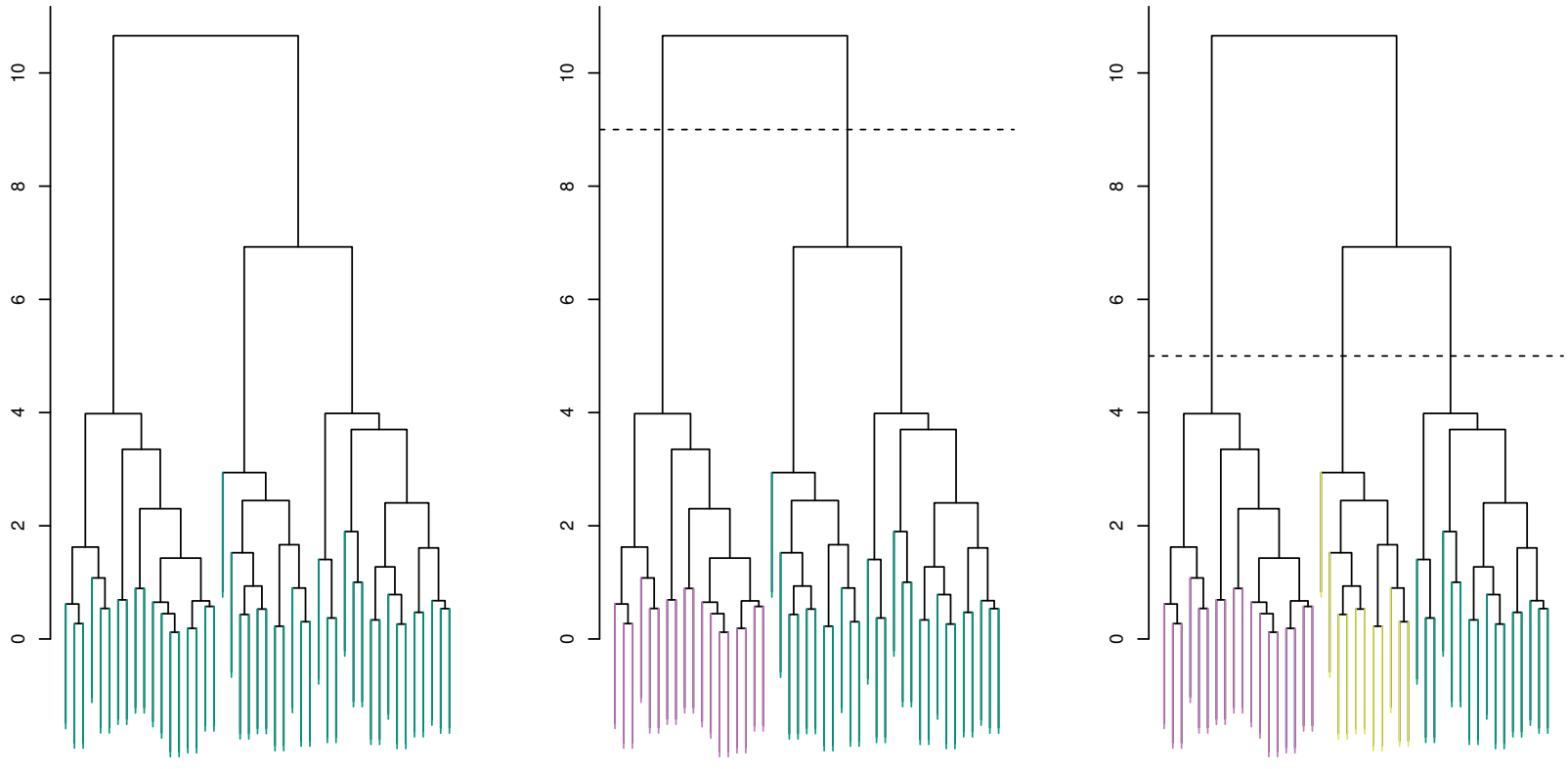
Note that:

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = 2 \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2$$

K-Means Illustration



Dendrogram: Similarity Tree



What Next?

- This course: How to use machine learning. More tools:
  - Gaussian processes
  - Time series models
  - Domain specific models (e.g., natural language processing)

What Next?

- This course: How to use machine learning. More tools:
  - Gaussian processes
  - Time series models
  - Domain specific models (e.g., natural language processing)

- Doing ML Research
  - Musts: Linear algebra, statistics, convex optimization
  - Important: Probably Approximately Correct Learning