STA 695: Statistical Machine Learning and Predictive Modeling

Zeya Wang

University of Kentucky

Fall 2024

Outline of the Course

1. Introduction to Machine Learning
2. Supervised Learning & Classification
3. Model Assessment
4. Model Selection
5. Data Preprocessing
6. Ensemble Methods
7. Neural Networks/Deep Learning
8. Unsupervised Learning
9. Intro to Extra Topics (e.g., Graphical Models)

Introduction to Machine Learning

- Machine learning: building statistical models and algorithms that allow computers to perform specific tasks on data

CS	STAT	Applied Math
Algorithm	Probability	Optimization
Database	Stat inference	Numerical analysis
	Sample size & complexity	

- Predictive modeling: the process of developing a mathematical tool or model that generates an accurate prediction

Introduction to Machine Learning

- Machine learning: building statistical models and algorithms that allow computers to perform specific tasks on data

CS	STAT	Common Areas
Deep Learning	Confidence Sets	Prediction (Regression & Classification)
Reinforcement Learning	Large Sample Theory	Probability Bounds
Efficient Computation	Statistical Optimality	Clustering
Large Language Model	Causality	Graphical Models

- Predictive modeling: the process of developing a mathematical tool or model that generates an accurate prediction

Introduction to Machine Learning

- Example 1: Email Spam. X_i: an email;

$$Y_i = \begin{cases} 0, & \textit{email} \\ 1, & \textit{spam} \end{cases}$$

- Feature extraction: e.g. frequency of certain key words
- A classification problem: 0-1
- Metric: misclassification rate

Introduction to Machine Learning

- Example 2. Predict the log of prostate-specific antigen (PSA) from a number of measurements
- A regression problem.
- Example 3. Handwritten digit recognition.
- X_i: a 16×16 eight-bit grayscale maps; $Y_i \in \{0, 1, \cdots, 9\}$
- Feature transformation: vectorized X_i or summary statistics, e.g., marginal histograms or numbers of crossing changes

Introduction to Machine Learning

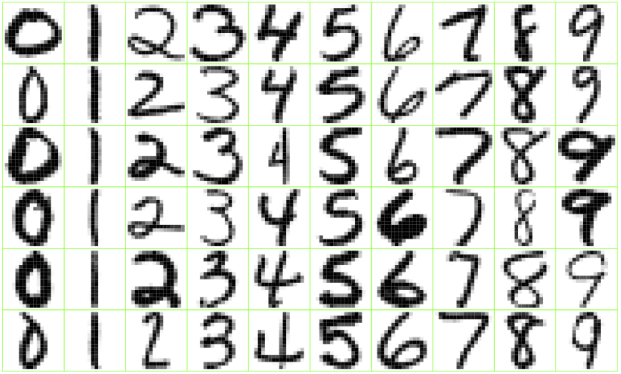


FIGURE 1.2. *Examples of handwritten digits from U.S. postal envelopes.*

Introduction to Machine Learning

- Example 4. Microarray gene expression data.
- X_i : gene expression levels
- Supervised learning: a classification problem when we have a label. Y_i: tumor types. (a typical small n, large p problem)
- Unsupervised learning: find subtypes of cancer when we don't have a label (clustering analysis)
- Semi-supervised learning: we have a label but possibly novel tumor subtypes in prediction

Introduction to Machine Learning

Introduction to Machine Learning

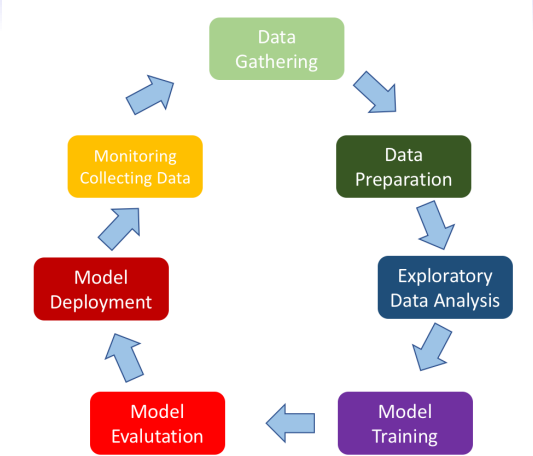


Figure: Life cycle of machine learning

Supervised Learning vs. Unsupervised Learning

- Supervised Learning: have labels and outcomes $\{Y_{n\times 1}, X_{n\times p}\}$
  - $X_{n\times p}$ Data Matrix (given or fixed)
  - $Y_{n\times 1}$ Outcomes: ordinal/quantitative (regression)/binary (classification)/categorical (classification)
- Unsupervised Learning
  - Clustering groups of samples/features
  - Pattern Recognition
  - Association between features

Training vs. Testing

- Training: fit a model $\{X^{train}, Y^{train}\}$
- Testing: assess the performance $\{X^{test}, Y^{test}\}$
- Overfitting: fit training data well, but terribly predict test data

Least Squares & Linear Regression

- Linear model

$$\begin{aligned}\hat{Y} &= \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j \\ &= X\hat{\beta}\end{aligned}$$

$X = [1, X_1, \cdots, X_p]$, $\hat{\beta} = [\hat{\beta}_0, \cdots, \hat{\beta}_p]^T$

- Residual sum of squares (RSS)

$$\begin{aligned}RSS(\beta) &= \sum_{i=1}^{N} (y_i - x_i^T \beta)^2 \\ &= (\boldsymbol{y} - \boldsymbol{X}\beta)^T (\boldsymbol{y} - \boldsymbol{X}\beta) \\ &= ||\boldsymbol{y} - \boldsymbol{X}\beta||_2^2\end{aligned}$$

Least Squares & Linear Regression

- Minimize RSS: $\min_\beta ||\boldsymbol{y} - \boldsymbol{X}\beta||_2^2$

$$\begin{aligned} \frac{\partial RSS}{\partial \beta} &= 0 \\ -\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\beta) &= 0 \\ \hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} & \end{aligned}$$

- Hat matrix H

$$\begin{aligned} \hat{Y} &= X\hat{\beta} \\ &= X(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \\ &= Hy \end{aligned}$$

Least Squares & Linear Regression

- Property of $\hat{\beta}$ (BLUE)

$$\begin{aligned} E[\hat{\beta}] &= \beta \\ Var(\hat{\beta}) &= (X^T X)^{-1}\sigma^2 \end{aligned}$$

- Gauss-Markov theorem: $\hat{\beta}$ has minimum variance among all linear unbiased estimators
- Estimate of the variance σ^2

$$\hat{\sigma}^2 = \frac{1}{n-p-1}(y-\hat{y})^T(y-\hat{y})$$

- Property of $\hat{\sigma}$

$$(N-p-1)\hat{\sigma}^2 \sim \sigma^2 \chi^2_{N-p-1}$$

Least Squares & Linear Regression

- T-test $\beta_j = 0$

$$z_j = \frac{\hat{\beta}_j}{\hat{\sigma}\sqrt{(X_T X)^{-1}_{j\times j}}} \sim t_{N-p-1}$$

- F-test $\beta_j = \beta_{j+1} = \cdots = \beta_{j+k} = 0$

$$F = \frac{(RSS_0 - RSS_1)/(p_1 - p_0))}{RSS_1/(N - p_1 - 1)} \sim F_{p_1-p_0, N-p_1-1}$$

Least Squares & Linear Regression

- Multiple outputs

$$\begin{aligned} Y_k &= \beta_{0k} + \sum_{j=1}^{p} X_j \beta_{jk} + \varepsilon_k \\ Y &= XB + E \end{aligned}$$

$Y = [Y_1, Y_2, \cdots, Y_K]$, $B = [\beta_1, \beta_2, \cdots, \beta_K]$, where $\beta_k = [\beta_{0k}, \cdots, \beta_{p\kappa}]^T$

- Loss function (when error is uncorrelated):

$$\begin{aligned} RSS(B) &= \sum_{k=1}^{K} \sum_{i=1}^{N} (Y - \beta_{0k} - \sum_{j=1}^{p} X_j \beta_{jk})^2 \\ &= tr[(Y - XB)^T (Y - XB)] \end{aligned}$$

Issues for Least Square

- Colinearity: correlated predictors
  - $p > n \Rightarrow$ overfitting!
  - $X^T X$ (nearly) singular, not invertible!
- Low bias but high variance
- Interpretation
- Feature selection/Shrinkage method
  - Subset selection
  - Shrinkage/regularization method

Subset Selection

- Best-subset selection: leaps and bounds *leaps()*
- Forward-stepwise selection: QR Decomposition (ESLII Exercise 3.9)
- Backward-stepwise selection: Z-score (ESLII Exercise 3.10)
- Forward-stagewise regression
  1. Start with $\mathbf{r} = \mathbf{y} - \overline{\mathbf{y}}, \beta_1, \beta_2, \ldots \beta_p = 0$.
  2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.
  3. Update $\beta_j \leftarrow \beta_j + \delta_j$, where $\delta_j = \frac{<\boldsymbol{r}, \boldsymbol{x_j}>}{<\boldsymbol{x_j}, \boldsymbol{x_j}>}$
  4. Update $\mathbf{r} \leftarrow \mathbf{r} - \delta_j \mathbf{x}_j$, and repeat steps 2 and 3 until no predictor has any correlation with $\mathbf{r}$.

Shrinkage or regularization methods

- Regularized or penalized RSS:

$$\text{PRSS}(\beta) = \text{RSS}(\beta) + \lambda J(\beta).$$

  - λ: penalization parameter (tuning/hyperparameter) to be determined
  - $J()$: constraint prior; log prior density (a Bayesian interpretation)

Ridge Regression

- $J(\beta) = \sum_{j=1}^{p} \beta_j^2$ (Tikhonov Regularization)

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}.$$

  $\lambda \geq 0$ controls the amount of shrinkage

- Equivalent problem:

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2,$$

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

- Closed-form solution

$$\hat{\beta}^{ridge} = \left(X^T X + \lambda I\right)^{-1} X^T Y.$$

Ridge Regression

- Nonsingular even if $X^T X$ is not full rank
- Biased but small variances,
  $E\left(\hat{\beta}^R\right) = \left(X^T X + \lambda I\right)^{-1} X^T X \beta$,
  $\text{Var}\left(\hat{\beta}^R\right) = \sigma^2 \left(X^T X + \lambda I\right)^{-1} X^T X \left(X^T X + \lambda I\right)^{-1} \leq \text{Var}(\hat{\beta})$,
- Prior of β: $\beta_j \sim N\left(0, \tau^2\right)$. (ESLII Exercise 3.6)

Ridge Regression

- Special case: orthonormal inputs $\hat{\beta}^{ridge} = \frac{\hat{\beta}}{1+\lambda}$
- (Thin) SVD decomposition (when $N > p$)

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

  $\boldsymbol{U}$: $N \times p$ matrix; $\boldsymbol{V}$: $p \times p$ matrix; $\boldsymbol{D}$: $p \times p$ diagonal matrix;
- $\boldsymbol{U}$: column vectors $u_1, u_2, \cdots, u_p$ form a orthonormal set, i.e., $U^T U = I$
- $\boldsymbol{V}$: column vectors $v_1, v_2, \cdots, v_p$ form a orthonormal set, i.e., $V^T V = I$

Ridge Regression

- (Thin) SVD decomposition (when $N > p$)
- $$\begin{aligned}\mathbf{X}\hat{\beta}^{1s} &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &= \mathbf{U}\mathbf{U}^T\mathbf{y}\end{aligned}$$
- $$\begin{aligned}\mathbf{X}\hat{\beta}^{\text{ridge}} &= \mathbf{X}\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \\ &= \mathbf{U}\mathbf{D}\left(\mathbf{D}^2 + \lambda\mathbf{I}\right)^{-1}\mathbf{D}\mathbf{U}^T\mathbf{y} \\ &= \sum_{j=1}^{p}\mathbf{u}_j\frac{d_j^2}{d_j^2 + \lambda}\mathbf{u}_j^T\mathbf{y}\end{aligned}$$

Ridge Regression

- Effective degree of freedom
  $df(\lambda) = \text{tr}\left[X\left(X^TX + \lambda I\right)^{-1}X^T\right] \leq df(0) = \text{tr}\left(X\left(X^TX\right)^{-1}X^T\right) =$
  $\text{tr}\left(\left(X^TX\right)^{-1}X^TX\right) = p$
- X is orthonormal

$$df(\lambda) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}$$

Lasso Regression

- $J(\beta) = \sum_{j=1}^{p} |\beta_j|$.
- Equivalent problem:

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2,$$
$$\text{subject to } \sum_{j=1}^{p} |\beta_j| \leq t,$$

- No closed-form solution

Lasso Regression

- Prior of β: β_j Laplace or DE $\left(0, \frac{1}{\lambda}\right)$;
- Biased but small variances
- $\hat{\beta}_j^{Lasso} = 0$ for large λ (but not $\hat{\beta}_j^{Ridge}$)
- $df\left(\hat{\beta}^{Lasso}\right) = \#$ of non-zero $\hat{\beta}_j^{Lasso}$'s (Zou et al).
- Special case: orthonormal inputs:
  - $\hat{\beta}_j^{Lasso} = \mathrm{ST}\left(\hat{\beta}_j, \lambda\right) = \mathrm{sign}\left(\hat{\beta}_j\right)\left(\left|\hat{\beta}_j\right| - \lambda\right)_+$
  - $\hat{\beta}_j^{Ridge} = \hat{\beta}_j/(1+\lambda)$
  - $\hat{\beta}_j^{Best} = \mathrm{HT}\left(\hat{\beta}_j, M\right) = \hat{\beta}_j I\left(\mathrm{rank}\left(\hat{\beta}_j\right) \leq M\right)$

Lasso Regression

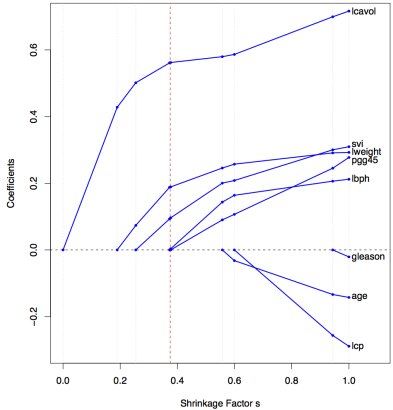


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t/\sum_1^p |\hat{\beta}_j|$. A vertical line is drawn at $s = 0.36$, the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*

Lasso vs. Ridge

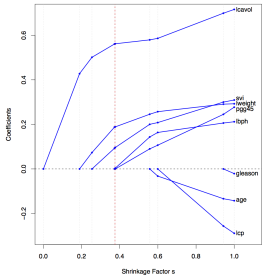


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus $s = t/\sum_1^p |\beta_j|$. A vertical line is drawn at $s = 0.36$, the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*

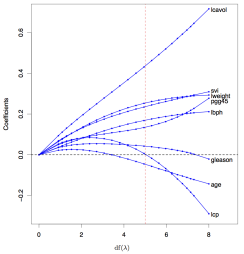
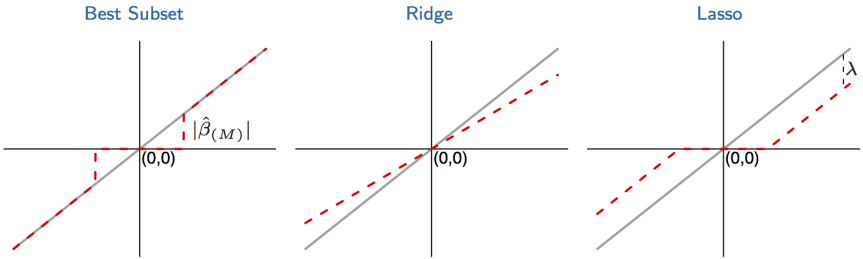


FIGURE 3.8. *Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter λ is varied. Coefficients are plotted versus $df(\lambda)$, the effective degrees of freedom. A vertical line is drawn at $df = 5.0$, the value chosen by cross-validation.*

Lasso vs. Ridge



Lasso vs. Ridge

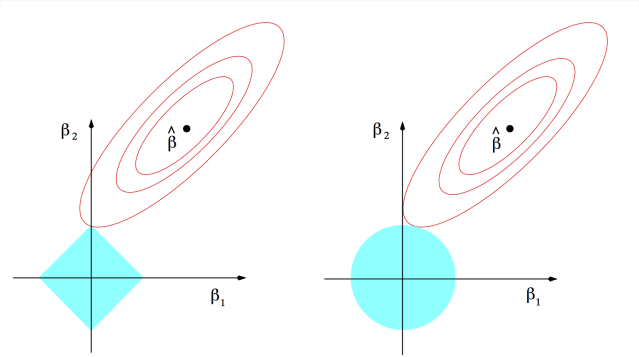


FIGURE 3.11. *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

Regularization methods

- Generalized ridge and lasso

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q \right\}.$$

  $q \geq 0$

- Elastic-net (Zou and Hastie (2005))

$$\lambda \sum_{j=1}^{p} (\alpha\beta_j^2 + (1-\alpha)|\beta_j|)$$

  $0 < \alpha < 1$

Least Angle Regression (LAR)

Algorithm 3.2 *Least Angle Regression.*

1. Standardize the predictors to have mean zero and unit norm. Start with the residual $\mathbf{r} = \mathbf{y} - \bar{\mathbf{y}}$, $\beta_1, \beta_2, \ldots, \beta_p = 0$.

2. Find the predictor $\mathbf{x}_j$ most correlated with $\mathbf{r}$.

3. Move β_j from 0 towards its least-squares coefficient $\langle \mathbf{x}_j, \mathbf{r} \rangle$, until some other competitor $\mathbf{x}_k$ has as much correlation with the current residual as does $\mathbf{x}_j$.

4. Move β_j and β_k in the direction defined by their joint least squares coefficient of the current residual on $(\mathbf{x}_j, \mathbf{x}_k)$, until some other competitor $\mathbf{x}_l$ has as much correlation with the current residual.

5. Continue in this way until all p predictors have been entered. After $\min(N - 1, p)$ steps, we arrive at the full least-squares solution.

LAR vs. Lasso

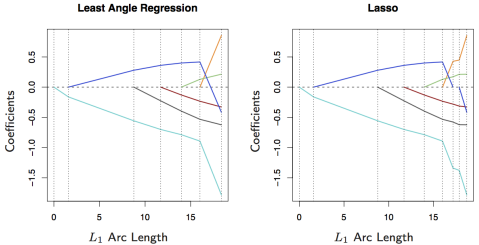


FIGURE 3.15. *Left panel shows the LAR coefficient profiles on the simulated data, as a function of the L_1 arc length. The right panel shows the Lasso profile. They are identical until the dark-blue coefficient crosses zero at an arc length of about* 18.

LAR vs. Lasso

Algorithm 3.2a *Least Angle Regression: Lasso Modification.*

4a. If a non-zero coefficient hits zero, drop its variable from the active set of variables and recompute the current joint least squares direction.

Lasso Algorithms

- Quadratic programming
- LARS (2001)
- Coordinate descent/shooting (2007) in ESL
- Proximal Gradient (2009) (Iterate shrinkage and thresholding algorithm)

Lasso Regression

- Shrink non-zero coefficient $\Rightarrow$ Not consistent
- Smoothly clipped absolute deviation (SCAD) penalty (Fan and Li, 2005)

$$J(\beta) = \lambda\{I(\beta \leq \lambda) + \frac{(\alpha\lambda - \beta)_+}{(\alpha - 1)\lambda} I(\beta > \lambda)\}$$

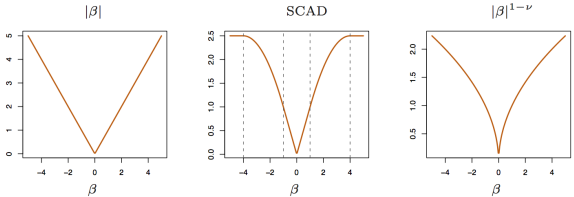


FIGURE 3.20. *The lasso and two alternative non-convex penalties designed to penalize large coefficients less. For SCAD we use $\lambda = 1$ and $a = 4$, and $\nu = \frac{1}{2}$ in the last panel.*

Lasso Regression

- Adaptive Lasso:

$$J(\beta) = \lambda \sum_i \hat{\omega}_j |\beta_j|$$

  β_j increases, ω_j decreases such that less shrinkage. One choice of weights can be: $\hat{\omega}_j = 1/|\hat{\beta}_j^{LS}|^\nu$

- MC+: Smooth link between hard-threshold and least square; a minimax concave penalty (MCP) (Zhang, 2010)

R-implementation for Lasso Regression

- Many R-routines to compute a Lasso solution of path, a function of tuning parameter λ.
- Lasso and elastic-net regularized generalized linear models*glmnet*: `http://cran.r-project.org/web/packages/glmnet/index.html`
- Improved *glmnet*: `https://www.csie.ntu.edu.tw/~cjlin/papers/l1_glmnet/long-glmnet.pdf`. Highly recommended faster than (5-10 times).

R-implementation for Lasso Regression

```
# Fit and predict
install.packages("glmnet")
library(glmnet)
x=matrix(rnorm(100*20),100,20)
y=rnorm(100)
fit.lasso=glmnet(x,y)
coef(fit.lasso,s=0.01)
coef(fit.lasso,s=0.1)
predict(fit.lasso,newx=x[1:10,],s=c(0.01,0.1))
```

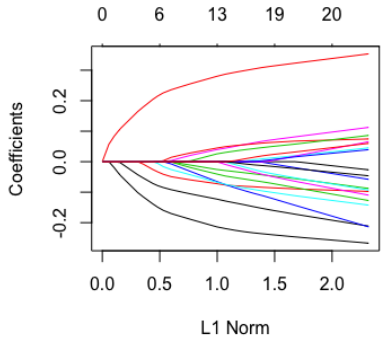
Python-implementation for Lasso Regression

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_regression
from sklearn.linear_model import Lasso
import numpy as np

n_samples, n_features = 1000, 20
rng = np.random.RandomState(0)
X, y = make_regression(n_samples, n_features, random_state=rng)
sample_weight = rng.rand(n_samples)
X_train, X_test, y_train, y_test, sw_train, sw_test = train_test_split(
    X, y, sample_weight, random_state=rng)
reg = Lasso()
reg.fit(X_train, y_train, sample_weight=sw_train)
print(reg.score(X_test, y_test, sw_test))
```

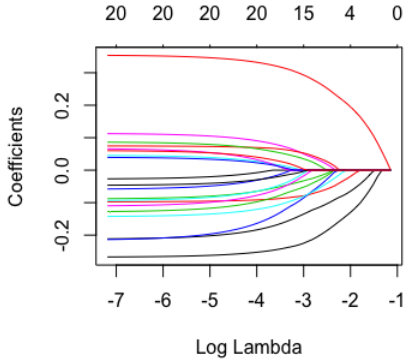
R-implementation for Lasso Regression

```
# Regularization path
plot(fit.lasso, xvar="norm" )
```



R-implementation for Lasso Regression

```
# Regularization path
plot(fit.lasso, xvar="lambda")
# For python, please refer to sklearn.linear_model.lasso_path
```



Implementation

```
# LARS - R
library("lars")
lars.fit <- lars(x,y, type="lasso")
# LARS - Python
from sklearn.linear_model import lars
reg = Lars(n_nonzero_coefs=1)
reg.fit(X_train, y_train)
```

Lasso Regression

- Predictors belong to pre-defined L groups
- Grouped Lasso:

$$\hat{\beta}^{\text{glasso}} = \underset{\beta}{\text{argmin}} ||y_i - \beta_0 \mathbf{1} - \sum_{\ell=1}^{L} \boldsymbol{X}_\ell \beta_\ell||_2^2 + \lambda \sum_{\ell=1}^{L} \sqrt{p_\ell} ||\beta_\ell||_2$$

$\sqrt{p_\ell}$: group size
$\boldsymbol{X}_\ell$: Predictors for the ℓ-th group
β_ℓ: coefficient vector for the ℓ-th group

R-implementation

```
library(grpgreg)
library("lars")
group <- c(rep(1,4), rep(2,6))
fit <- grpreg(x,y,group,penalty="grLasso")
plot(fit,main = "Group Lasso")
fit <- grpreg(x,y,group,penalty="grMCP")
plot(fit, main = "Group MCP")
fit <- grpreg(x,y,group,penalty="grSCAD")
plot(fit, main = "Group SCAD")
```

Using Dervied Input Directions

- Principle Components Regression (variation in X is useful for predicting Y)
  1. Standardize X
  2. Take SVD of X and find truncated $Z = UD$ (truncated $k < r$)
  3. regression on Z

$$\hat{\beta}^{PCR} = (Z^T Z)^{-1} Z^T y = (DU^T UD)^{-1} DU^T y = D^{-1} U^T y$$

- Partial Least Square Regression
  - PCA maximizes $Cov(X)$ while PLS maximizes $Cov(X, y)$
  - Find direction with variation in X related to Y

Classification Methods

- $Y_{n\times 1}$: class labels (binary classification: $Y_i = 0$ or 1)
- Decision boundary: a surface that partitions the underlying vector space into two sets, one for each class
- Linear: monotone transformation of discriminant function $\delta_k(x)$ or $Pr(Y = k|X = x)$ is linear
- Linear regression: $E(Y_i \mid X_i) = \beta_0 + X_i^T\beta$
  - Use LS to estimate β^T 's $\Longrightarrow \hat{Y}_i$; $\tilde{Y}_i = I\left(\hat{Y}_i \geq 0.5\right)$
  - Decision boundary: $\hat{Y}(x) = \hat{\beta}_0 + x^T\hat{\beta} = 0.5$, linear

Classification Methods

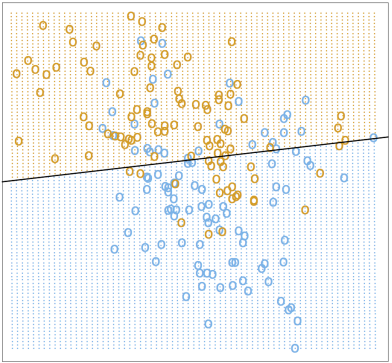


FIGURE 2.1. *A classification example in two dimensions. The classes are coded as a binary variable (*BLUE $= 0$, ORANGE $= 1$*), and then fit by linear regression. The line is the decision boundary defined by* $x^T\hat{\beta} = 0.5$*. The orange shaded region denotes that part of input space classified as* ORANGE*, while the blue region is classified as* BLUE*.*

Nearest Neighbor Methods

- KNN: $N_{k(x)}$ is the k nearest training data points that are closest to x (Euclidean distance)

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} Y_i.$$

- Majority vote: $\hat{Y}(x) > 0.5$
- Key: choice of k, or how much "smoothness" is to be assumed
- Modeling assumption: larger k, higher or lower model complexity?
- Try a few values of k, then ...

Nearest Neighbor Methods

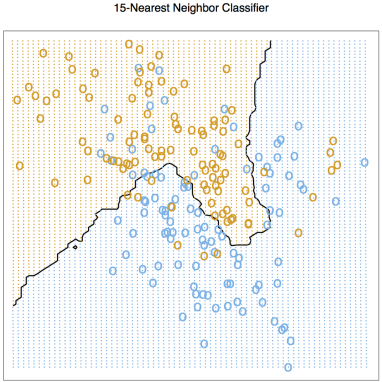


FIGURE 2.2. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$) *and then fit by* 15*-nearest-neighbor averaging as in (2.8). The predicted class is hence chosen by majority vote amongst the* 15*-nearest neighbors.*

Nearest Neighbor Methods

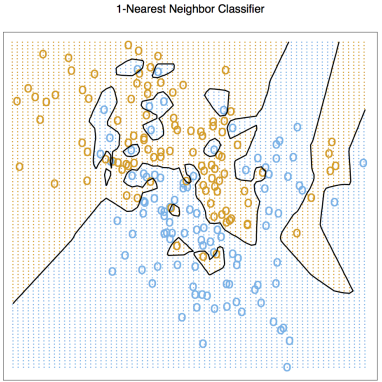


FIGURE 2.3. *The same classification example in two dimensions as in Figure 2.1. The classes are coded as a binary variable* (BLUE $= 0$, ORANGE $= 1$)*, and then predicted by* 1*-nearest-neighbor classification.*

Nearest Centroid (Prototype) Classifier

- Assign the label of the class of training samples whose mean (centroid) is closest
- Training: compute the per-class centroids $\vec{\mu}_\ell = \frac{1}{|C_\ell|} \sum_{i \in C_\ell} \vec{x}_i$ where C_ℓ is the index set of samples belonging to class $\ell \in \mathbf{Y}$.
- Prediction: $\hat{y} = \arg\min_{\ell \in \mathbf{Y}} \|\vec{\mu}_\ell - \vec{x}\|$

Naive Bayes Classifier

- Bayes theorem (choose the class with the highest posterior density)

$$P(Y = k \mid X) = \frac{\pi_k P(X \mid Y = k)}{\sum_l \pi_l P(X \mid Y = l)}$$

  where $\pi_k = P(Y \in C(k))$ is the class prior. (usually, $\hat{\pi}_k = \frac{n_k}{n}$)

- Gaussian Bayes Classifier:

$$x \mid Y = k \sim N\left(\mu_k, \sigma_k^2 \mathbf{I}_{p \times p}\right)$$

- Parameter estimation:

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i \in C(k)} X_i$$

Linear Regression of an Indicator Matrix

- Categorical Y with K classes; $K \geq 2$
- One-hot encoding: Indicator variable $Y_k = 1$ if $Y = k$ for $k = 1, \cdots, K$ (similar to dummy encoding)
- LM: for each class k, $f_k(x) = E(y_k \mid x) = \Pr(Y = k \mid x) = \beta_{0k} + \beta_k^T x$
- $\hat{Y}(x) = \arg\max_k \hat{f}_k(x)$
- Multivariate response $(Y_1, \cdots, Y_K)$
- Decision boundary: $f_k(x) = f_l(x)$, linear
- Classes can be masked by others

Least Squares & Linear Regression

- Multiple outputs

$$\begin{aligned} Y_k &= \beta_{0k} + \sum_{j=1}^{p} X_j \beta_{jk} + \varepsilon_k \\ Y &= XB + E \end{aligned}$$

$Y = [Y_1, Y_2, \cdots, Y_K]$, $B = [\beta_1, \beta_2, \cdots, \beta_K]$, where $\beta_k = [\beta_{0k}, \cdots, \beta_{p\kappa}]^T$

- Loss function (when error is uncorrelated):

$$\begin{aligned} RSS(B) &= \sum_{k=1}^{K} \sum_{i=1}^{N} (Y - \beta_{0k} - \sum_{j=1}^{p} X_j \beta_{jk})^2 \\ &= tr[(Y - XB)^T (Y - XB)] \end{aligned}$$

Linear Regression of an Indicator Matrix

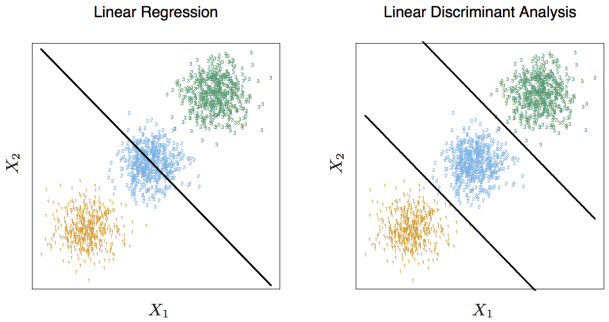


FIGURE 4.2. *The data come from three classes in* $\mathbb{R}^2$ *and are easily separated by linear decision boundaries. The right plot shows the boundaries found by linear discriminant analysis. The left plot shows the boundaries found by linear regression of the indicator response variables. The middle class is completely masked (never dominates).*

Linear Regression of an Indicator Matrix

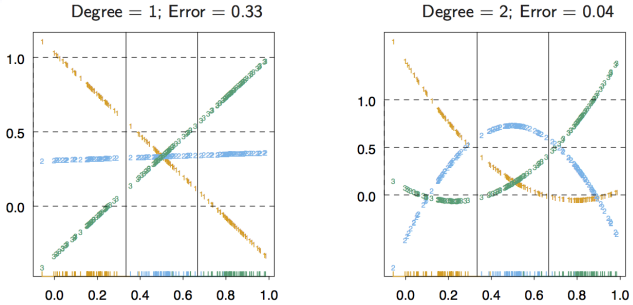


FIGURE 4.3. *The effects of masking on linear regression in* $\mathbb{R}$ *for a three-class problem. The* rug plot *at the base indicates the positions and class membership of each observation. The three curves in each panel are the fitted regressions to the three-class indicator variables; for example, for the blue class,* y_{blue} *is* 1 *for the blue observations, and* 0 *for the green and orange. The fits are linear and quadratic polynomials. Above each plot is the training error rate. The Bayes error rate is* 0.025 *for this problem, as is the LDA error rate.*

Discriminant Analysis

- Optimal Bayes rule: $\arg\max_k \Pr(Y = k \mid X)$.

$$P(Y = k \mid X) = \frac{\pi_k P(X \mid Y = k)}{\sum_l \pi_l P(X \mid Y = l)} = \frac{\pi_k f_k(x)}{\sum_{l=1}^K \pi_l f_l(x)}$$

- Assume $f_k = MVN(\mu_k, \Sigma) \Longrightarrow$ LDA.
- Assume $f_k = MVN(\mu_k, \Sigma_k) \Longrightarrow$ QDA.
- FDA, assume homoscedasticity $\Longrightarrow$ LDA
- Estimate f_k nonparametrically, e.g. by kernel density estimation $\Longrightarrow$ KDA.
  General, but not working well for large p - "curse of dim."
- Naive Bayes: assuming independence among the predictors, $f_k(x) = \prod_{j=1}^p f_{kj}(x_j)$.

Linear Discriminant Analysis

- Assume: $x \mid k \sim N(\mu_k, \Sigma)$.

$$\delta_k(x) \propto \log(\pi_k f_k(x)) \propto \log \pi_k + x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k$$

- $\hat{y} = \arg\max_k \Pr(k \mid x) = \arg\max_k \delta_k(x)$.
- In practice, estimate

$$\begin{aligned}
\hat{\pi}_k &= n_k / n, \\
\hat{\mu}_k &= \sum_{G_i = k} x_i / n_k, \\
\hat{\Sigma} &= \sum_{k=1}^{K} \sum_{G_i = k} (x_i - \hat{\mu}_k)^T (x_i - \hat{\mu}_k) / (N - K),
\end{aligned}$$

Linear Discriminant Analysis

- Log-ratio between two classes k and l

$$\begin{aligned}\log \frac{\Pr(k \mid x)}{\Pr(l \mid x)} &= \delta_k(x) - \delta_l(x) \\ &= \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + x^T \Sigma^{-1}(\mu_k - \mu_l) \\ &= \beta_{0,kl} + x^T \beta_{kl},\end{aligned}$$

- Linear decision boundary
- when $K = 2$,
  - LDA rule:

$$x^T \hat{\boldsymbol{\Sigma}}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) > \frac{1}{2}\hat{\mu}_2^T \hat{\boldsymbol{\Sigma}}^{-1}\hat{\mu}_2 - \frac{1}{2}\hat{\mu}_1^T \hat{\boldsymbol{\Sigma}}^{-1}\hat{\mu}_1 + \log(N_1/N) - \log(N_2/N)$$

  - Code $y = -N/N1, N/N2$, LM ($E(y) = b_0 + x^T b$) rule: $\hat{b} \propto \hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$ (Ex. 4.2)

Quadratic Discriminant Analysis

- Assume: $x \mid k \sim N(\mu_k, \Sigma_k)$.

$$\delta_k(x) \propto \log \pi_k - \frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k).$$

quadratic, or

$$\log \frac{\Pr(k \mid x)}{\Pr(l \mid x)} = \beta_{0,kl} + x^T \beta_{1,kl} + x^T B_{2,kl} x.$$

quadratic logit model.

- LDA vs. QDA: much larger number of parameters; $(K-1) \times (p+1)$ vs. $(K-1) \times \{p(p+3)/2 + 1\}$

Quadratic Discriminant Analysis

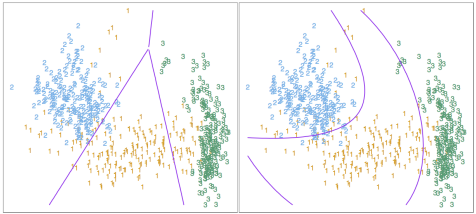


FIGURE 4.1. *The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$. Linear inequalities in this space are quadratic inequalities in the original space.*

Quadratic Discriminant Analysis

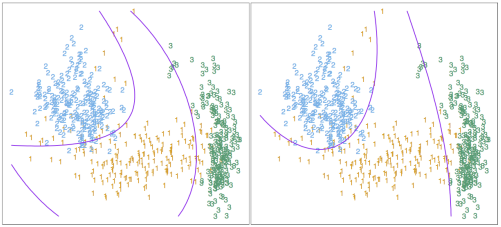


FIGURE 4.6. *Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is usually the case.*

Regularized Discriminant Analysis

- A compromise between LDA and QDA: use $\tilde{\Sigma}_k(\alpha)$ in QDA,

$$\tilde{\Sigma}_k(\alpha) = \alpha\hat{\Sigma}_k + (1-\alpha)\hat{\Sigma}$$

  where $\alpha \in [0, 1]$
- Shrinkage: Replace $\hat{\Sigma}$ with $\hat{\Sigma}(\gamma) = \gamma\hat{\Sigma} + (1-\gamma)\hat{\sigma}^2 I$,

Logistic Regression

- Assume $y_i \sim \text{Bernolli}(p)$

$$f(y_i) = p_i^{y_i} (1 - p_i)^{1-y_i} \text{ where } y_i \in \{0, 1\}$$
$$E[y_i] = p_i \in [0, 1] \overset{\text{set}}{\neq} X_i^T \beta \in (-\infty, \infty)$$

- A link function: logit function $g : [0, 1] \to \mathbb{R}$:

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1 - p_i}\right) \overset{\text{set}}{=} X_i^T \beta \text{ where } p_i \in (0, 1)$$

- Logistic function

$$p_i = g^{-1}(X_i^T) = \frac{e^{X_i^T \beta}}{1 + e^{X_i^T \beta}}$$

Logistic Regression

- Likelihood function:

$$\begin{aligned} L(\beta) &= \prod_{i=1}^{n} p_i^{y_i} \left(1 - p_i\right)^{1-y_i} \\ &= \prod_{i=1}^{n} \left[\frac{e^{X_i^T \beta}}{1 + e^{X_i^T \beta}}\right]^{y_i} \left[1 - \frac{e^{X_i^T \beta}}{1 + e^{X_i^T \beta}}\right]^{(1-y_i)} \end{aligned}$$

- Log-likelihood

$$\ell(\beta) = \sum_i y_i X_i^T \beta - \log\left(1 + \exp\left(X_i \beta\right)\right)$$

- Penalized logistic regression:

$$-\ell(\beta) + \lambda P(\beta)$$

Logistic Regression

- The derivative of log-likelihood (score function)

$$\frac{\partial l}{\partial \beta} = \sum_i x_i \left(y_i - \frac{e^{x_i^T \beta}}{1 + e^{x_i^T \beta}} \right) \stackrel{\text{set}}{=} 0$$

- Newton method

$$\beta^{\text{new}} = \beta^{\text{old}} - \left[\frac{\partial^2 l}{\partial \beta^2} \left(\beta^{\text{old}} \right) \right]^{-1} \frac{\partial l}{\partial \beta} \left(\beta^{\text{old}} \right)$$

- Estimation of penalized logistic regression: proximal gradient descent

Logistic Regression

- Wald test $H_0 : \beta_j = 0$

$$z = \frac{\hat{\beta}_j}{SE\left(\hat{\beta}_j\right)} \xrightarrow{D} N(0,1)$$

- Likelihood ratio test $H_0 : \beta_q = \beta_{q+1} = \cdots = \beta_p = 0$. Test the goodness of fit between the reduced model and the full model.

$$\lambda = \frac{L(R)}{L(F)}$$

$$-2\log\lambda \xrightarrow{D} \chi^2_{df_{full} - df_{reduced}}$$

Multi-class Logistic Regression

- Multiple classes: $y_i \in \{1, \cdots, K\}$, choose a basis class K (can be an arbitrary class)

$$\log \frac{P(Y = k \mid X = x)}{P(Y = K \mid X = x)} = \beta_k^T X, k = 1, \cdots, K-1$$

- As a set of independent binary regressions

$$P(Y = k \mid X = x) = \frac{e^{\beta_K^T X}}{1 + \sum_{i=1}^{k-1} e^{\beta_i^T X}}, k = 1, \cdots, K-1$$

$$P(Y = K \mid X = x) = \frac{1}{1 + \sum_{i=1}^{k-1} e^{\beta_i^T X}}$$

- Log linear model / softmax function

$$\Pr(Y_i = k) = \frac{e^{\boldsymbol{\beta}_k \cdot \mathbf{x}_i}}{\sum_{j=1}^{K} e^{\boldsymbol{\beta}_j \cdot \mathbf{x}_i}}$$

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.

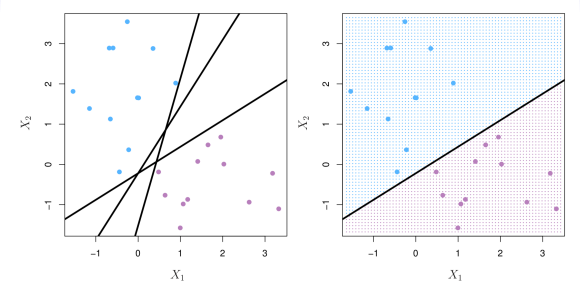
What is a Hyperplane?

- A hyperplane in p dimensions is a flat subspace of dimension $p-1$.
- In general the equation for a hyperplane has the form

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p = 0.$$

- In $p = 2$ dimensions a hyperplane is a line.
- If $\beta_0 = 0$, the hyperplane goes through the origin, otherwise not.
- The vector $\beta = (\beta_1, \beta_2, \cdots, \beta_p)$ is called the normal vector — it points in a direction orthogonal to the surface of a hyperplane.

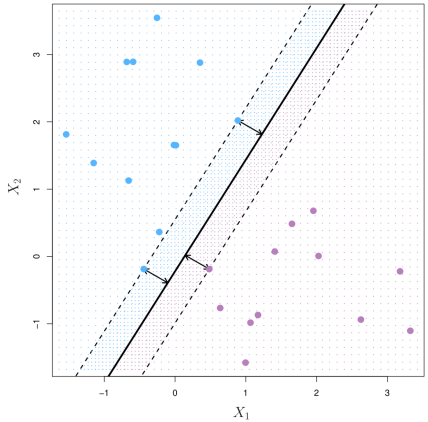
Separating Hyperplanes



- 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 purple, then if $Y_i \cdot f(X_i) > 0$ for all i, $f(X) = 0$ defines a **separating hyperplane**.

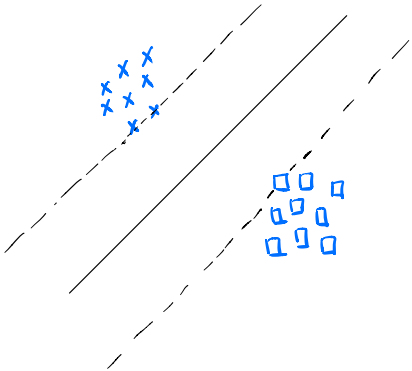
Maximal Margin Classifier

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



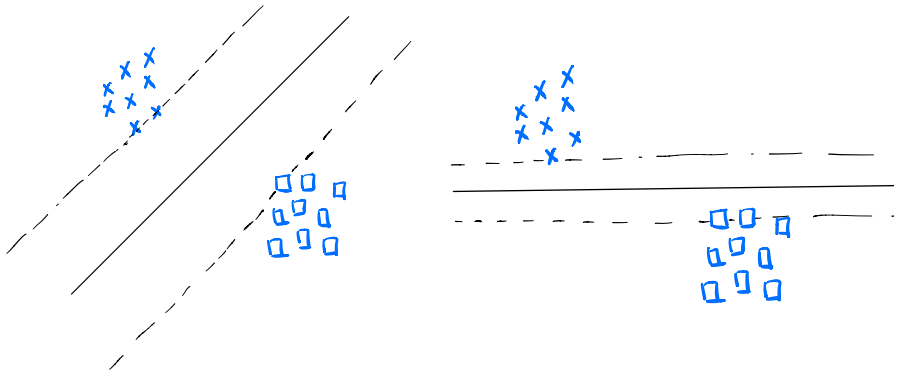
Why biggest margin?

Training



Why biggest margin?

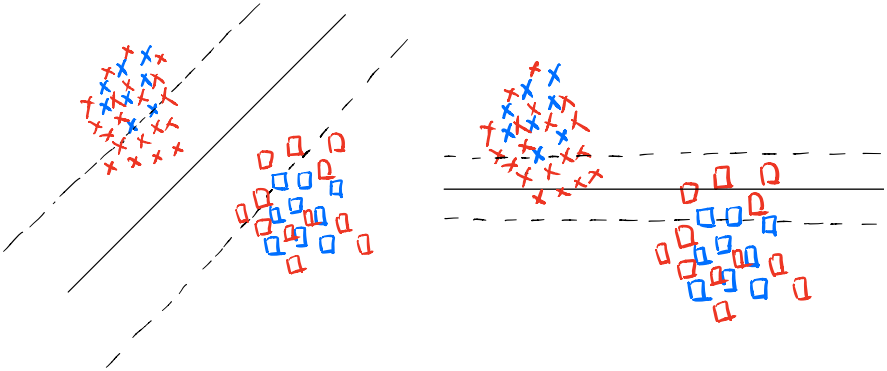
Training



makes no obvious difference.

Why biggest margin?

Testing



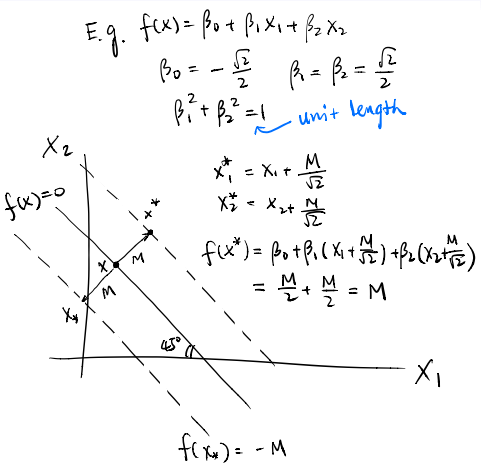
Wide margin is clearly better.

Constrained optimization problem

$$\begin{aligned} &\text{maximize}_{\beta_0,\beta_1,\cdots,\beta_p}\ M \\ &\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1, \\ &y_i(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \geq M \qquad \text{for all } i = 1, \cdots, N. \end{aligned}$$

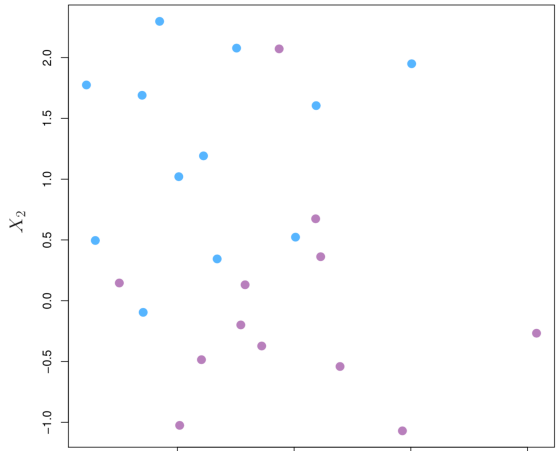
This can be rephrased as a convex quadratic program, and solved efficiently. The function *svm()* in package *e1071* solves this problem efficiently.

Constraint explained

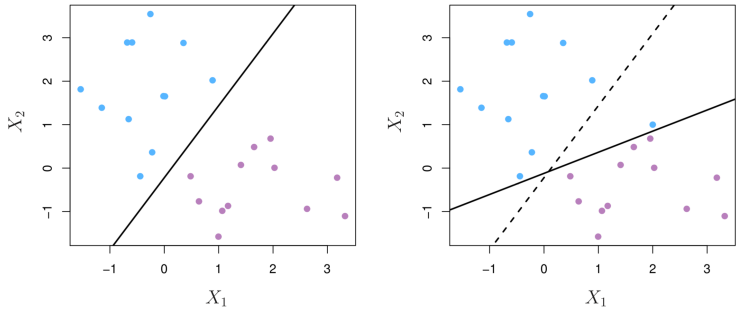


Non-separable Data

The data on the left are not separable by a linear boundary.
This is often the case when $n > p$.



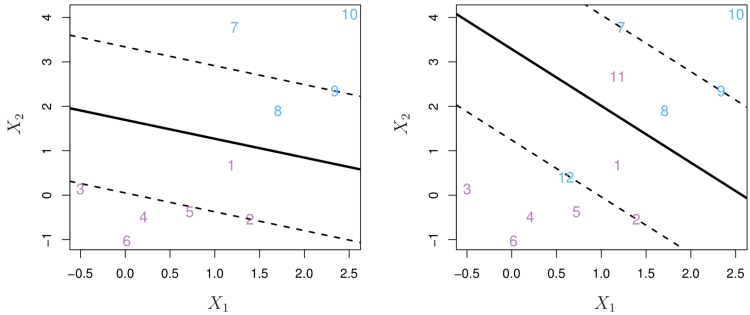
Noisy Data



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



For each point on the wrong side of the dashed line, we pay a "price". And we have a total budget of C to spend. The price is proportional to the distance from the point to the dashed line.

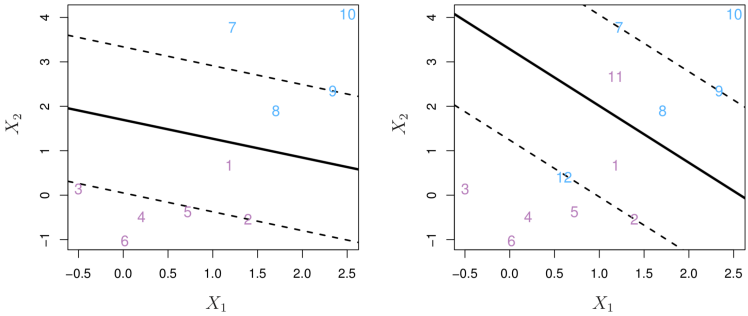
For each point on the wrong side of the dashed line, we pay a "price". And we have a total budget of C to spend. The price is proportional to the distance from the point to the dashed line.

$$\text{maximize}_{\beta_0,\beta_1,\cdots,\beta_p,\epsilon_1,\cdots,\epsilon_n} \quad M \quad \text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \epsilon_i),$$

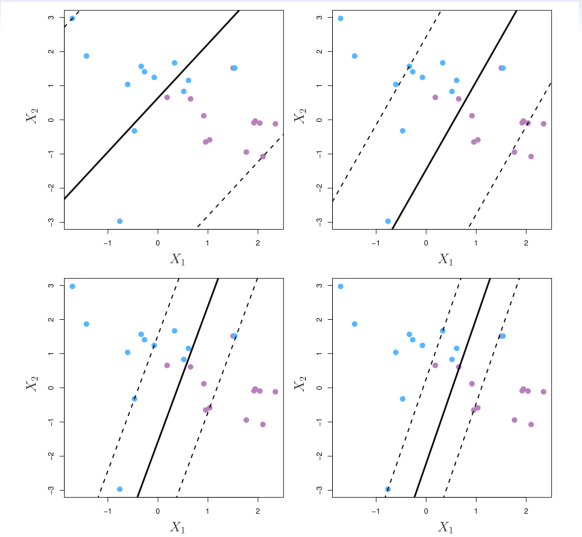
$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C.$$

Support Vector Classifier

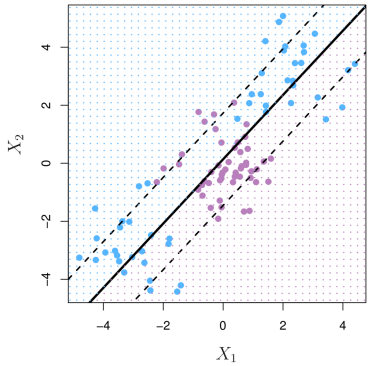


Observations that lie directly on the margin, or on the wrong side of the margin for their class, are known as support vectors.
Left: 1,2,8,9 are support vectors. Right: 1,2,7,8,9,11,12 are support vectors

C is a regularization parameter

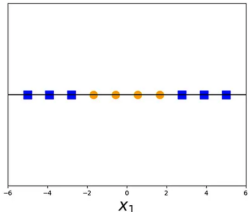


Linear boundary can fail

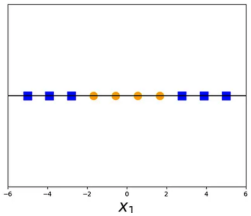


Sometimes a linear boundary simply won't work, no matter what value of C.
What to do?

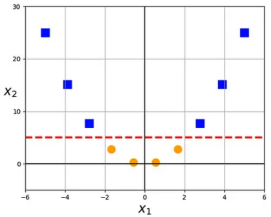
One-dimensional Example



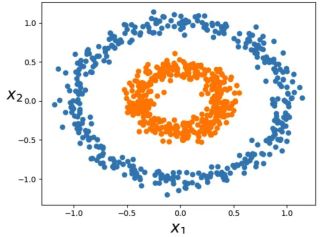
One-dimensional Example



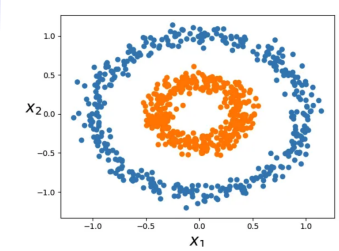
$x \to (x, x^2)$



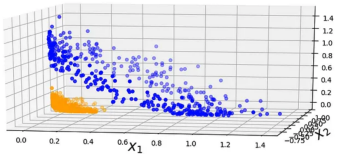
Two-dimensional Example



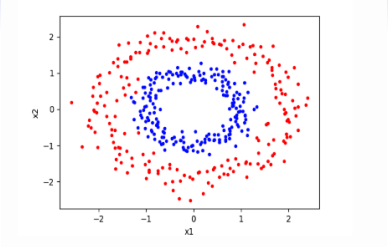
Two-dimensional Example



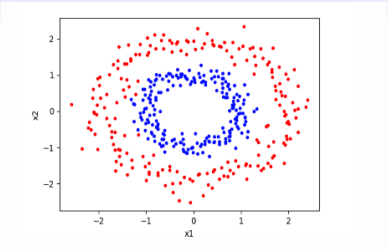
$$x_1, x_2 \to (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



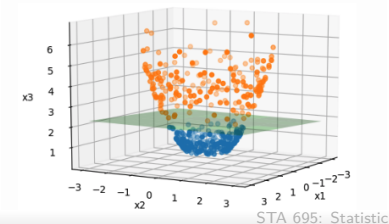
Two-dimensional Example



Two-dimensional Example



$x_1, x_2 \rightarrow (x_1, x_2, x_1^2 + x_2^2)$



Feature Expansion

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

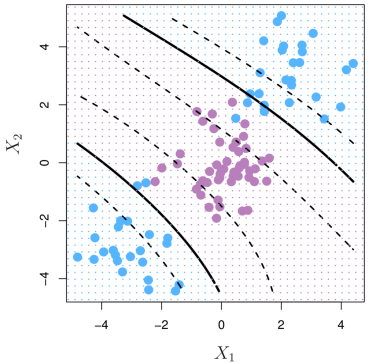
Example: Suppose we use $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$ instead of just (X_1, X_2). Then the decision boundary would be of the form

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

This leads to nonlinear decision boundaries in the original space.

Cubic Polynomials

Here we use a basis expansion of cubic polynomials
From 2 variables to 9
The support-vector classifier in the enlarged space solves the problem in the lower-dimensional space.



Nonlinearities and Kernels

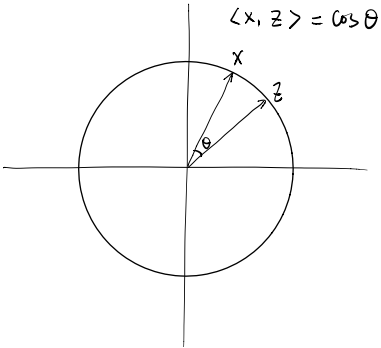
- Polynomials are hard to compute even for moderate degree when p is large.
- There is a more elegant and controlled way to introduce nonlinearities in support-vector classifiers — through the use of **kernels**.
  - Computationally, only need to compute $\binom{n}{2}$ inner products of p-dimensional vectors. Details later.
  - Choosing kernel is not an easy task.
- Before we discuss these, we must understand the role of **inner products** in support-vector classifiers.

Inner products

Inner (dot) product between vectors

$$< x, z >= \sum_{j=1}^{p} x_j z_j$$

Inner product measures the similarity of two vectors.



Inner products and support vectors

- The solution to the linear support vector classifier can be written as

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\alpha}_i < x, x_i >$$

- To estimate the parameters $\alpha_1, \cdots, \alpha_n$ and β_0, all we need are the $\binom{n}{2}$ inner products $< x_i, x_{i'} >$ for all pairs of training observations.
- It turns out that that α_i is nonzero only for the support vectors.

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

$\mathcal{S}$ is the collection of indices of these support vectors.

Kernels and Support Vector Machines

- We can replace the inner product by other similarity measures.
- Kernel functions can do this for us. E.g. polynomial kernel

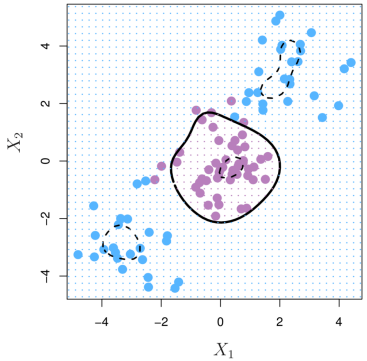
$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^{p} x_{ij} x_{i'j}\right)^d$$

- The solution has the form

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

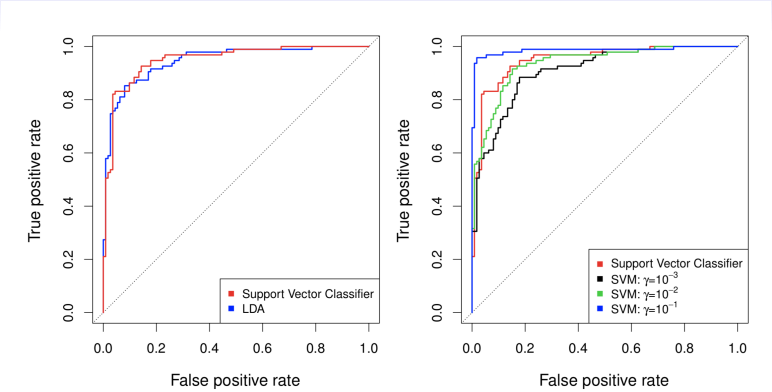
Radial Kernel

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



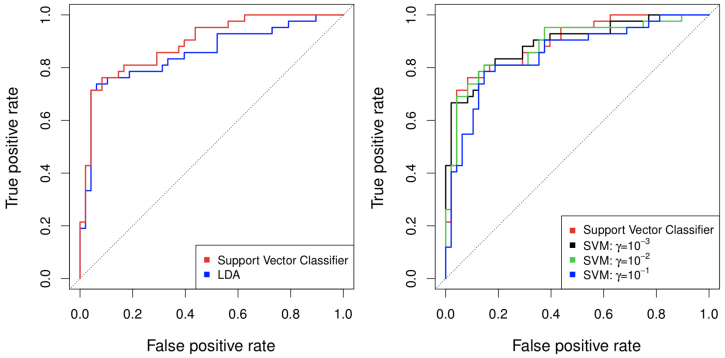
Larger γ leads to more flexible decision boundary.

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?

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 $\hat{f}_k(x), k = 1, \cdots, K$; 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 $\hat{f}_k(x)$. Classify x^* to the class that wins the most pairwise competitions.

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

Which to use: SVM or Logistic Regression or LDA

- When classes are (nearly) separable, SVM and LDA are better than LR.
- When not, LR, LDA and SVM very similar.
- If you wish to estimate probabilities, use LR or LDA.
- For nonlinear boundaries, kernel SVMs are popular. Can use kernels with LR and LDA as well, but computations are more expensive.
- When sample size is large, kernel SVMs can be very slow

Tree-based Methods

- Here we describe tree-based methods for regression and classification.
- These involve **stratifying** or **segmenting** the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as **tree-based** or **decision-tree** methods.

Pros and Cons

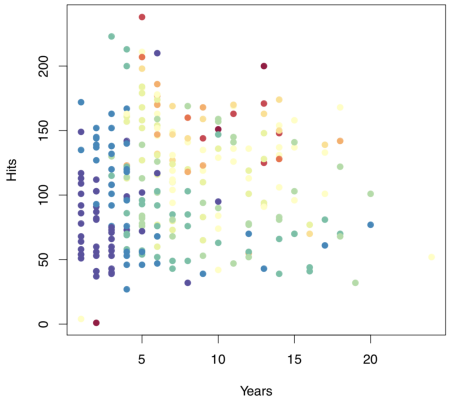
- Tree-based methods are simple and useful for interpretation.
- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.
- Hence we will also discuss **bagging, random forests**, and **boosting** in the topic of *Ensemble Learning*. These methods grow multiple trees which are then combined to yield a single consensus prediction.
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss of interpretation.

The Basics of Decision Trees

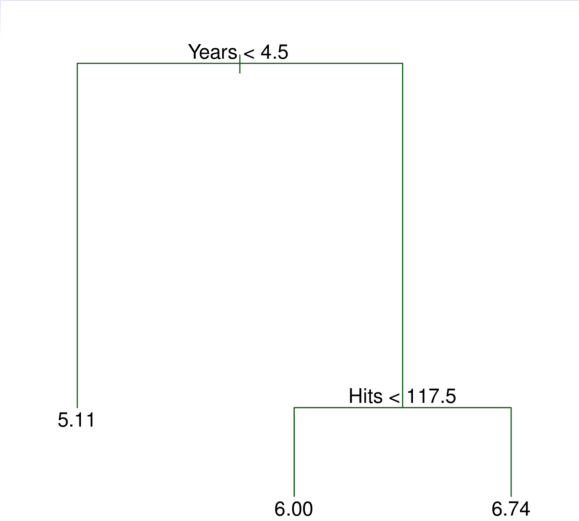
- Decision trees can be applied to both regression and classification problems.
- We first consider regression problems, and then move on to classification.

Baseball salary data: how would you stratify it?

Salary is color-coded from low (blue, green) to high (yellow,red)



Decision tree for these data

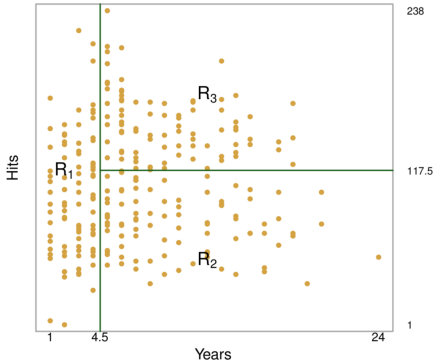


Details of the tree

- For the Hitters data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year.
- At a given internal node, the label (of the form $X_j < t_k$) indicates the condition satisfied on the left-hand branch of the split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to $Years < 4.5$, and the right-hand branch corresponds to $Years >= 4.5$.
- The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.

Results

- Overall, the tree stratifies or segments the players into three regions of predictor space: $R_1 = \{X | Years < 4.5\}$, $R_2 = \{X | Years >= 4.5, Hits < 117.5\}$, and $R_3 = \{X | Years >= 4.5, Hits >= 117.5\}$.



Terminology for Trees

- In keeping with the **tree** analogy, the regions R_1, R_2, and R_3 are known as **terminal nodes**.
- Decision trees are typically drawn **upside down**, in the sense that the leaves are at the bottom of the tree.
- The points along the tree where the predictor space is split are referred to as **internal nodes**.
- In the hitters tree, the two internal nodes are indicated by the text $\textit{Years} < 4.5$ and $\textit{Hits} < 117.5$.

Interpretation of Results

- Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players.
- Given that a player is less experienced, the number of Hits that he made in the previous year seems to play little role in his Salary.
- But among players who have been in the major leagues for five or more years, the number of Hits made in the previous year does affect Salary, and players who made more Hits last year tend to have higher salaries.
- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain.

Details of the tree-building process

1. We divide the predictor space — that is, the set of possible values for $X_1, X_2, \cdots, X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, \cdots, R_J$.
2. For every observation that falls into the region R_j , we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

More details of the tree-building process

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or **boxes**, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes $R_1, \cdots, R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

More details of the tree-building process

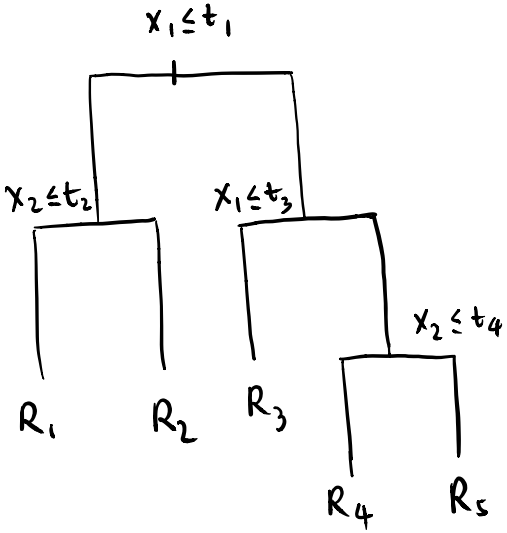
- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a **top-down**, **greedy** approach that is known as recursive binary splitting.
- The approach is **top-down** because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- It is **greedy** because at each step of the tree-building process, the **best** split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

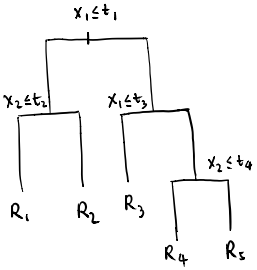
Details– Continued

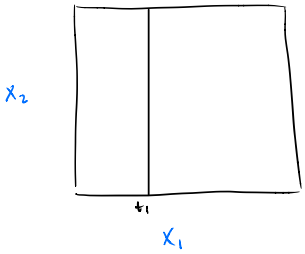
- We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.
- Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.
- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions.
- Again, we look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.

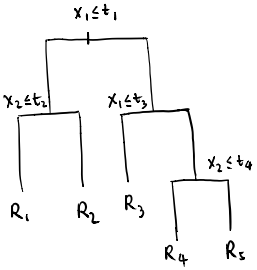
Predictions

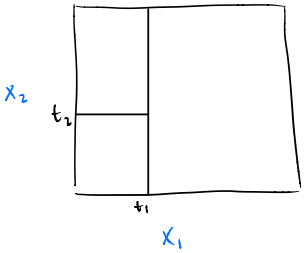
- We predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs.
- A five-region example of this approach is shown next.

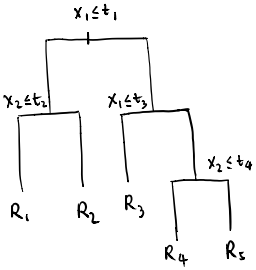

$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
R_1
R_2
R_3
R_4
R_5

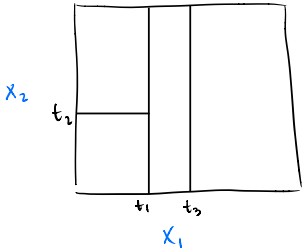

$X_1 \le t_1$
$X_2 \le t_2$
$X_1 \le t_3$
$X_2 \le t_4$
R_1
R_2
R_3
R_4
R_5

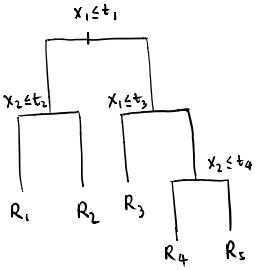

X_2
t_1
X_1

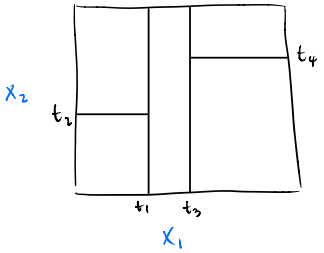

$X_1 \le t_1$
$X_2 \le t_2$
$X_1 \le t_3$
$X_2 \le t_4$
R_1
R_2
R_3
R_4
R_5

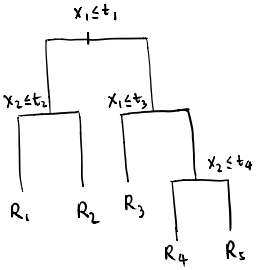

X_2
t_2
t_1
X_1

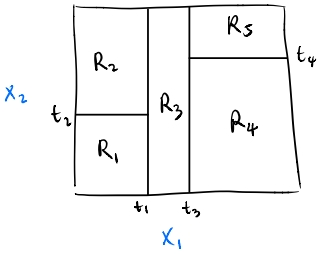

$X_1 \le t_1$
$X_2 \le t_2$
$X_1 \le t_3$
$X_2 \le t_4$
R_1
R_2
R_3
R_4
R_5


X_2
t_2
t_1
t_3
X_1

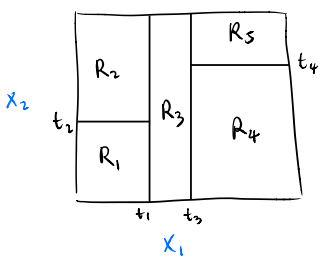

$X_1 \le t_1$
$X_2 \le t_2$
$X_1 \le t_3$
$X_2 \le t_4$
R_1
R_2
R_3
R_4
R_5

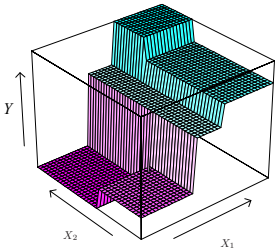

X_2
t_2
t_4
t_1
t_3
X_1


$X_1 \le t_1$
$X_2 \le t_2$
$X_1 \le t_3$
$X_2 \le t_4$
R_1
R_2
R_3
R_4
R_5


X_2
t_2
t_4
R_1
R_2
R_3
R_4
R_5
t_1
t_3
X_1

Prediction





To predict Y^* at $X^* = (X_1^*, X_2^*)$:

1. Find the region that X^* belongs to according to the tree or partition.
2. Suppose the region is R_j. Let $Y^* = Ave_{i \in R_j}(y_i)$.

Pruning a tree

- The process described above may produce good predictions on the training set, but is likely to **overfit** the data, leading to poor test set performance. **Why?**
- A smaller tree with fewer splits (that is, fewer regions $R_1, \cdots, R_J$) might lead to lower variance and better interpretation at the cost of a little bias.
- One possible alternative to the process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold.
- This strategy will result in smaller trees, but is too **short-sighted**: a seemingly worthless split early on in the tree might be followed by a very good split — that is, a split that leads to a large reduction in RSS later on. We will come back to this point after discussing classification tree.

Pruning a tree– continued

- A better strategy is to grow a very large tree T_0, and then **prune** it back in order to obtain a **subtree**.
- **Cost complexity pruning** — also known as **weakest link pruning** — is used to do this
- We consider a sequence of trees indexed by a nonnegative tuning parameter α. For each value of α

$$\min_{T \subset T_0} \sum_{j=1}^{J_T} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 + \alpha J_T$$

Here J_T is the number of regions or terminal nodes of the tree T, R_j is the rectangle (i.e. the subset of predictor space) corresponding to the jth terminal node, and $\hat{y}_{R_j}$ is the mean of the training observations in R_j.

Choosing the best subtree

- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.
- We select an optimal value $\hat{\alpha}$ using cross-validation.
- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$.

Summary: tree algorithm

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees for a range of α values.

3. Use K-fold cross-validation to choose α. For each $k = 1, \cdots, K$:

   - Repeat Steps 1 and 2 on the $\frac{K-1}{K}$th fraction of the training data, excluding the kth fold.
   - Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

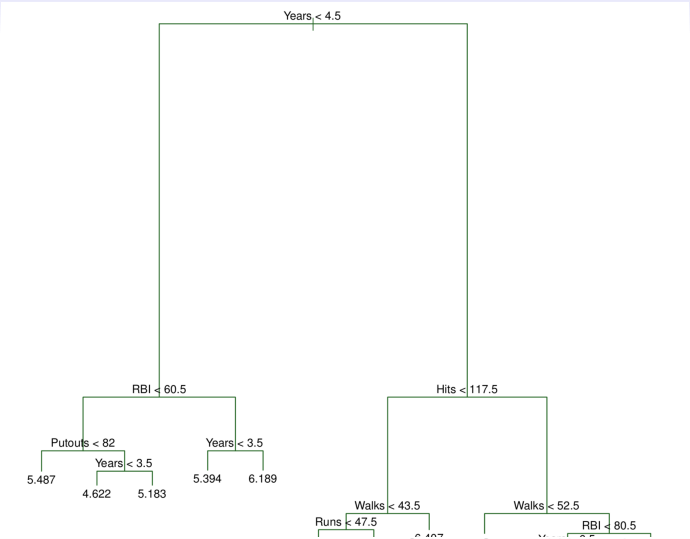
   Average the results, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

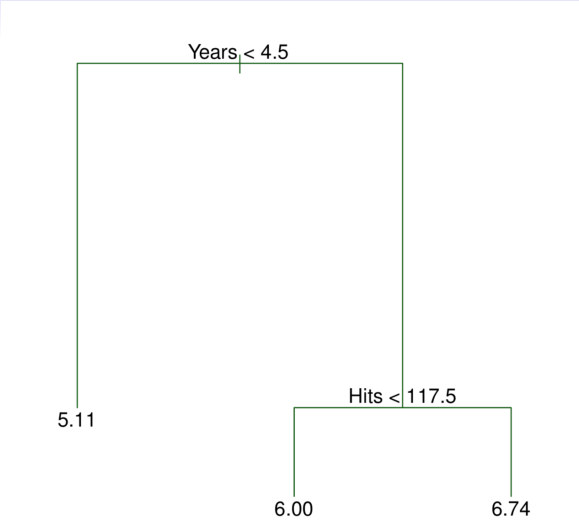
Baseball example continued

- First, we randomly divided the data set in half, yielding 132 observations in the training set and 131 observations in the test set.
- We then built a large regression tree on the training data and varied α in order to create subtrees with different numbers of terminal nodes.
- Finally, we performed six-fold cross-validation in order to estimate the cross-validated MSE of the trees as a function of α.

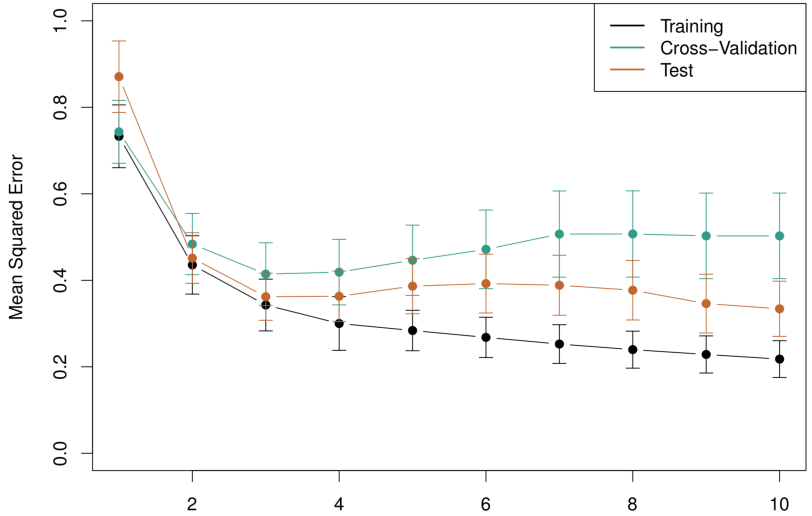
Unpruned tree



Pruned tree



Baseball example continued



Classification Trees

- Very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one.
- For a classification tree, we predict that each observation belongs to the **most commonly occurring class** of training observations in the region to which it belongs.

Details of classification trees

- Just as in the regression setting, we use recursive binary splitting to grow a classification tree.
- In the classification setting, RSS cannot be used as a criterion for making the binary splits
- A natural alternative to RSS is the **classification error rate**. This is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E_j = 1 - \max_k(\hat{p}_{jk}).$$

  Here $\hat{p}_{jk}$ represents the proportion of training observations in the jth region that are from the kth class.
- However classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable. Will explain later.

Gini index and Deviance

- The **Gini index** is defined by

$$G_j = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk}),$$

  a measure of total variance across the K classes. The Gini index takes on a small value if all of the $\hat{p}_{jk}$'s are close to zero or one.

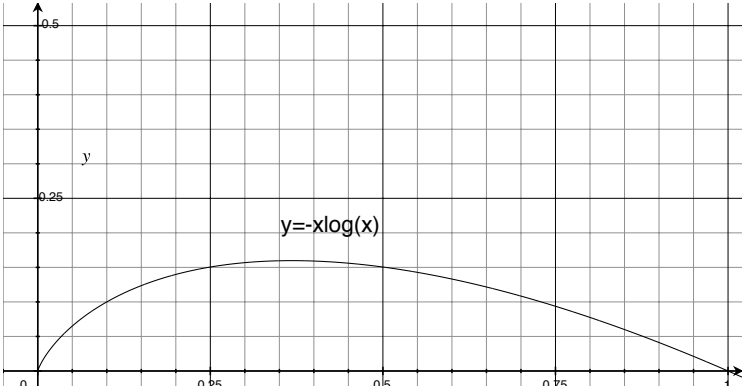
- For this reason the Gini index is referred to as a measure of node **purity** — a small value indicates that a node contains predominantly observations from a single class.

- The overall purity of a tree is a weighted average $\frac{1}{n}\sum_{j=1}^{J} n_j G_j$ with n_j being the number of training observations in region R_j.

Entropy

- An alternative to the Gini index is **entropy**, given by

$$D_j = -\sum_{k=1}^{K} \hat{p}_{jk} \log \hat{p}_{jk}.$$



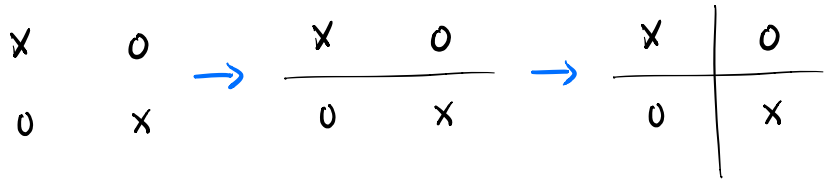
Comparison of classification error rate vs Gini index

Consider two scenarios for three-class classification

1. $p_1 = 50\%, p_2 = 25\%, p_3 = 25\%$
   - Classification error 0.5 vs Gini index 0.625
2. $p_1 = 50\%, p_2 = 50\%, p_3 = 0\%$
   - Classification error 0.5 vs Gini index 0.5

Gini index would favor second scenario as it is more pure whereas classification error rate cannot distinguish between the two.

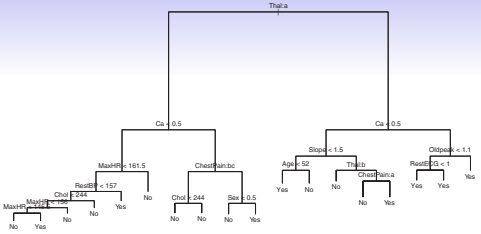
Why stopping growing a tree at some threshold is short-sighted?

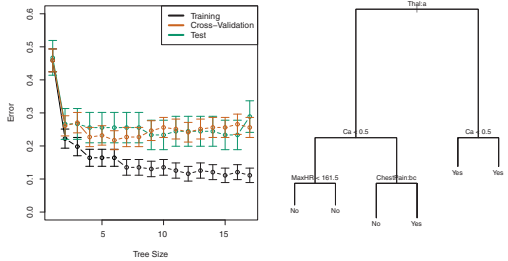


Gini 50% 50% 0%

Example: heart data

- These data contain a binary outcome *HD* for 303 patients who presented with chest pain.
- An outcome value of *Yes* indicates the presence of heart disease based on an angiographic test, while *No* means no heart disease.
- There are 13 predictors including *Age*, *Sex*, *Chol* (a cholesterol measurement), and other heart and lung function measurements.
- Cross-validation yields a tree with six terminal nodes. See next figure.


Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Slope < 1.5
Oldpeak < 1.1
RestBP < 157
Age < 52
Thal:b
RestECG < 1
Chol < 244
ChestPain:a
MaxHR < 156
Chol < 244
Sex < 0.5
No
Yes


Training
Cross−Validation
Test
Error
Tree Size
Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
No
Yes

Trees Versus Linear Models

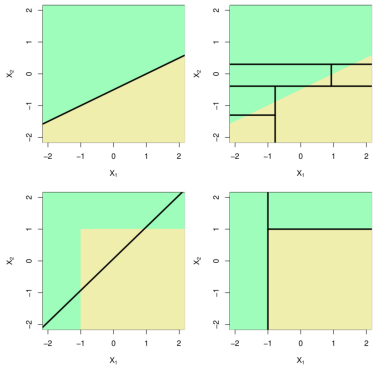


Figure: Top Row: True linear boundary; **Bottom row**: true non-linear boundary. **Left column**: linear model; **Right column**: tree-based model