Statistics 202: Statistical Aspects of Data Mining

Professor David Mease

Tuesday, Thursday 9:00-10:15 AM Terman 156

Lecture 12 = More of Chapter 5

Agenda:

1) Assign 5th Homework (due Tues 8/14 at 9AM)
2) Discuss Final Exam
3) Lecture over more of Chapter 5

Homework Assignment:

Chapter 5 Homework Part 2 and Chapter 8 Homework is due Tuesday 8/14 at 9AM.

Either email to me (dmease@stanford.edu), bring it to class, or put it under my office door.

SCPD students may use email or fax or mail.

The assignment is posted at

http://www.stats202.com/homework.html

Important: If using email, please submit only a single file (word or pdf) with your name and chapters in the file name. Also, include your name on the first page. Finally, please put your name and the homework # in the subject of the email.

Final Exam

I have obtained permission to have the final exam from 9 AM to 12 noon on Thursday 8/16 in the classroom (Terman 156)

I will assume the same people will take it off campus as with the midterm so please let me know if

1) You are SCPD and took the midterm on campus but need to take the final off campus

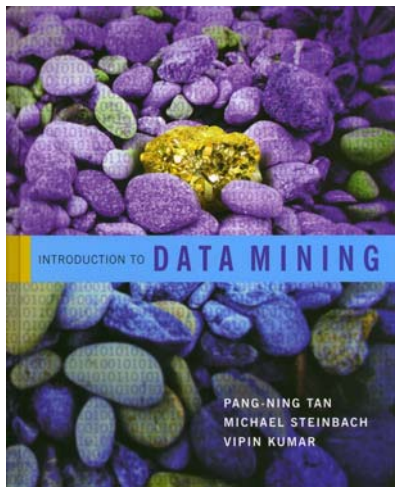
or

2) You are SCPD and took the midterm off campus but want to take the final on campus

More details to come...

Introduction to Data Mining

by
Tan, Steinbach, Kumar



Chapter 5: Classification: Alternative Techniques

The ROC Curve (Sec 5.7.2, p. 298)

- **ROC stands for Receiver Operating Characteristic**

- **Since we can "turn up" or "turn down" the number of observations being classified as the positive class, we can have many different values of true positive rate (TPR) and false positive rate (FPR) for the same classifier.**

$$\text{TPR} = \frac{TP}{TP + FN} \qquad \text{FPR} = \frac{FP}{FP + TN}$$

- **The ROC curve plots TPR on the y-axis and FPR on the x-axis**

The ROC Curve (Sec 5.7.2, p. 298)

- The ROC curve plots TPR on the y-axis and FPR on the x-axis
- The diagonal represents random guessing
- A good classifier lies near the upper left
- ROC curves are useful for comparing 2 classifiers
- The better classifier will lie on top more often
- The Area Under the Curve (AUC) is often used a metric

In class exercise #40:

This is textbook question #17 part (a) on page 322. It is part of your homework so we will not do all of it in class. We will just do the curve for M_1.

You are asked to evaluate the performance of two classification models, M_1 and M_2. The test set you have chosen contains 26 binary attributes, labeled as A through Z.

Table 5.5 shows the posterior probabilities obtained by applying the models to the test set. (Only the posterior probabilities for the positive class are shown). As this is a two-class problem, $P(-) = 1 - P(+)$ and $P(-|A, \ldots, Z) = 1 - P(+|A, \ldots, Z)$. Assume that we are mostly interested in detecting instances from the positive class.

Instance	True Class	$P(+\|A, \ldots, Z, M_1)$	$P(+\|A, \ldots, Z, M_2)$
1	+	0.73	0.61
2	+	0.69	0.03
3	−	0.44	0.68
4	−	0.55	0.31
5	+	0.67	0.45
6	+	0.47	0.09
7	−	0.08	0.38
8	−	0.15	0.05
9	+	0.45	0.01
10	−	0.35	0.04

(a) Plot the ROC curve for both M_1 and M_2. (You should plot them on the same graph.) Which model do you think is better? Explain your reasons.

In class exercise #41:

This is textbook question #17 part (b) on page 322.

(b) For model M_1, suppose you choose the cutoff threshold to be $t = 0.5$. In other words, any test instances whose posterior probability is greater than t will be classified as a positive example. Compute the precision, recall, and F-measure for the model at this threshold value.

Additional Classification Techniques

- **Decision trees are just one method for classification**

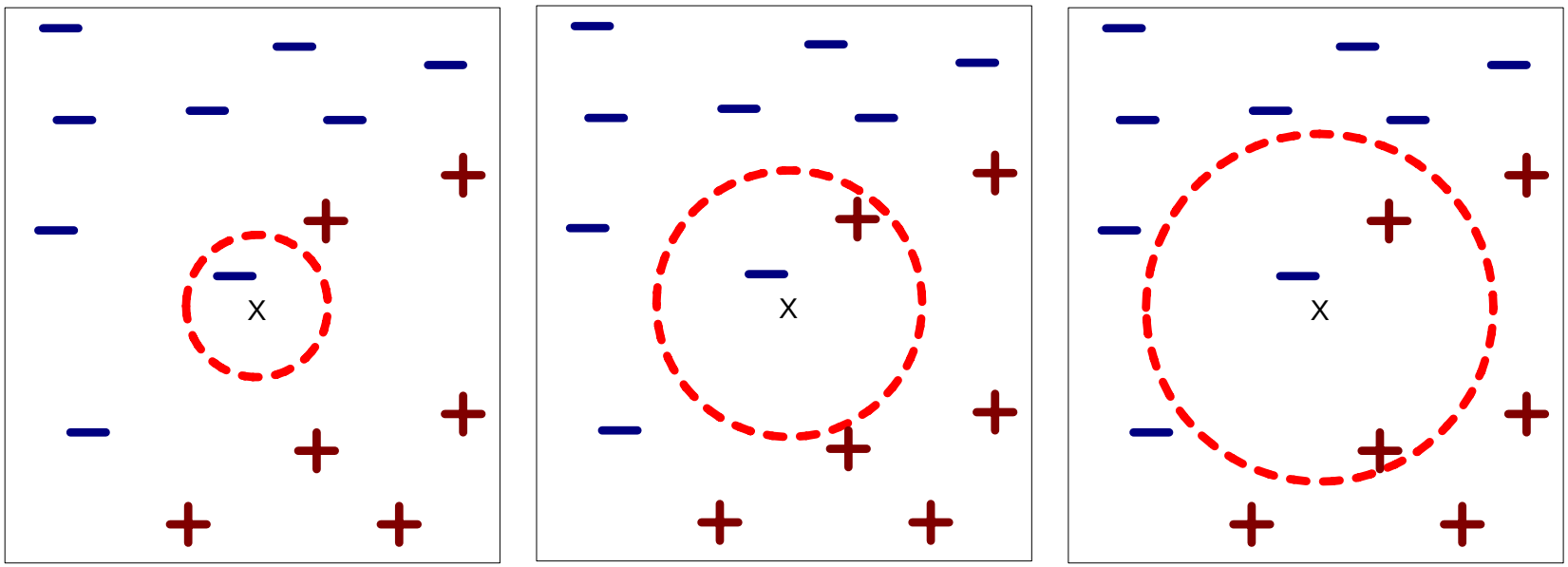
- **We will learn additional methods in this chapter:**

- Nearest Neighbor
- Support Vector Machines
- Bagging
- Random Forests
- Boosting

Nearest Neighbor (Section 5.2, page 223)

- **You can use nearest neighbor classifiers if you have some way of defining "distances" between attributes**

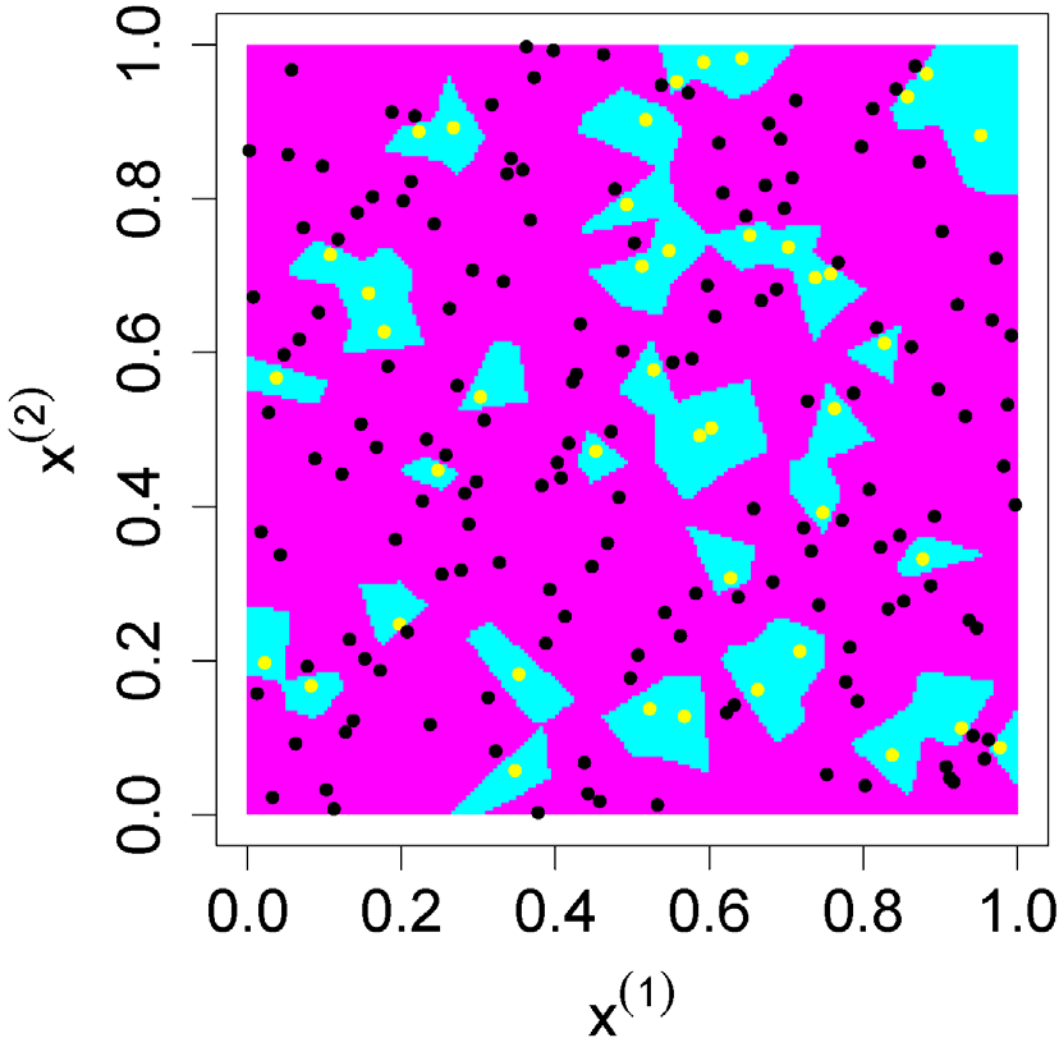
- **The *k*-nearest neighbor classifier classifies a point based on the majority of the *k* closest training points**



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Nearest Neighbor (Section 5.2, page 223)

- **Here is a plot I made using R showing the 1-nearest neighbor classifier on a 2-dimensional data set.**



Nearest Neighbor (Section 5.2, page 223)

- **Nearest neighbor methods work very poorly when the dimensionality is large (meaning there are a large number of attributes)**

- **The scales of the different attributes are important. If a single numeric attribute has a large spread, it can dominate the distance metric. A common practice is to scale all numeric attributes to have equal variance.**

- **The knn() function in R in the library "class" does a *k*-nearest neighbor classification using Euclidean distance.**

In class exercise #42:

Use knn() in R to fit the 1-nearest-nieghbor classifier to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Use all the default values. Compute the misclassification error on the training data and also on the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv

In class exercise #42:

Use knn() in R to fit the 1-nearest-nieghbor classifier to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Use all the default values. Compute the misclassification error on the training data and also on the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv

Solution:

```
install.packages("class")
library(class)
train<-read.csv("sonar_train.csv",header=FALSE)
y<-as.factor(train[,61])
x<-train[,1:60]
fit<-knn(x,x,y)
1-sum(y==fit)/length(y)
```

In class exercise #42:

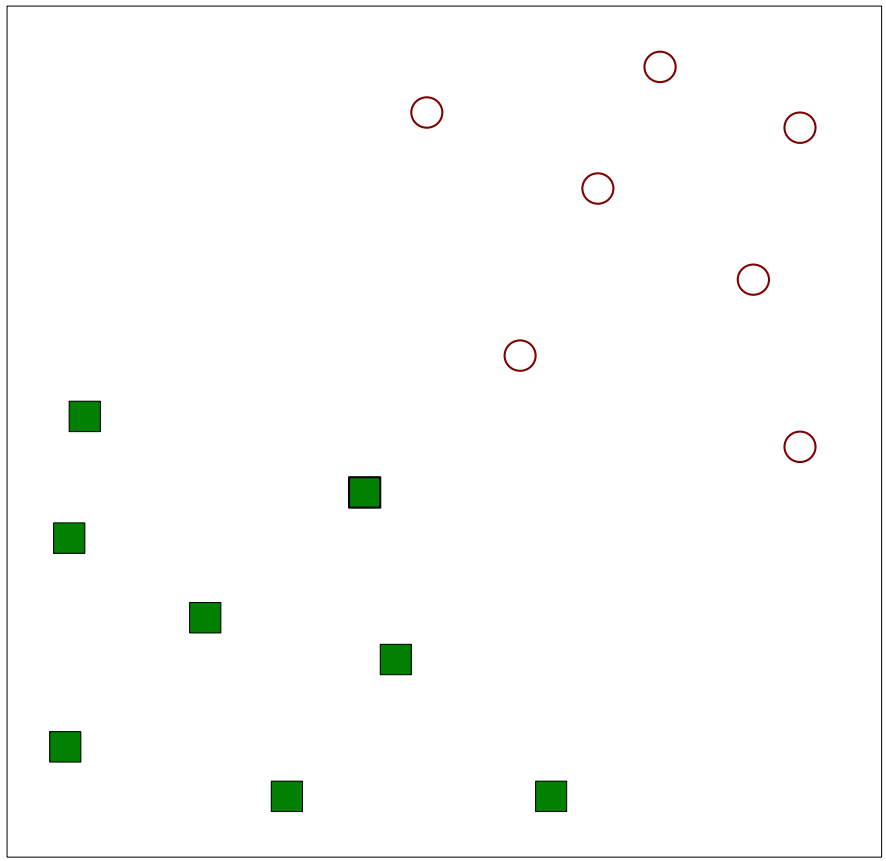
Use knn() in R to fit the 1-nearest-nieghbor classifier to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Use all the default values. Compute the misclassification error on the training data and also on the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv

Solution (continued):

```
test<-read.csv("sonar_test.csv",header=FALSE)
y_test<-as.factor(test[,61])
x_test<-test[,1:60]
fit_test<-knn(x,x_test,y)
1-sum(y_test==fit_test)/length(y_test)
```

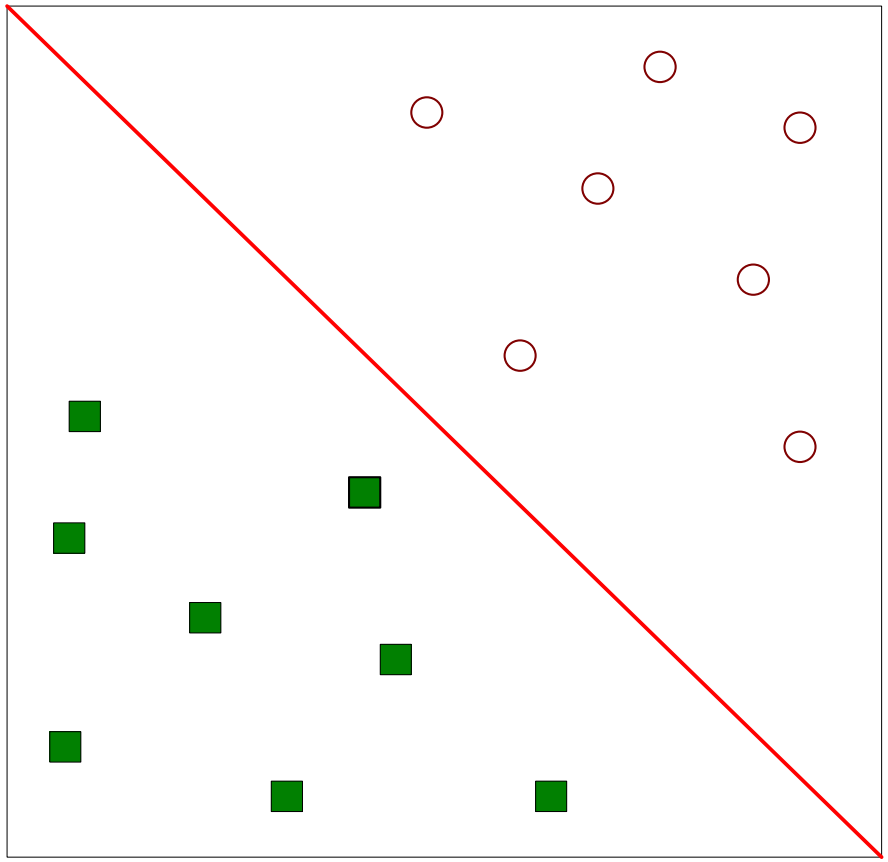
Support Vector Machines (Section 5.5, page 256)

- **If the two classes can be separated perfectly by a line in the x space, how do we choose the "best" line?**



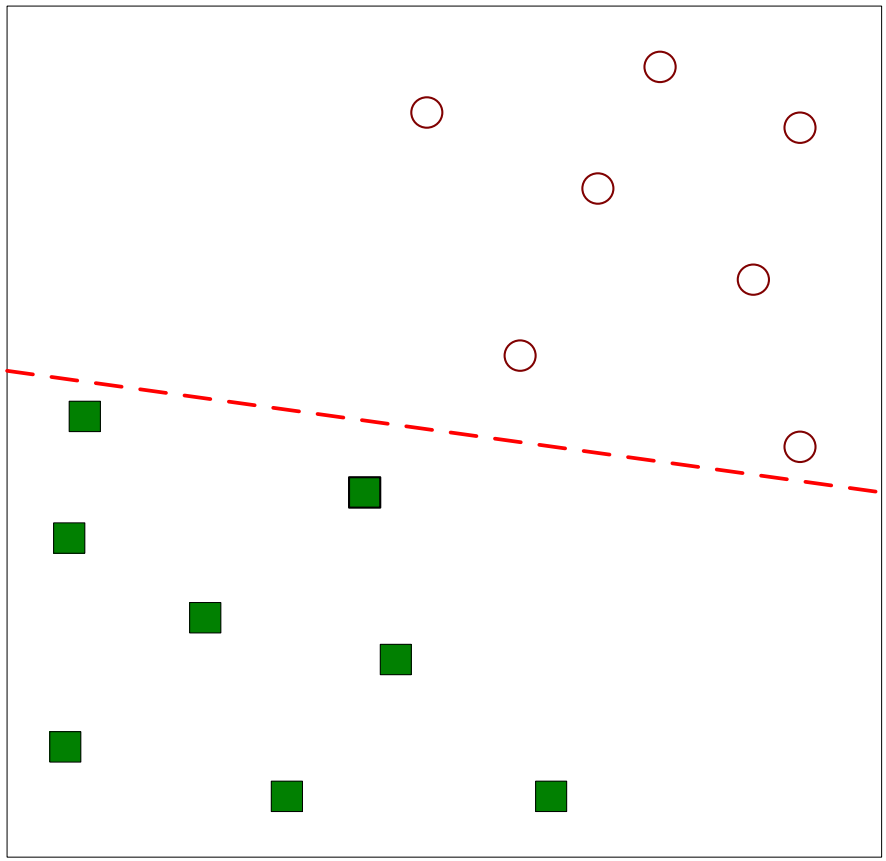
Support Vector Machines (Section 5.5, page 256)

- **If the two classes can be separated perfectly by a line in the x space, how do we choose the "best" line?**



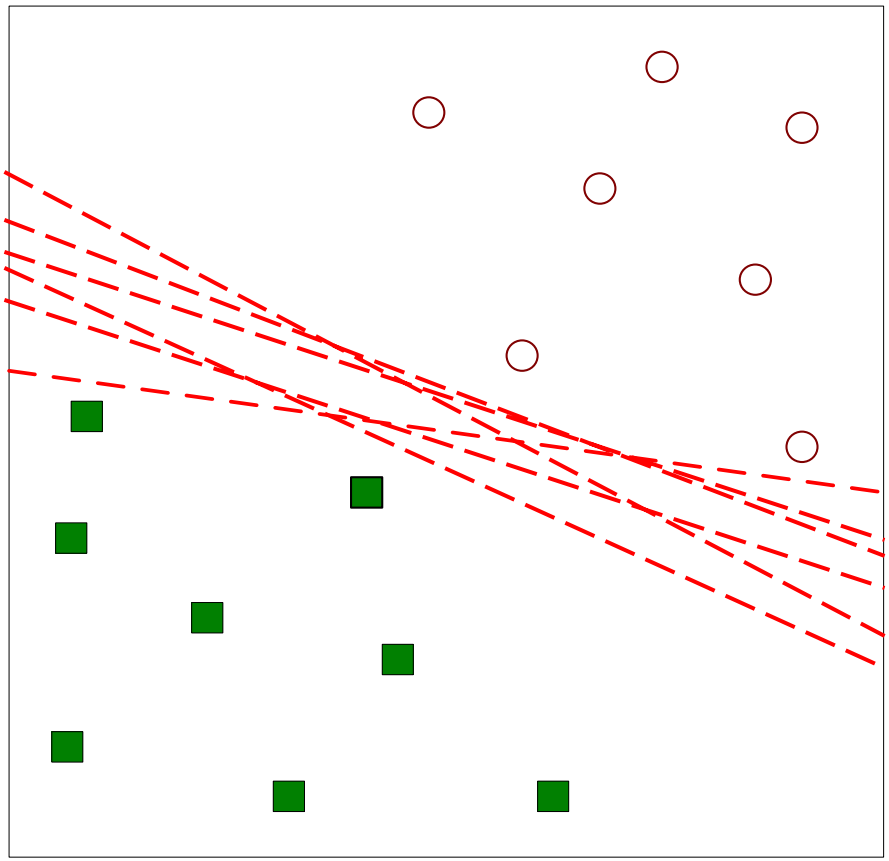
Support Vector Machines (Section 5.5, page 256)

- If the two classes can be separated perfectly by a line in the x space, how do we choose the “best” line?



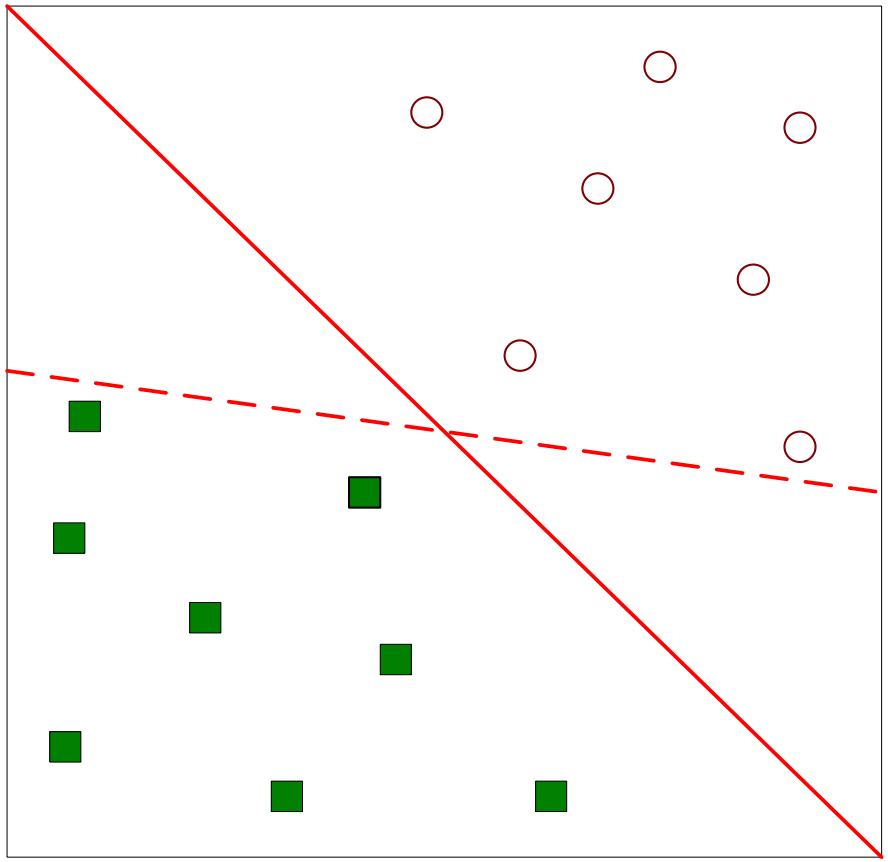
Support Vector Machines (Section 5.5, page 256)

- If the two classes can be separated perfectly by a line in the x space, how do we choose the “best” line?



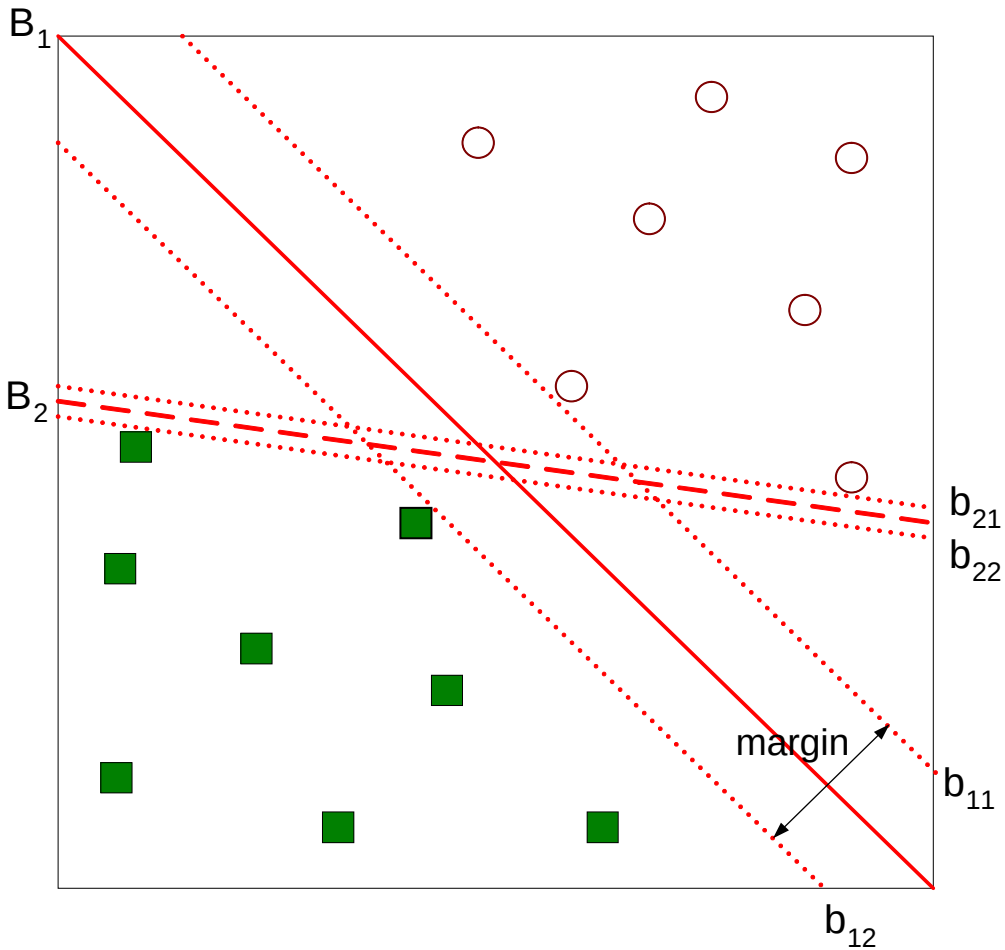
Support Vector Machines (Section 5.5, page 256)

- **If the two classes can be separated perfectly by a line in the x space, how do we choose the "best" line?**



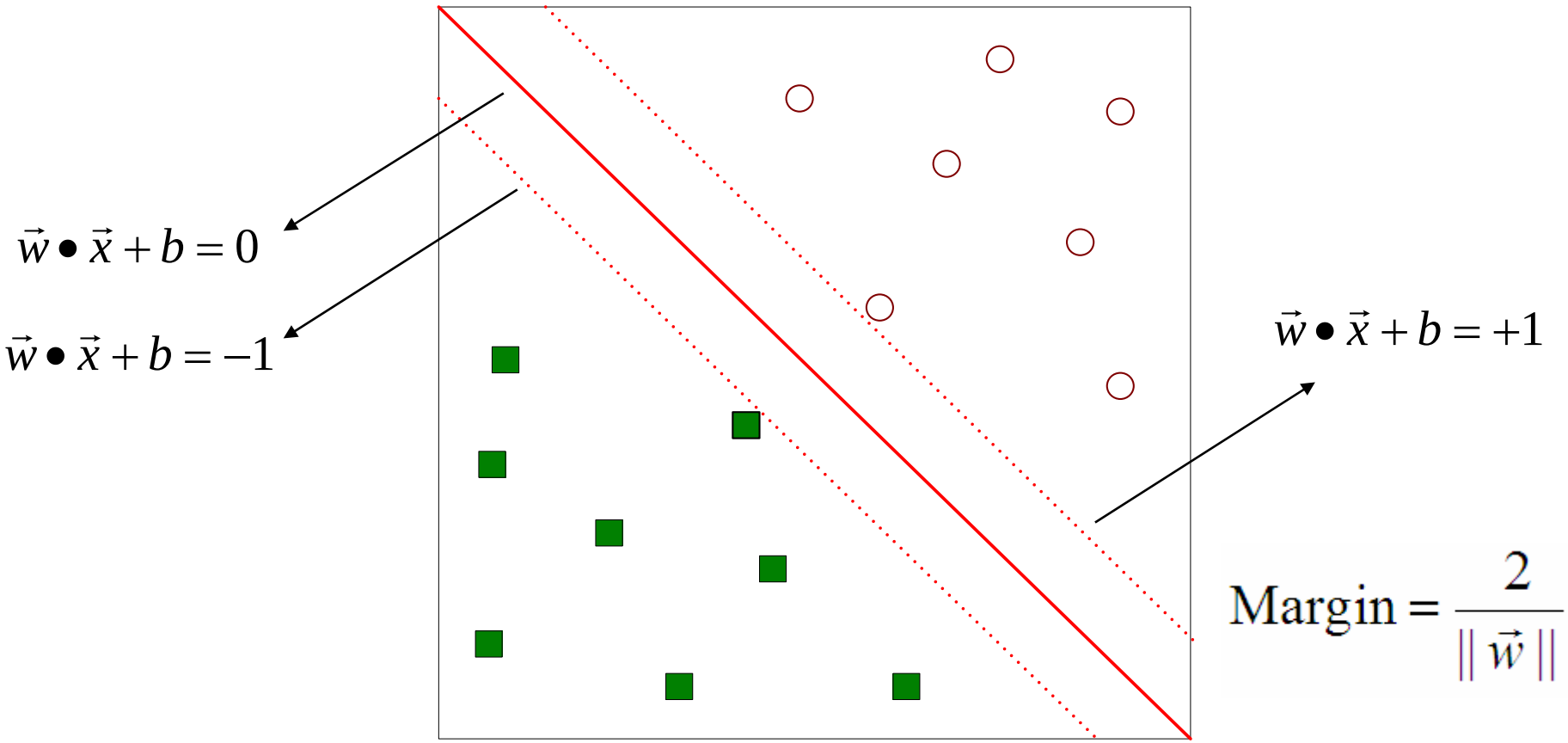
Support Vector Machines (Section 5.5, page 256)

- **One solution is to choose the line (hyperplane) with the largest *margin*. The margin is the distance between the two parallel lines on either side.**



Support Vector Machines (Section 5.5, page 256)

- **Here is the notation your book uses:**



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

Support Vector Machines (Section 5.5, page 256)

- **This can be formulated as a constrained optimization problem.**

- **We want to maximize** $\text{Margin} = \dfrac{2}{\|\vec{w}\|}$

- **This is equivalent to minimizing** $L(w) = \dfrac{\|\vec{w}\|^2}{2}$

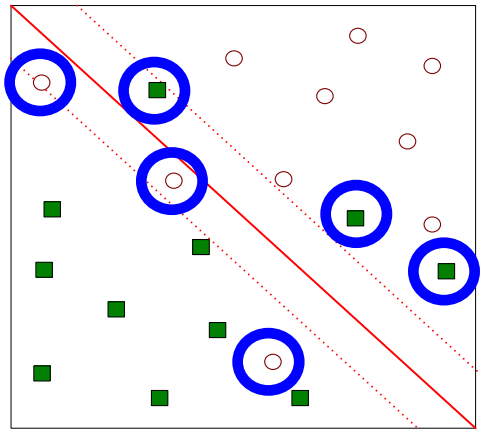
- **We have the following constraints**

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 \end{cases}$$

- **So we have a quadratic objective function with linear constraints which means it is a convex optimization problem and we can use Lagrange multipliers**

Support Vector Machines (Section 5.5, page 256)

- **What if the problem is not linearly separable?**



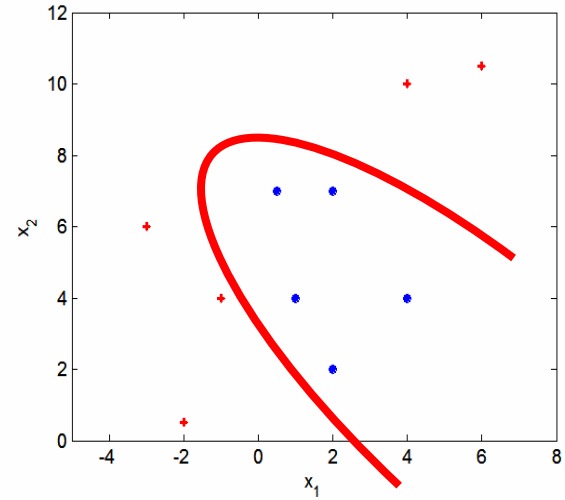
- **Then we can introduce slack variables:**

Minimize $L(w) = \frac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k \right)$

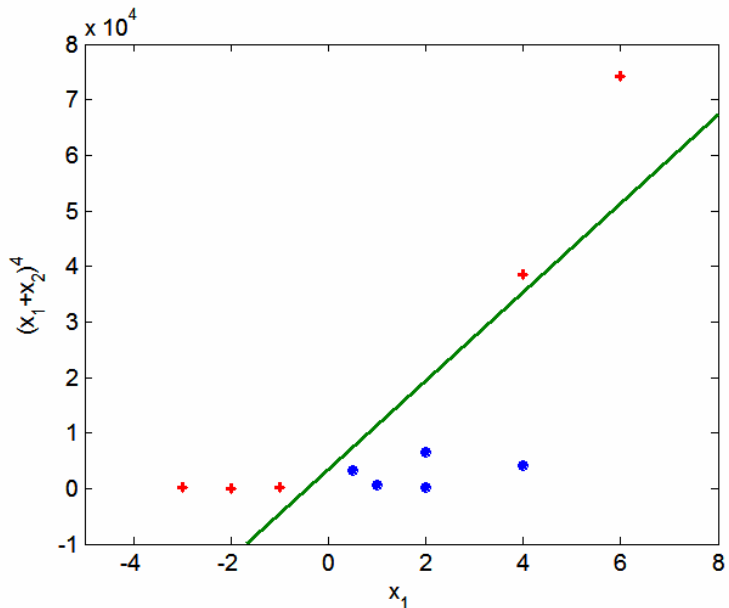
Subject to $f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq 1 - \xi_i \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \end{cases}$

Support Vector Machines (Section 5.5, page 256)

- **What if the boundary is not linear?**



- **Then we can use transformations of the variables to map into a higher dimensional space**



Support Vector Machines in R

- **The function svm in the package e1071 can fit support vector machines in R**

- **Note that the default *kernel* is not linear – use kernel="linear" to get a linear kernel**

svm(e1071) R Documentation

Support Vector Machines

Description

svm is used to train a support vector machine. It can be used to carry out general regression and classification (of nu and epsilon-type), as well as density-estimation. A formula interface is provided.

Usage

```
## S3 method for class 'formula':
svm(formula, data = NULL, ..., subset, na.action =
na.omit, scale = TRUE)
## Default S3 method:
svm(x, y = NULL, scale = TRUE, type = NULL, kernel =
"radial", degree = 3, gamma = if (is.vector(x)) 1 else 1 / ncol(x),
coef0 = 0, cost = 1, nu = 0.5,
class.weights = NULL, cachesize = 40, tolerance = 0.001, epsilon = 0.1,
shrinking = TRUE, cross = 0, probability = FALSE, fitted = TRUE,
..., subset, na.action = na.omit)
```

In class exercise #43:

Use svm() in R to fit the default svm to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Compute the misclassification error on the training data and also on the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv

In class exercise #43:

Use svm() in R to fit the default svm to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Compute the misclassification error on the training data and also on the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv

Solution:

```
install.packages("e1071")
library(e1071)
train<-read.csv("sonar_train.csv",header=FALSE)
y<-as.factor(train[,61])
x<-train[,1:60]
fit<-svm(x,y)
1-sum(y==predict(fit,x))/length(y)
```

In class exercise #43:

Use svm() in R to fit the default svm to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Compute the misclassification error on the training data and also on the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv

Solution (continued):

```
test<-read.csv("sonar_test.csv",header=FALSE)
y_test<-as.factor(test[,61])
x_test<-test[,1:60]
1-sum(y_test==predict(fit,x_test))/length(y_test)
```

In class exercise #44:

Use svm() in R with kernel="linear“ and cost=100000 to fit the toy 2-dimensional data below. Provide a plot of the resulting classification rule.

x_1	x_2	y
0	0.1	-1
0.8	0.9	-1
0.4	0.5	-1
0.3	0.7	-1
0.1	0.4	-1
0.7	0.3	1
0.5	0.2	1
0.8	0.6	1
0.8	0	1
0.8	0.3	1

In class exercise #44:

Use svm() in R with kernel="linear“ and cost=100000 to fit the toy 2-dimensional data below. Provide a plot of the resulting classification rule.

x_1	x_2	y
0	0.1	-1
0.8	0.9	-1
0.4	0.5	-1
0.3	0.7	-1
0.1	0.4	-1
0.7	0.3	1
0.5	0.2	1
0.8	0.6	1
0.8	0	1
0.8	0.3	1

Solution:

```
x<-matrix(c(0,.1,.8,.9,.4,.5,
.3,.7,.1,.4,.7,.3,.5,.2,.8,.6,.8,0,.8,.3),
ncol=2,byrow=T)

y<-as.factor(c(rep(-1,5),rep(1,5)))

plot(x,pch=19,xlim=c(0,1),ylim=c(0,1),
col=2*as.numeric(y),cex=2,
xlab=expression(x[1]),ylab=expression(x[2]))
```

In class exercise #44:

Use svm() in R with kernel="linear“ and cost=100000 to fit the toy 2-dimensional data below. Provide a plot of the resulting classification rule.

x_1	x_2	y
0	0.1	-1
0.8	0.9	-1
0.4	0.5	-1
0.3	0.7	-1
0.1	0.4	-1
0.7	0.3	1
0.5	0.2	1
0.8	0.6	1
0.8	0	1
0.8	0.3	1

Solution (continued):

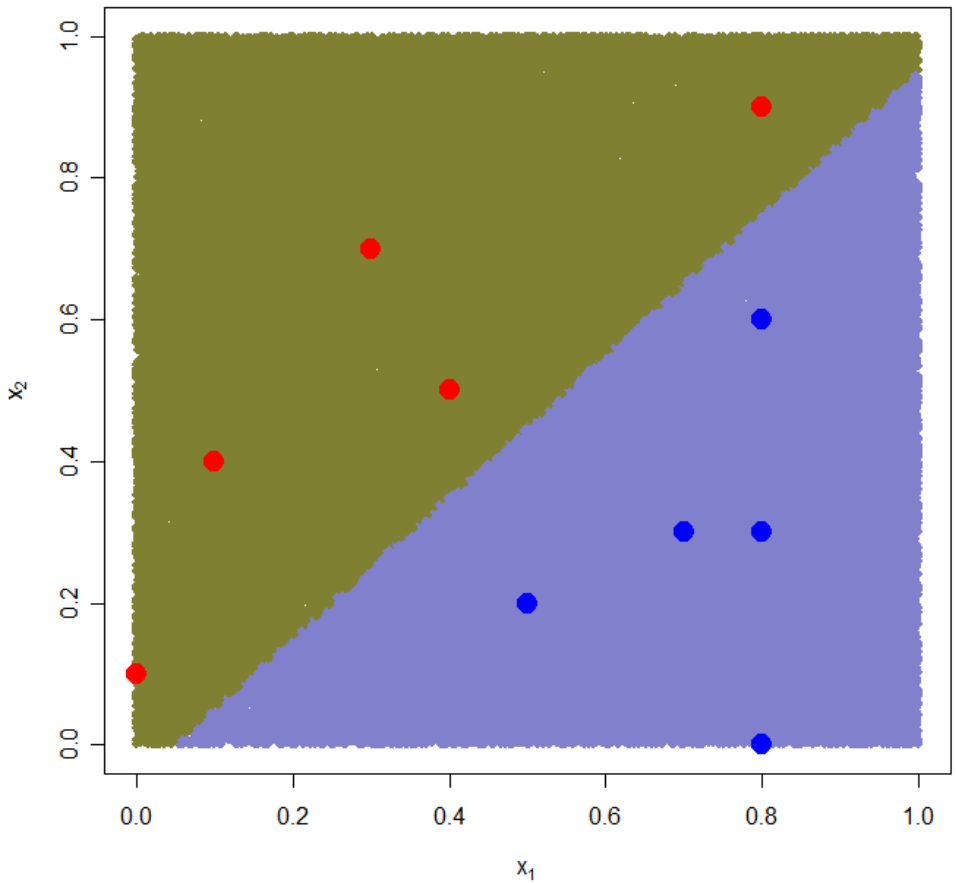
```
fit<-svm (x,y,kernel="linear",cost=100000)

big_x<-matrix(runif(200000),ncol=2,byrow=T)

points(big_x,col=rgb(.5,.5,
.2+.6*as.numeric(predict(fit,big_x)==1)),pch=19)

points(x,pch=19,col=2*as.numeric(y),cex=2)
```

In class exercise #44:

Use svm() in R with kernel="linear“ and cost=100000 to fit the toy 2-dimensional data below. Provide a plot of the resulting classification rule.

x_1	x_2	y
0	0.1	-1
0.8	0.9	-1
0.4	0.5	-1
0.3	0.7	-1
0.1	0.4	-1
0.7	0.3	1
0.5	0.2	1
0.8	0.6	1
0.8	0	1
0.8	0.3	1

Solution (continued):



Ensemble Methods (Section 5.6, page 276)

- **Ensemble methods aim at "improving classification accuracy by aggregating the predictions from multiple classifiers" (page 276)**

- **One of the most obvious ways of doing this is simply by averaging classifiers which make errors somewhat independently of each other**

In class exercise #45:

Suppose I have 5 classifiers which each classify a point correctly 70% of the time. If these 5 classifiers are completely independent and I take the majority vote, how often is the majority vote correct for that point?

In class exercise #45:

Suppose I have 5 classifiers which each classify a point correctly 70% of the time. If these 5 classifiers are completely independent and I take the majority vote, how often is the majority vote correct for that point?

Solution (continued):

```
10*.7^3*.3^2 + 5*.7^4*.3^1 + .7^5
```

or

```
1-pbinom(2, 5, .7)
```

In class exercise #46:

Suppose I have 101 classifiers which each classify a point correctly 70% of the time. If these 101 classifiers are completely independent and I take the majority vote, how often is the majority vote correct for that point?

In class exercise #46:

Suppose I have 101 classifiers which each classify a point correctly 70% of the time. If these 101 classifiers are completely independent and I take the majority vote, how often is the majority vote correct for that point?

Solution (continued):

```
1-pbinom(50, 101, .7)
```

Ensemble Methods (Section 5.6, page 276)

- **Ensemble methods include**

-Bagging (page 283)

-Random Forests (page 290)

-Boosting (page 285)

- **Bagging builds different classifiers by training on repeated samples (with replacement) from the data**

- **Random Forests averages many trees which are constructed with some amount of randomness**

- **Boosting combines simple base classifiers by upweighting data points which are classified incorrectly**

Random Forests (Section 5.6.6, page 290)

- One way to create random forests is to grow decision trees top down but at each terminal node consider only a random subset of attributes for splitting instead of all the attributes

- Random Forests are a very effective technique

- They are based on the paper

L. Breiman. Random forests. Machine Learning, 45:5-32, 2001

- They can be fit in R using the function randomForest() in the library randomForest

In class exercise #47:

Use randomForest() in R to fit the default Random Forest to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Compute the misclassification error for the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv

In class exercise #47:

Use randomForest() in R to fit the default Random Forest to the last column of the sonar training data at http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Compute the misclassification error for the test data at http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv

Solution:

```
install.packages("randomForest")
library(randomForest)
train<-read.csv("sonar_train.csv",header=FALSE)
test<-read.csv("sonar_test.csv",header=FALSE)
y<-as.factor(train[,61])
x<-train[,1:60]
y_test<-as.factor(test[,61])
x_test<-test[,1:60]
fit<-randomForest(x,y)
1-sum(y_test==predict(fit,x_test))/length(y_test)
```

Boosting (Section 5.6.5, page 285)

- **Boosting has been called the "best off-the-shelf classifier in the world"**

- **There are a number of explanations for boosting, but it is not completely understood why it works so well**

- **The most popular algorithm is AdaBoost from**

Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, pages 148–156, 1996.

Boosting (Section 5.6.5, page 285)

- **Boosting can use any classifier as its *weak learner* (base classifier) but decision trees are by far the most popular**

- **Boosting usually gives zero training error, but rarely overfits which is very curious**

Boosting (Section 5.6.5, page 285)

- **Boosting works by upweighing points at each iteration which are misclassified**

- **On paper, boosting looks like an optimization (similar to maximum likelihood estimation), but in practice it seems to benefit a lot from averaging like Random Forests does**

- **There exist R libraries for boosting, but these are written by statisticians who have their own views of boosting, so I would not encourage you to use them**

- **The best thing to do is to write code yourself since the algorithms are very basic**

AdaBoost

- **Here is a version of the AdaBoost algorithm**

- Fit the classifier g_m to the training data using weights w_i where g_m maps each x_i to -1 or 1.
- Compute the weighted error rate $\epsilon_m \equiv \sum_{i=1}^{n} w_i \mathrm{I}[y_i \neq g_m(x_i)]$ and half its log-odds, $\alpha_m \equiv \frac{1}{2} \log \frac{1-\epsilon_m}{\epsilon_m}$.
- Let $F_m = F_{m-1} + \alpha_m g_m$.
- Replace the weights w_i with $w_i \equiv w_i e^{-\alpha_m g_m(x_i) y_i}$ and then renormalize by replacing each w_i by $w_i / (\sum w_i)$.

- **The algorithm repeats until a chosen stopping time**

- **The final classifier is based on the sign of F_m**

In class exercise #48:

Use R to fit the AdaBoost classifier to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Plot the misclassification error for the training data and the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv
as a function of the iterations. Run the algorithm for 500 iterations. Use default rpart() as the base learner.

Solution:

```
train<-read.csv("sonar_train.csv",header=FALSE)
test<-read.csv("sonar_test.csv",header=FALSE)
y<-train[,61]
x<-train[,1:60]
y_test<-test[,61]
x_test<-test[,1:60]
```

In class exercise #48:

Use R to fit the AdaBoost classifier to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Plot the misclassification error for the training data and the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv
as a function of the iterations. Run the algorithm for 500 iterations. Use default rpart() as the base learner.

Solution (continued):

```
train_error<-rep(0,500)
test_error<-rep(0,500)
f<-rep(0,130)
f_test<-rep(0,78)
i<-1
library(rpart)
```

In class exercise #48:

Use R to fit the AdaBoost classifier to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Plot the misclassification error for the training data and the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv
as a function of the iterations. Run the algorithm for 500 iterations. Use default rpart() as the base learner.

Solution (continued):

```
while(i<=500){
  w<-exp(-y*f)
  w<-w/sum(w)
  fit<-rpart(y~.,x,w,method="class")
  g<--1+2*(predict(fit,x)[,2]>.5)
  g_test<--1+2*(predict(fit,x_test)[,2]>.5)
  e<-sum(w*(y*g<0))
```

In class exercise #48:

Use R to fit the AdaBoost classifier to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Plot the misclassification error for the training data and the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv
as a function of the iterations. Run the algorithm for 500 iterations. Use default rpart() as the base learner.

Solution (continued):

```
  alpha<-.5*log ( (1-e) / e )
  f<-f+alpha*g
  f_test<-f_test+alpha*g_test
  train_error[i]<-sum(1*f*y<0)/130
  test_error[i]<-sum(1*f_test*y_test<0)/78
  i<-i+1
}
```

In class exercise #48:

Use R to fit the AdaBoost classifier to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Plot the misclassification error for the training data and the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv
as a function of the iterations. Run the algorithm for 500 iterations. Use default rpart() as the base learner.

Solution (continued):

```
plot(seq(1,500),test_error,type="l",
   ylim=c(0,.5),
   ylab="Error Rate",xlab="Iterations",lwd=2)
lines(train_error,lwd=2,col="purple")
legend(4,.5,c("Training Error","Test Error"),
    col=c("purple","black"),lwd=2)
```

In class exercise #48:

Use R to fit the AdaBoost classifier to the last column of the sonar training data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_train.csv
Plot the misclassification error for the training data and the test data at
http://www-stat.wharton.upenn.edu/~dmease/sonar_test.csv
as a function of the iterations. Run the algorithm for 500 iterations. Use default rpart() as the base learner.

Solution (continued):