Statistics 202: Statistical Aspects of Data Mining

Professor Rajan Patel

Monday,Wednesday 4:15-5:30 in Gates B1

Lecture 9 = More of Chapter 5

Agenda:

1)Reminder about Homework #4 (due Wed)

2) Lecture over more of Chapter 5

Homework Assignment:

Homework #4 is due today (Wednesday 7/28)

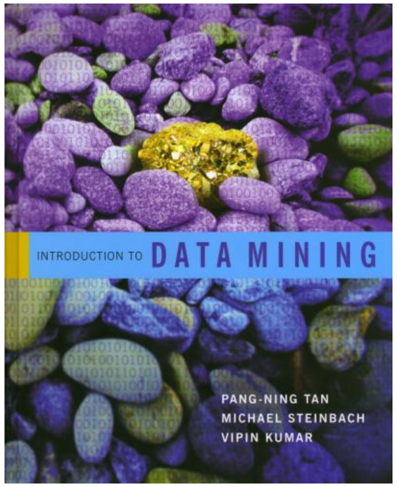
E-mail it to stats202@gmail.com as a pdf file, if possible.

The assignment is posted at
http://sites.google.com/site/stats202/homework

*If emailing, please send a single file (pdf is preferred) and make sure your name is on the first page and in the body of the email. Also, the file name should say "homework4" and your name.

Introduction to Data Mining

by
Tan, Steinbach, Kumar



Chapter 5: Classification: Alternative Techniques

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

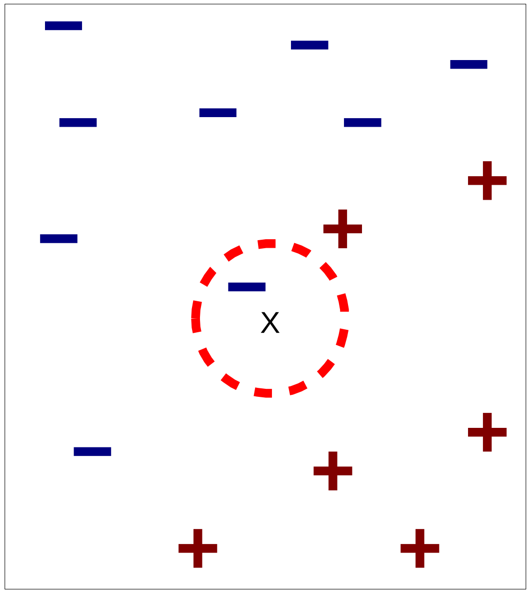
- Boosting

- Naïve Bayes

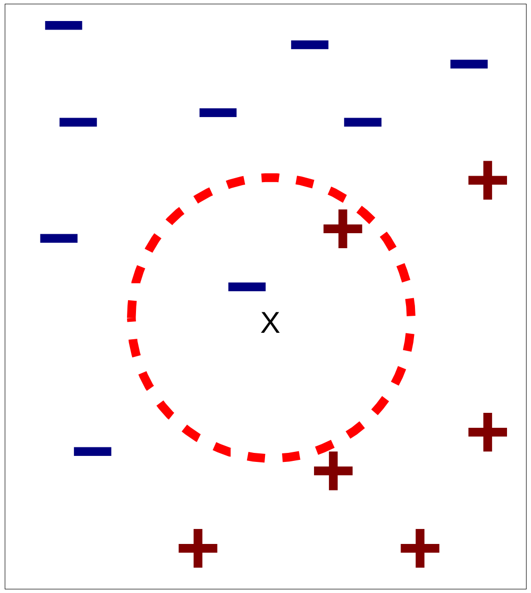
Nearest Neighbor (Section 5.2, page 223)

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

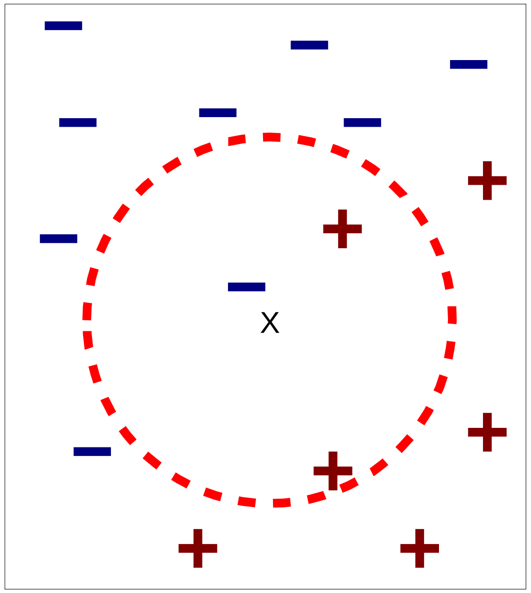
- **The *k*-nearest neighbor classifier classifies a point based on the majority of the *k* closest training points**



(a) 1-nearest neighbor



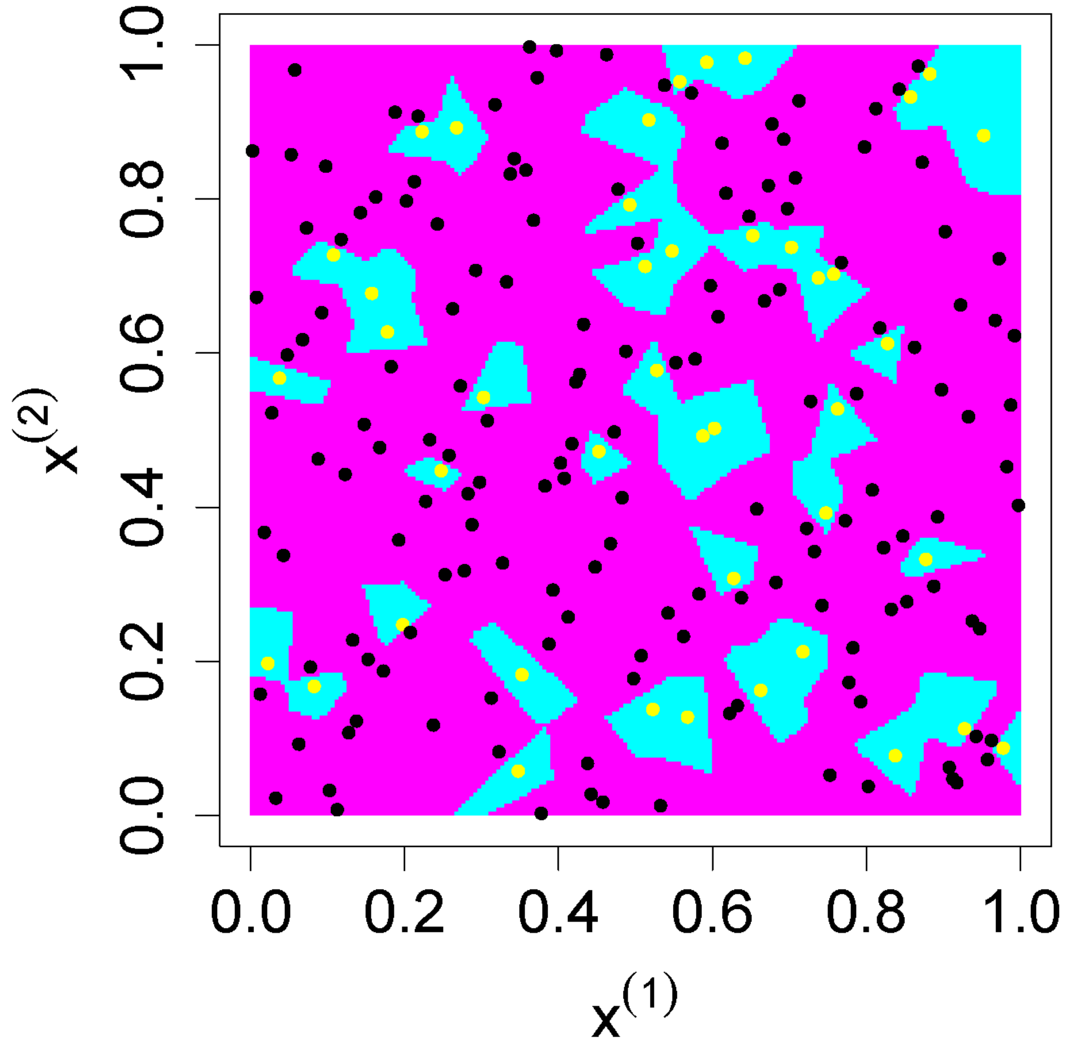
(b) 2-nearest neighbor



(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 #34:

Use knn() in R to fit the 1-nearest-neighbor classifier to the last column of the sonar training data at

http://sites.google.com/site/stats202/data/sonar_train.csv

Use all the default values. Compute the misclassification error on the training data and also on the test data at

http://sites.google.com/site/stats202/data/sonar_test.csv

In class exercise #34:

Use knn() in R to fit the 1-nearest-neighbor classifier to the last column of the sonar training data at
http://sites.google.com/site/stats202/data/sonar_train.csv
Use all the default values. Compute the misclassification error on the training data and also on the test data at
http://sites.google.com/site/stats202/data/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,k=1)
1-sum(y==fit)/length(y)
```

In class exercise #34:

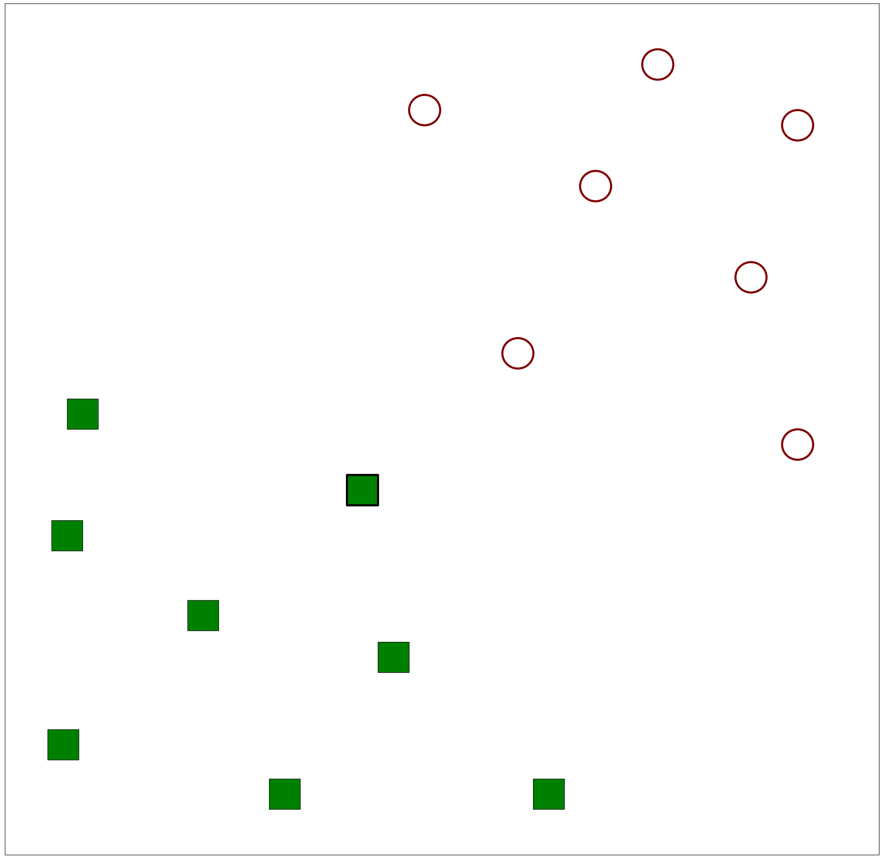
Use knn() in R to fit the 1-nearest-neighbor classifier to the last column of the sonar training data at
http://sites.google.com/site/stats202/data/sonar_train.csv
Use all the default values. Compute the misclassification error on the training data and also on the test data at
http://sites.google.com/site/stats202/data/sonar_test.csv

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,k=1)
1-sum(y_test==fit_test)/length(y_test)
```

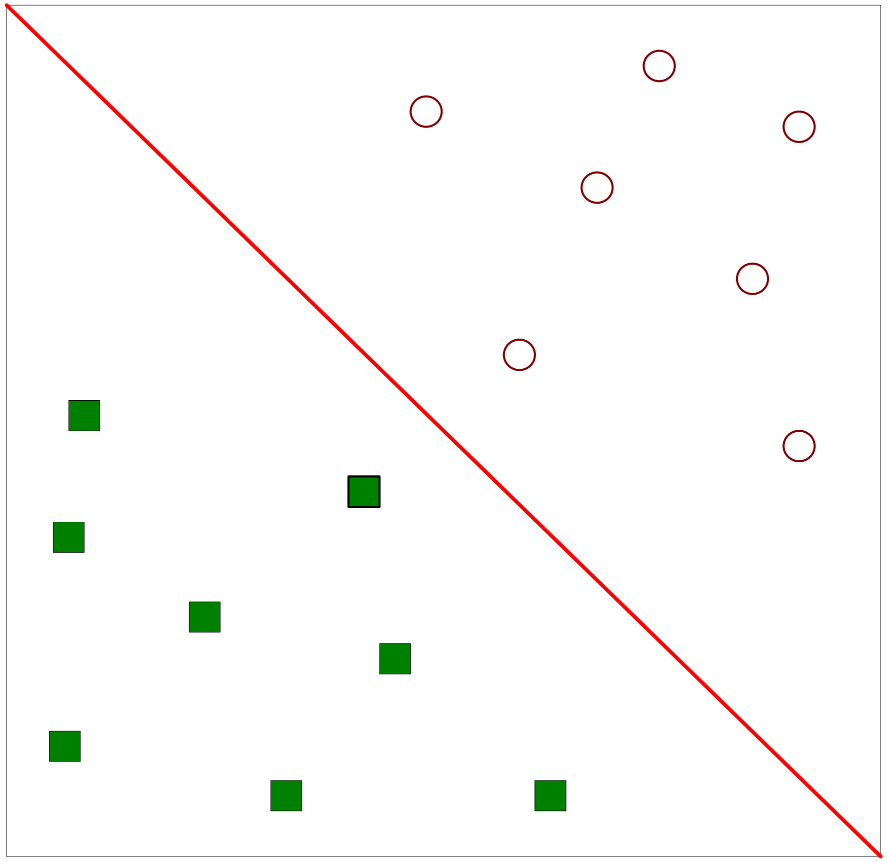
Support Vector Machines (Section 5.5, page 256)

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



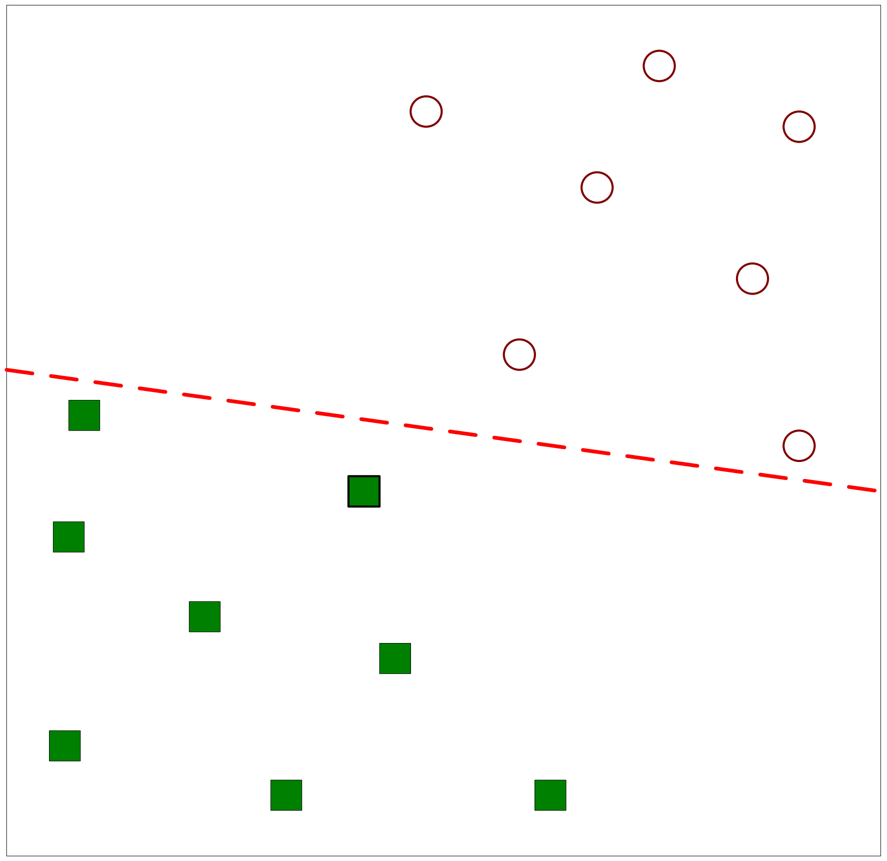
Support Vector Machines (Section 5.5, page 256)

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



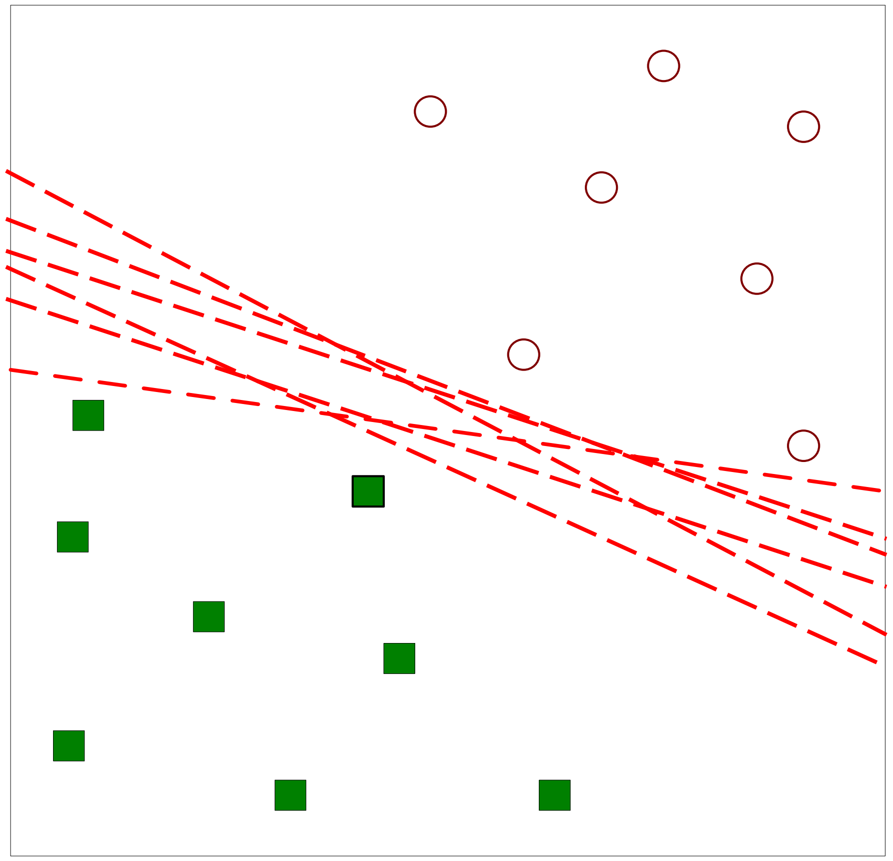
Support Vector Machines (Section 5.5, page 256)

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



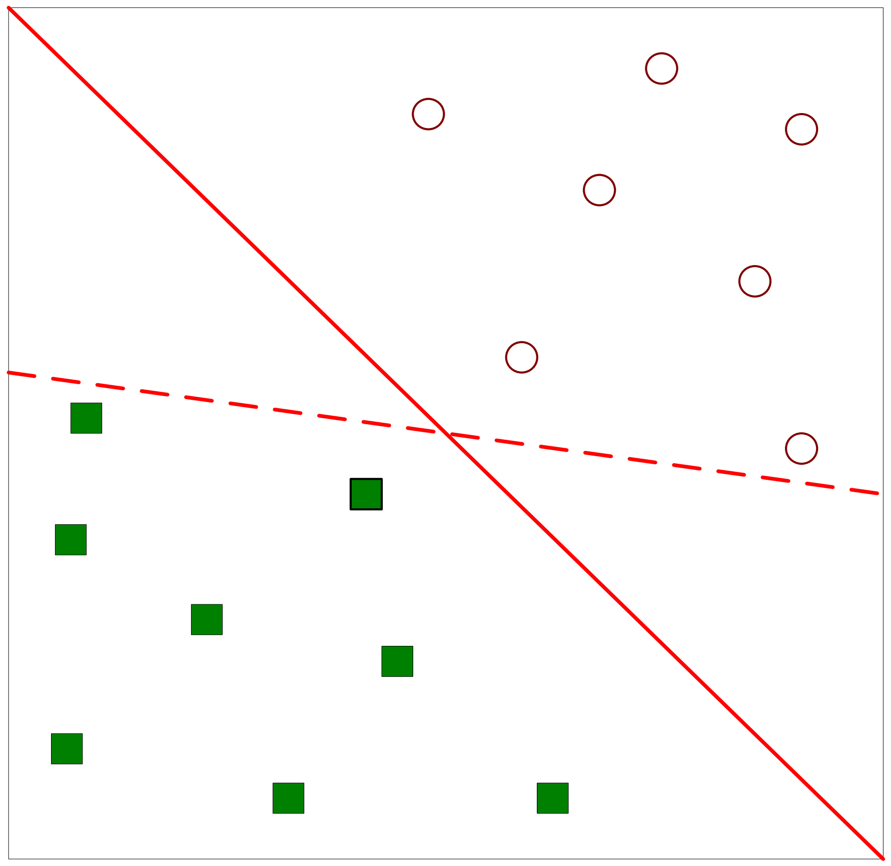
Support Vector Machines (Section 5.5, page 256)

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



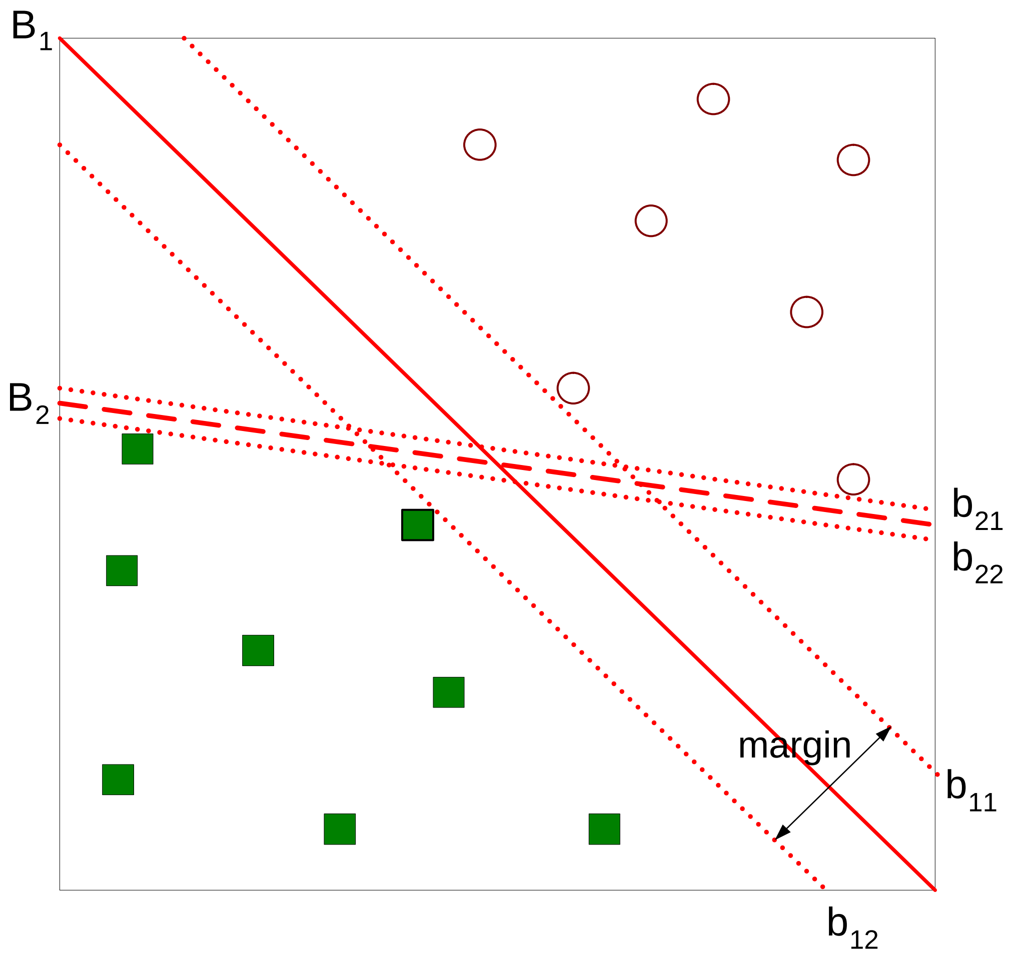
Support Vector Machines (Section 5.5, page 256)

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



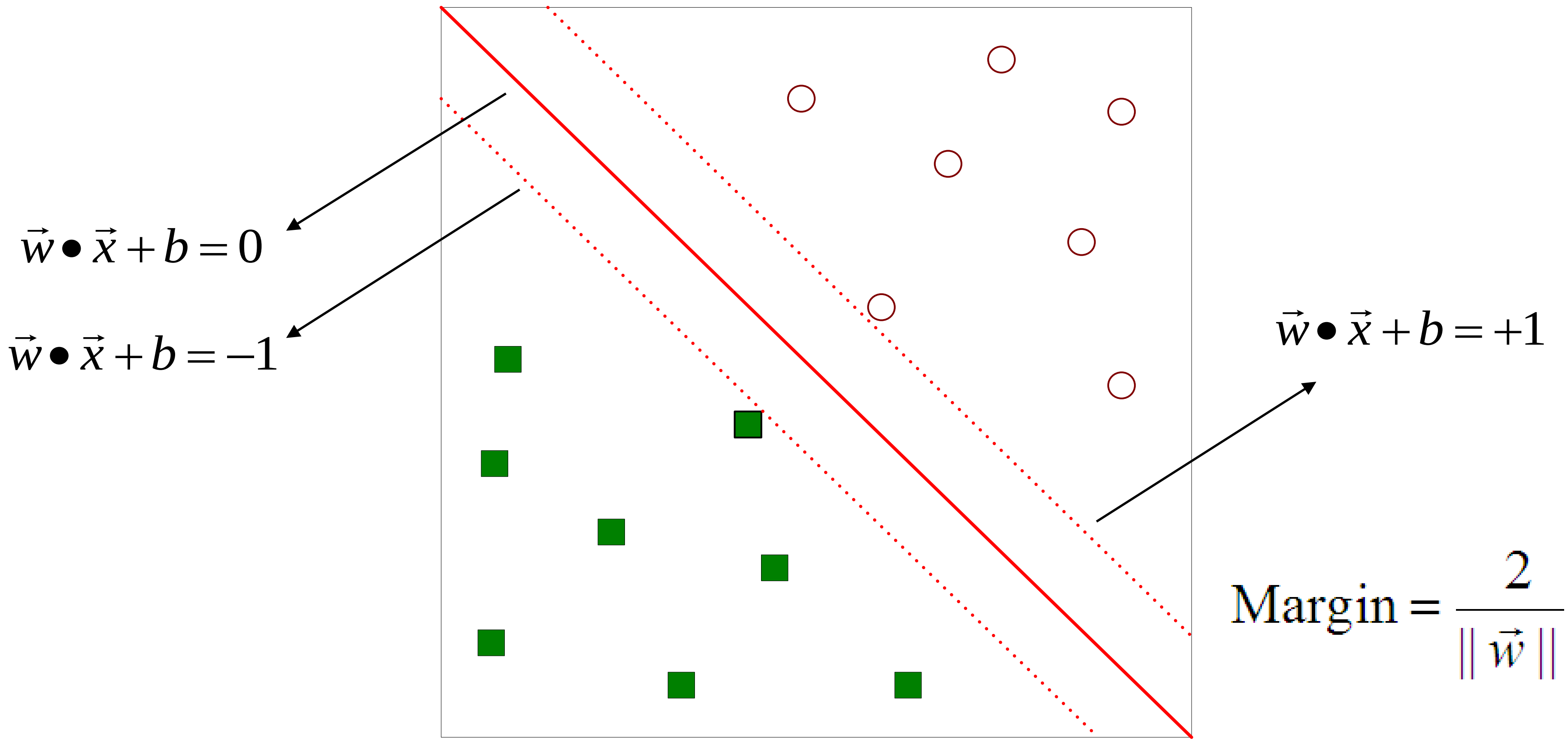
Support Vector Machines (Section 5.5, page 256)

- 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:**


$\vec{w} \bullet \vec{x} + b = 0$
$\vec{w} \bullet \vec{x} + b = -1$
$\vec{w} \bullet \vec{x} + b = +1$

$$\text{Margin} = \frac{2}{\|\vec{w}\|}$$

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

Support Vector Machines (Section 5.5, page 256)

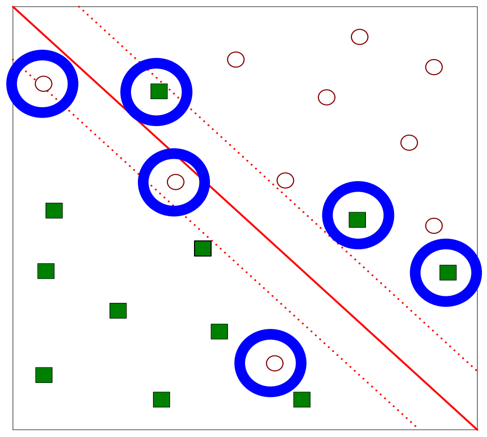
- This can be formulated as a constrained optimization problem.
- We want to maximize $\text{Margin} = \frac{2}{\|\vec{w}\|}$
- This is equivalent to minimizing $L(w) = \frac{\|\vec{w}\|^2}{2}$
- We have the following constraints

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

- 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?



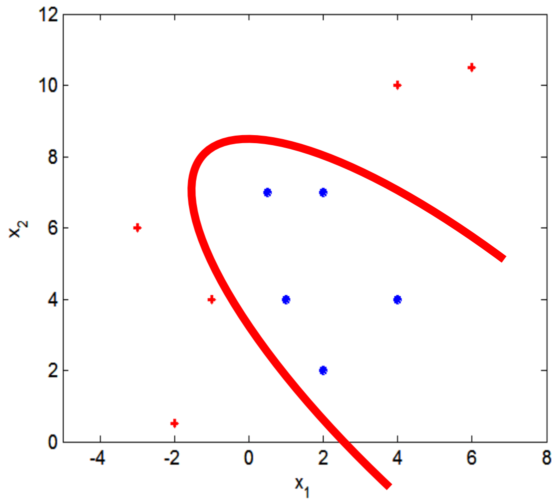
- Then we can introduce slack variables:

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

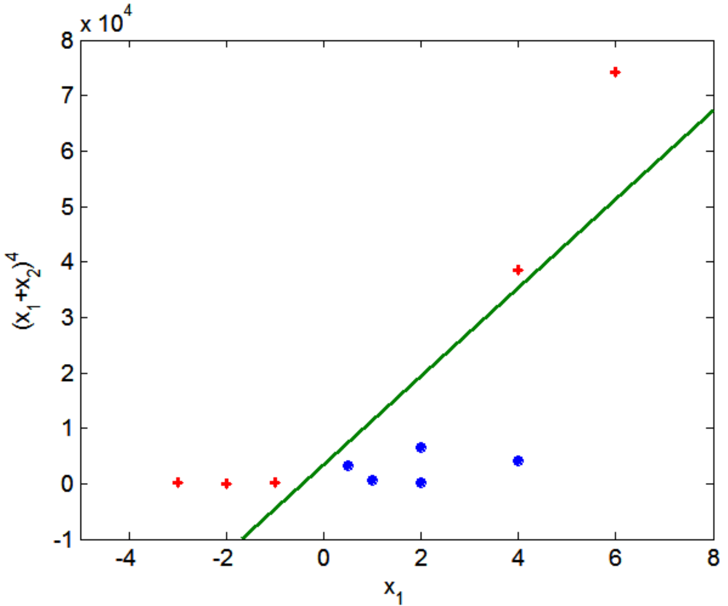
Subject to $$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x}_i + b \geq \boxed{1 - \xi_i} \\ -1 & \text{if } \vec{w} \bullet \vec{x}_i + b \leq \boxed{-1 + \xi_i} \end{cases}$$

Support Vector Machines (Section 5.5, page 256)

- What if the boundary is not linear?



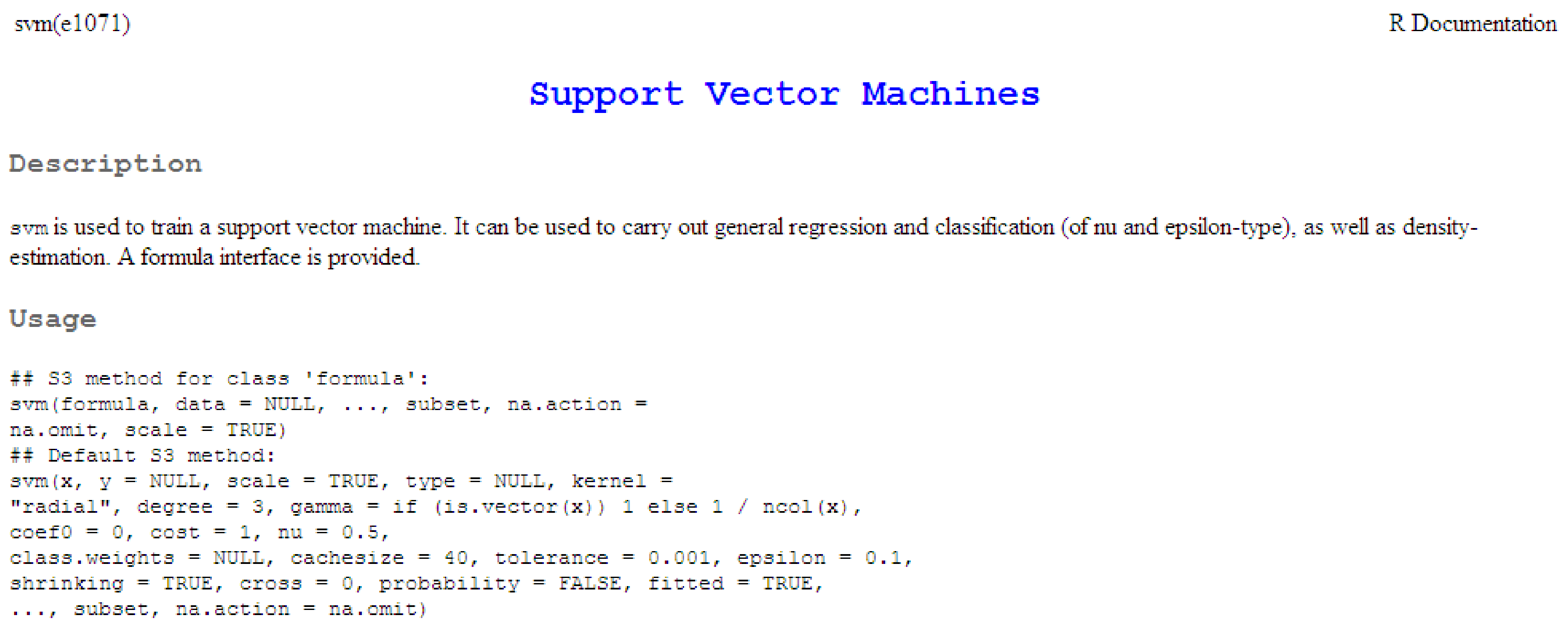
- Then we can use transformations of the variables to map into a higher dimensional space



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 #35:

Use svm() in R to fit the default svm to the last column of the sonar training data at

http://sites.google.com/site/stats202/data/sonar_train.csv

Compute the misclassification error on the training data and also on the test data at

http://sites.google.com/site/stats202/data/sonar_test.csv

In class exercise #35:

Use svm() in R to fit the default svm to the last column of the sonar training data at
http://sites.google.com/site/stats202/data/sonar_train.csv
Compute the misclassification error on the training data and also on the test data at

http://sites.google.com/site/stats202/data/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 #35:

Use svm() in R to fit the default svm to the last column of the sonar training data at
http://sites.google.com/site/stats202/data/sonar_train.csv
Compute the misclassification error on the training data and also on the test data at
http://sites.google.com/site/stats202/data/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 #36:

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 #36:

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]))
```

<u>In class exercise #36:</u>

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

<u>Solution (continued):</u>

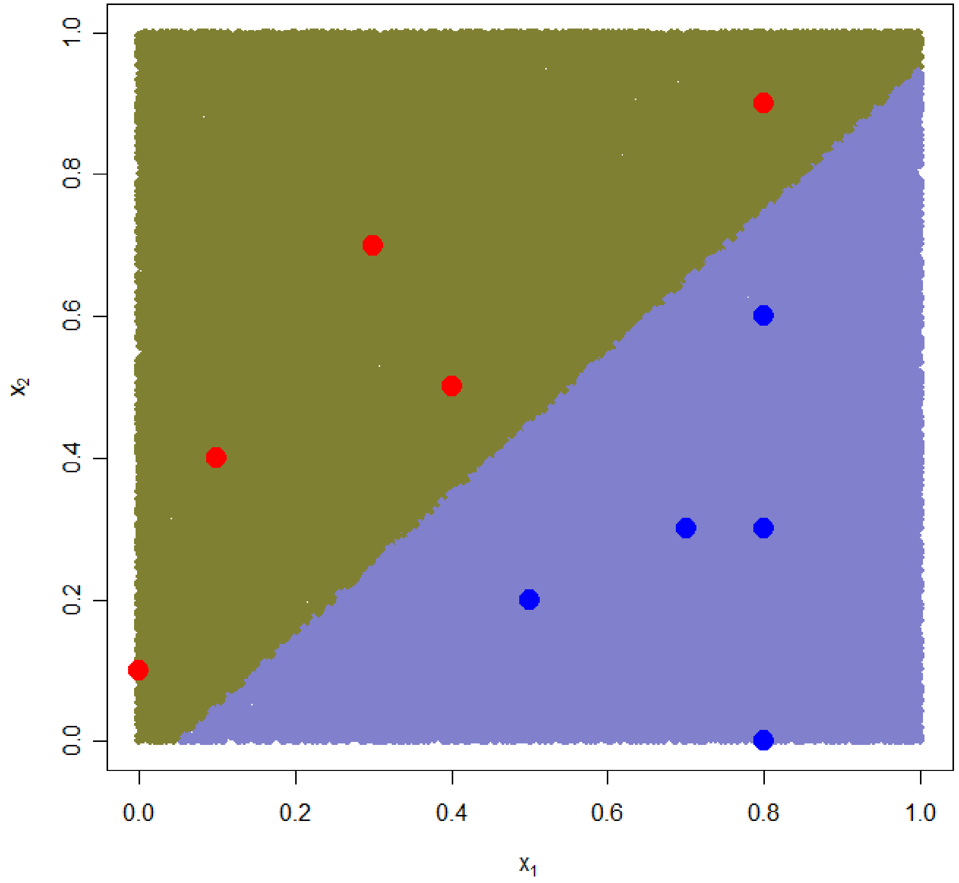
```
fit<-svm (x,y,kernel="linear",cost=100000)

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 #36:

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 #37:

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 #37:

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 #38:

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 #38:

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 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 #39:

Use randomForest() in R to fit the default Random Forest to the last column of the sonar training data at http://sites.google.com/site/stats202/data/sonar_train.csv

Compute the misclassification error for the test data at http://sites.google.com/site/stats202/data/sonar_test.csv

In class exercise #39:

Use randomForest() in R to fit the default Random Forest to the last column of the sonar training data at http://sites.google.com/site/stats202/data/sonar_train.csv **Compute the misclassification error for the test data at** http://sites.google.com/site/stats202/data/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)
```