Lin Xiao

XIAO_Lin_Solutions_hw7

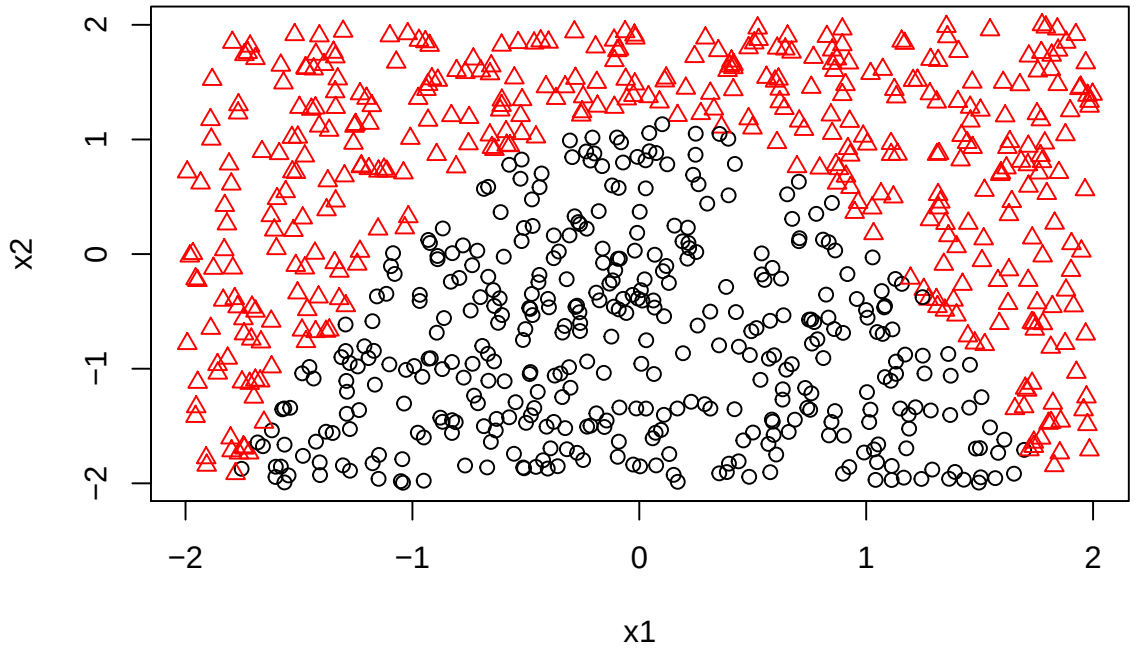
Lin XIAO

November 17, 2015

1. Here you will learn how to draw more flexible boundaries than simple linear ones, using logistic regression. Download the file "hw6prob1.Rdata" from Sakai and load it into your **R** session. Now you should have the predictor matrix x, which contains n = 800 points in p = 2 dimensions. Each point falls into either class 0 or 1. There are two scenarios for the class labels, given by y_1 and y_2.

   (a) Plot the data in x with the class labels given by y_1. (Use the option col or pch or both to distinguish between the classes.) Run logistic regression, using the glm function with family="binomial", to build a prediction rule. What is the training misclassification rate of this rule? (Hint: use the predict function; to read the relevant documentation, type ?predict.glm.)

```
load("/Users/LinXIAO/hw6prob1.rdata")

# Plot the data in x with the class labels given by y1
# Red ones are in class 1
plot(x, col = y1+1, pch = y1+1, xlab="x1",ylab="x2")
```




```
# Fit a logistic regression
fit1 <- glm(y1~x,family = "binomial")

# Predict the values(probabilities)
pred1 = predict.glm(fit1, type = "response")

# Use 0.5 as the boundary to reassign 0 or 1 values
y_new <- c()
for(i in 1:length(pred1)){
  if(pred1[i]>0.5){
    y_new[i] <- 1
```

Lin Xiao

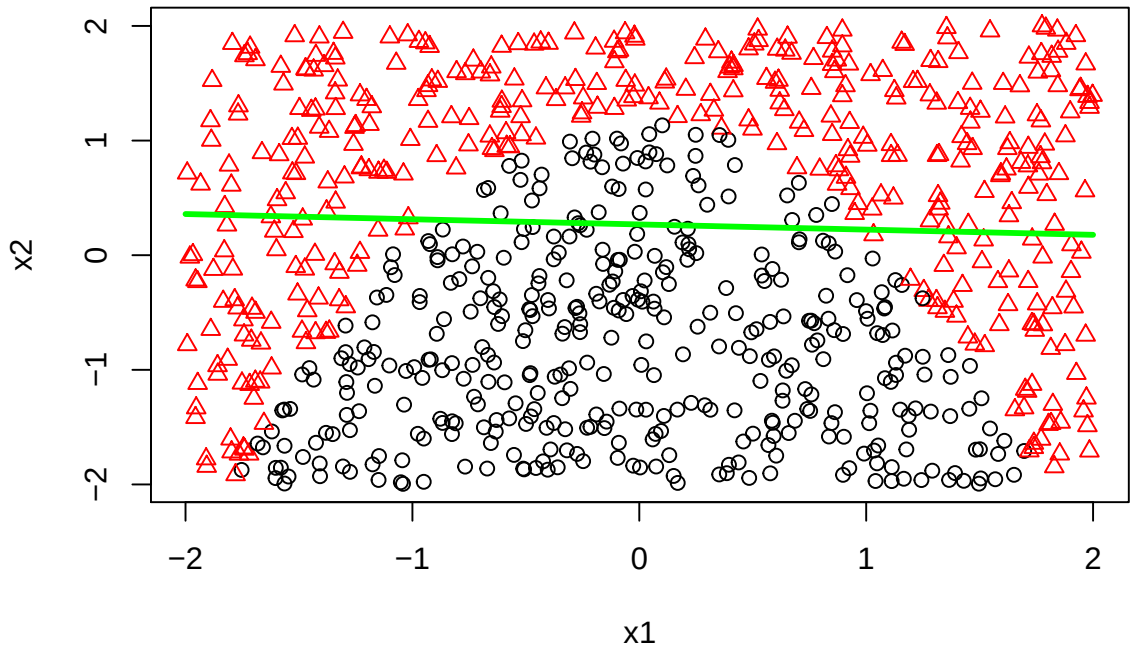
```
  }else{
    y_new[i] <- 0
  }
}

# Caluculate the training misclassification rate
sum(y_new != y1)/800
```

```
## [1] 0.235
```

The training misclassification rate of this rule is 0.235

(b) Draw the decision boundary in R^2 of the logistic regression rule from (a), on top of your plot from (a). What shape is it? Does this boundary look like it adequately separates the classes?

```
x1_bound <- seq(-2,2, length.out = 800)
x2_bound <- ( 0.5- coef(fit1)[1] - coef(fit1)[2] *x1_bound)/coef(fit1)[3]

plot(x, col = y1+1, pch = y1+1, xlab="x1",ylab="x2")
# Decision Boundary
# y1, x
lines(x1_bound, x2_bound, col = "green", lwd = 3)
```



The boundary is a line painted in green. It looks like this boundary doesn't adequately separate the classes.

(c) Run logistic regression on the predictors in x, as well as the predictor $x[,1]^2$. This is analogous to adding a quadratic term to a linear regression. To do this, define a new predictor matrix $x.quad = cbind(x, x[,1]^2)$, and run a logistic regression of y_1 on $x.quad$. What is the training misclassification rate of this rule? Why is this better than the rule from (a)?

```
# Define a new predictor matrix
x.quad <- cbind(x,x[,1]^2)

# Run a logistic regression of y1 on x.quad
fit2 <- glm(y1~x.quad,family = "binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
# Predict the values(probabilities)
pred2 = predict.glm(fit2, type = "response")

# Use 0.5 as the boundary to reassign 0 or 1 values
y_new <- c()
for(i in 1:length(pred2)){
  if(pred2[i]>0.5){
    y_new[i] <- 1
  }else{
    y_new[i] <- 0
  }
}

# Caluculate the training misclassification rate
sum(y_new != y1)/800
```

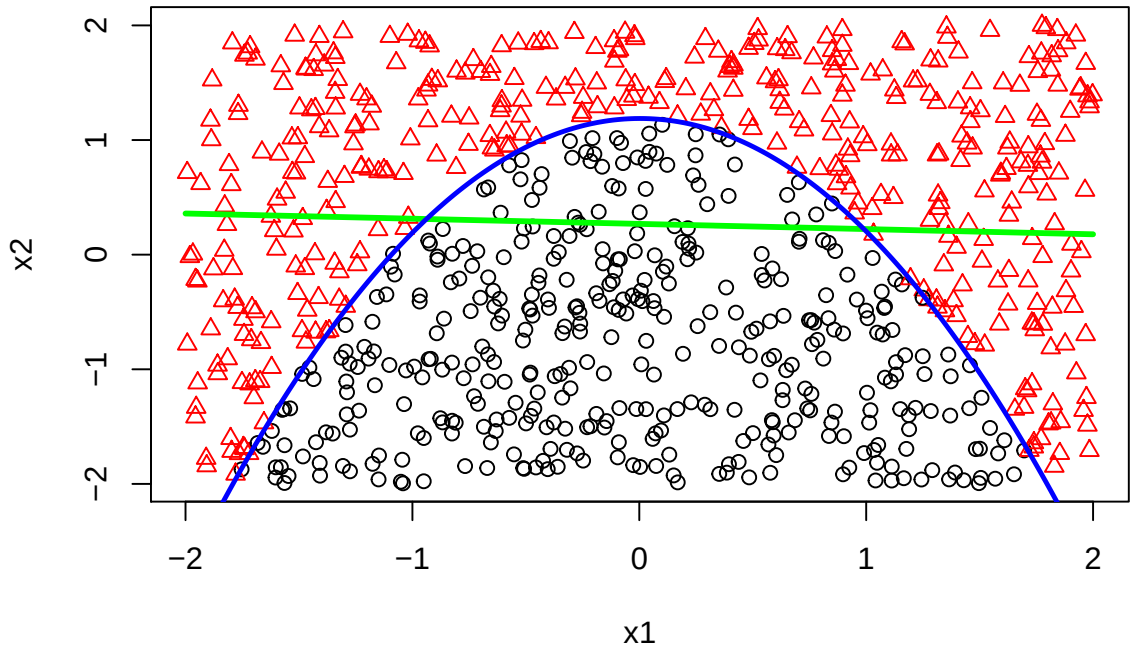
```
## [1] 0
```

The training misclassification rate of this rule is 0. It's better than the rule from (a) because we have added a quadratic term into the regression and fit the data better.

(d) In R^2, i.e., in the space $x[,1]$ versus $x[,2]$, what is the shape of the decision boundary of the logistic regression rule from (c)? Draw this decision boundary on top of a plot of the (appropriately color-coded or pch-coded) data x. What shape is it?

```
plot(x, col = y1+1, pch = y1+1, xlab="x1",ylab="x2")

x2_bound <- ( 0.5- coef(fit1)[1] - coef(fit1)[2] *x1_bound)/coef(fit1)[3]

# Decision Boundary
# y1, x
lines(x1_bound, x2_bound, col = "green", lwd = 3)

x2_bound <- (0.5- coef(fit2)[1] - coef(fit2)[2] *x1_bound -
               coef(fit2)[4]*x1_bound^2)/ coef(fit2)[3]
# Decision Boundary
# y1, cbind(x, x[,1]^2)2
lines(x1_bound, x2_bound, col = "blue", lwd = 2.5)
```

Lin Xiao



The boundary has a curve(parabola) shape, which is created by the quadratic term we introduced.

(e) Plot the data in x with the labels given by y_2. Try a running logistic regression of y_2 on x, and also on $x.quad = cbind(x, x[,1]^2)$. What are the training misclassification rates of these rules? Draw the decision boundaries of each rule on top of a plot of the data.

```
# Plot the data in x with the class labels given by y2
# Red ones are in class 1
plot(x, col = y2+1, pch = y2+1, xlab="x1",ylab="x2")

# Run a logistic regression of y2 on x
fit3 <- glm(y2~x,family = "binomial")

# Predict the values(probabilities)
pred3 = predict.glm(fit3, type = "response")

# Use 0.5 as the boundary to reassign 0 or 1 values
y_new <- c()
for(i in 1:length(pred3)){
  if(pred3[i]>0.5){
    y_new[i] <- 1
  }else{
    y_new[i] <- 0
  }
}

# Caluculate the training misclassification rate
sum(y_new != y2)/800
```

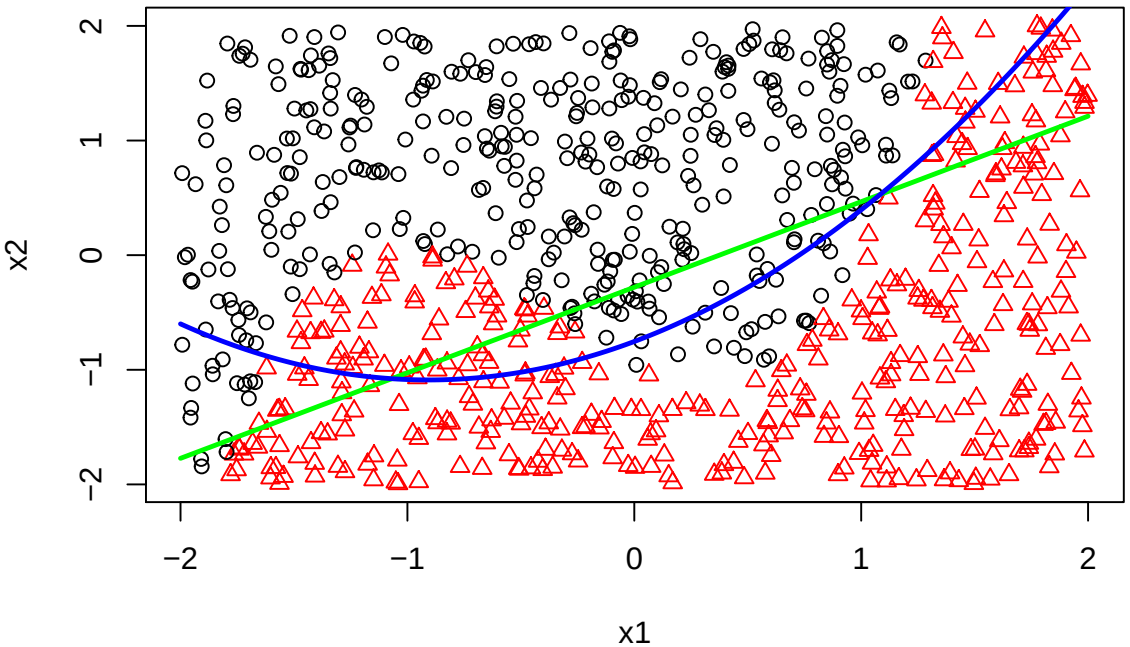
```
## [1] 0.185
```

```
# Run a logistic regression of y2 on x.quad
fit4 <- glm(y2~x.quad,family = "binomial")

# Predict the values(probabilities)
```

```
pred4 = predict.glm(fit4, type = "response")

# Use 0.5 as the boundary to reassign 0 or 1 values
y_new_2 <- c()
for(i in 1:length(pred4)){
  if(pred4[i]>0.5){
    y_new_2[i] <- 1
  }else{
    y_new_2[i] <- 0
  }
}


# Caluculate the training misclassification rate
sum(y_new_2 != y2)/800
```

```
## [1] 0.1375
```

```
x1_bound <- seq(-2,2, length.out = 800)
x2_bound <- (0.5- coef(fit3)[1] - coef(fit3)[2] *x1_bound)/coef(fit3)[3]
# Decision Boundary
# y2, x
lines(x1_bound, x2_bound, col = "green", lwd = 2.5)

x2_bound <- (0.5- coef(fit4)[1] - coef(fit4)[2] *x1_bound -
               coef(fit4)[4]*x1_bound^2)/ coef(fit4)[3]
# Decision Boundary
# y2, cbind(x, x[,1]^2)2
lines(x1_bound, x2_bound, col = "blue", lwd = 2.5)
```



The misclassfication rate of y_2 regressing on x is 0.185. The misclassfication rate of y_2 regressing on x_1, x_2, x_1^2 is 0.1375.

(f) Why are neither of the decision boundaries from (e) adequate? What additional predictors can you pass to logistic regression in order for it to do a better job of separating the classes? (Hint: draw a

curve between the classes by eye... what shape does this have?) Run a logistic regression with these additional predictors, report the training misclassification rate, and draw the new decision boundary. What shape is it?

The decision boundaries from (e) are not adequate because the misclassfication rates from (e) are both higher than 0. Since the boundary has a cubic curve, I would add a cubic term to the predictors.

```
# Add cubic term to the predictors
x.quad <- cbind(x,x[,1]^2, x[,1]^3)

# Fit Logistic regression and predict
fit5 <- glm(y2~x.quad,family = "binomial")
```

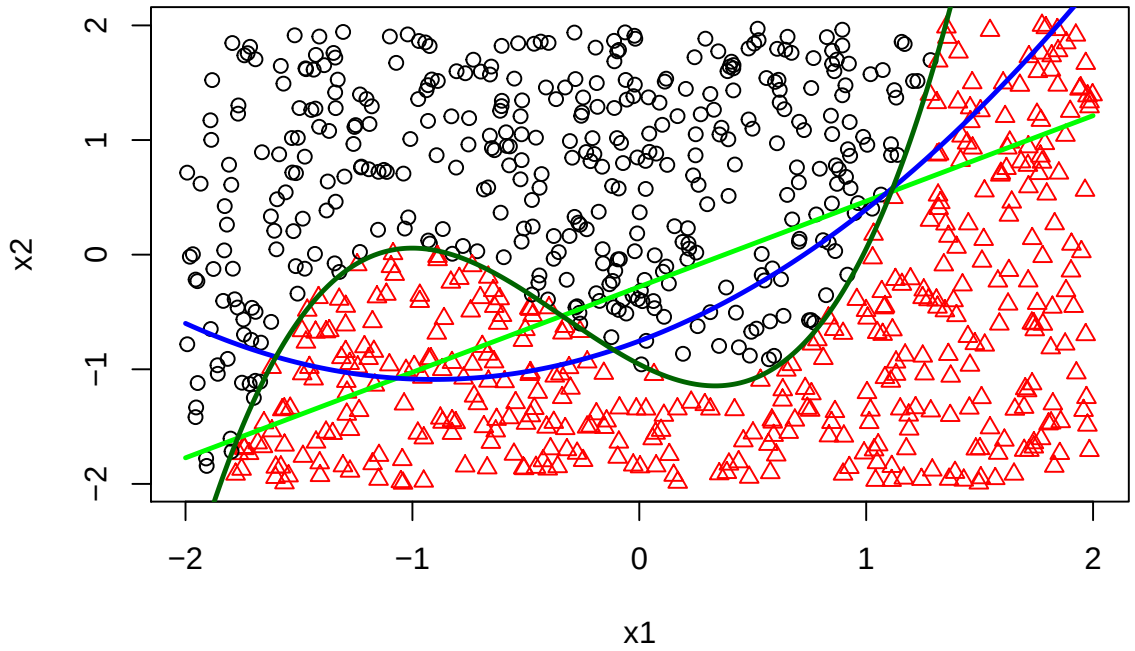
```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
pred5 = predict.glm(fit5, type = "response")

# Reassign 0 and 1 values
y_new <- c()
for(i in 1:length(pred5)){
  if(pred5[i]>0.5){
    y_new[i] <- 1
  }else{
    y_new[i] <- 0
  }
}

# Misclassification rate
sum(y_new != y2)/800
```

```
## [1] 0
```

```
plot(x, col = y2+1, pch = y2+1, xlab="x1",ylab="x2")
x2_bound <- (0.5- coef(fit3)[1] - coef(fit3)[2] *x1_bound)/coef(fit3)[3]
# Decision Boundary
# y2, x
lines(x1_bound, x2_bound, col = "green", lwd = 2.5)

x2_bound <- (0.5- coef(fit4)[1] - coef(fit4)[2] *x1_bound -
               coef(fit4)[4]*x1_bound^2)/ coef(fit4)[3]
# Decision Boundary
# y2, cbind(x, x[,1]^2)2
lines(x1_bound, x2_bound, col = "blue", lwd = 2.5)


x2_bound <- (0.5- coef(fit5)[1] - coef(fit5)[2] *x1_bound -
               coef(fit5)[4]*x1_bound^2 -
               coef(fit5)[5]*x1_bound^3)/ coef(fit5)[3]
# Draw the curve
# y2, cbind(x,x[,1]^2, x[,1]^3)
lines(x1_bound, x2_bound, col = "darkgreen", lwd = 2.5)
```



Its shape is cubic's curve. And the misclassfication rate of y_2 regressing on x_1, x_2, x_1^2, x_1^3 is 0.

(g) If adding polynomial terms seems to improve the training misclassification rate of the logistic regression rule, why don't we generally just keep including polynomial terms of higher and higher order? How could we choose how many polynomial terms to include in a principled manner?

If we keep adding polynomial terms of especially higher and higher order, the model becomes more complex and better fit but it will tend to become overfit and generally have poor predictive performance, as it can exaggerate minor fluctuations in the data.

We can use cross validation to fix the issue, by calculating MSE for each time we add a polynomial term. If the MSE barely changes when you add more complicated terms into the predictors, we can choose the most simple regression model as what we need.

A loose but general rule is that if $K \leq 3$ classes are lined up, polynomial terms up to degree $K-1$ might be needed to resolve the problem of masking.

2. In class, we learned three measures for impurity used by classification trees. We are given training data (x_i, y_i),i = 1,. . . ,n, where x_i are the feature vectors and $y_i \in 1, ..., K$ are the class labels. For a region R, remember that the proportion of points in R that are of class k is simply

$$\widehat{p}_k = \frac{1}{n_k} \sum_{x_i \in R} 1\{y_i = k\}$$

, where n_j is the total number of points in R. Let

$$k^* = \underset{k}{argmax} \widehat{p}_k$$

the most common class among points in R. The three measures of impurity for R are:

$$\text{Misclassification error: } 1 - \widehat{p}_{k^*}$$
$$\text{Gini index: } \sum_k \widehat{p}_k (1 - \widehat{p}_k)$$
$$\text{Cross entropy: } -\sum_k \widehat{p}_k \log \widehat{p}_k$$

(a) Suppose that there are only two classes, K = 2. Let p denote the proportion of points in R that are in the first class, i.e., $p = \widehat{p}_1$. Write out each of the above impurity measures as a function of p (and only p).

Lin Xiao

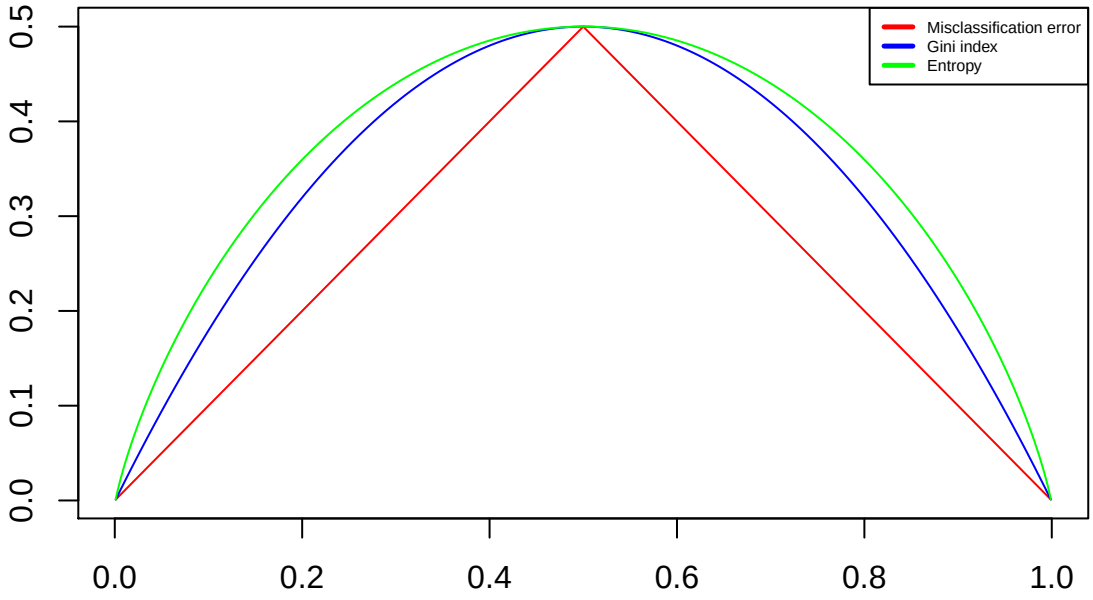
Misclassification error: $1 - \widehat{p}_{argmax_k \widehat{p}_k} = 1 - max(p, 1-p)$

Gini index: $\sum_k \widehat{p}_k(1-\widehat{p}_k) = 2p(1-p)$

Cross entropy: $-p\log p - (1-p)\log(1-p)$

(b) Plot the expressions for misclassification error, Gini index, and cross- entropy that you found in (a) as functions of p. When you plot cross- entropy, you can scale it by a constant so that it is equal to 1/2 when p = 1/2. Are the curves similar?

```
p <- seq(0.001, 0.999, by = 0.001)

y1 <- c()
for(i in 1:length(p)){
  y1[i] <- 1 - max(1-p[i], p[i])
}

y2 <- 2*p*(1-p)
y3 <- (-p*log(p)-(1-p)*log(1-p))/(2*log(2))

par(ann=FALSE)
plot(p, y1, col = "red", type = "l")
par(new = T)
plot(p, y2, col = "blue", type = "l", axes = F)
par(new = T)
plot(p, y3, col = "green", type = "l", axes = F)
legend("topright", c("Misclassification error","Gini index","Entropy"),
       lty=c(1,1,1), lwd=c(2.5,2.5,2.5),
       col=c("red", "blue", "green"), cex=0.5)
```



Cross Entropy has been scaled to pass through (0.5,0.5). The curves are similar but cross entropy and Gini index are differentiable and more senstive to changesin the node probabilities.