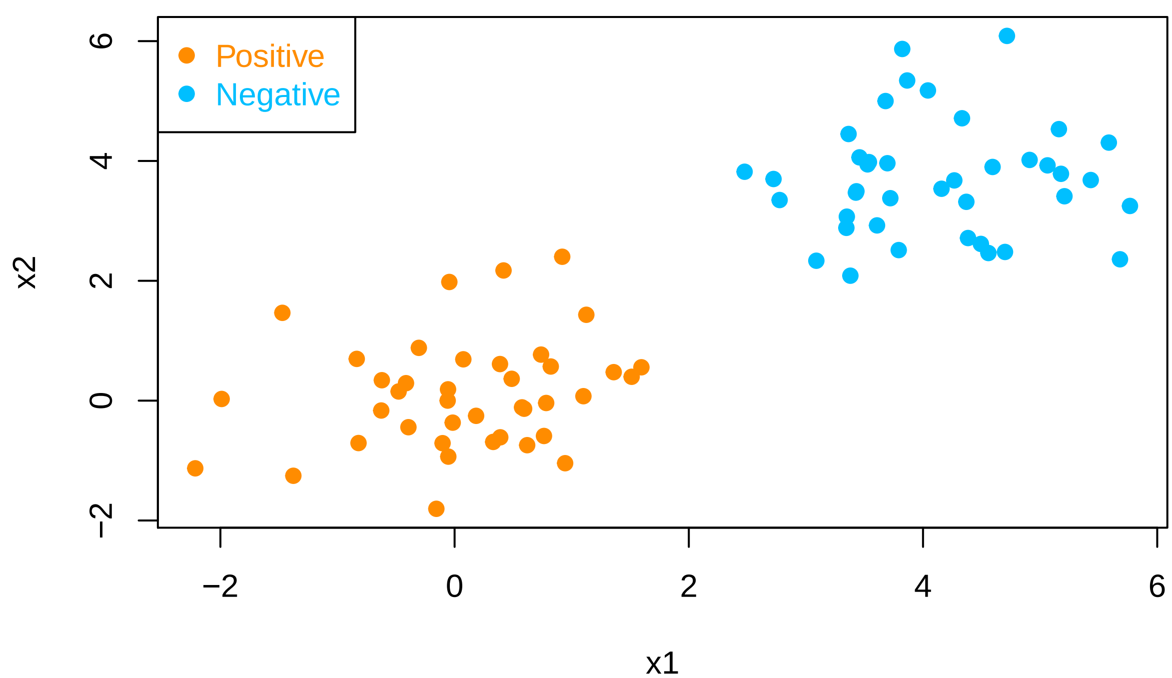
Question 1: Linearly Separable SVM using Quadratic Programming

```
rm(list=ls());

set.seed(1); n <- 40; p <- 2
xpos <- matrix(rnorm(n*p, mean=0, sd=1), n, p)
xneg <- matrix(rnorm(n*p, mean=4, sd=1), n, p)
x <- rbind(xpos, xneg)
y <- matrix(c(rep(1, n), rep(-1, n)))

plot(x,col=ifelse(y>0,"darkorange", "deepskyblue"), pch = 19, xlab = "x1", ylab = "x2")
legend("topleft", c("Positive","Negative"),
col=c("darkorange", "deepskyblue"), pch=c(19, 19), text.col=c("darkorange", "deepskyblue"))
```




```
library(quadprog)

eps <- 5e-4

# build the system matrices
Q <- sapply(1:(2*n), function(i) y[i]*t(x)[,i])
D <- t(Q)%*%Q
d <- matrix(1, nrow=2*n)
b0 <- rbind( matrix(0, nrow=1, ncol=1) , matrix(0, nrow=2*n, ncol=1) )
```

```r
A <- t(rbind(matrix(y, nrow=1, ncol=2*n), diag(nrow=2*n)))

# call the QP solver:
#sol <- solve.QP(D+eps*diag(2*n), d, A, b0, meq=1, factorized=TRUE)
sol <- solve.QP(D+eps*diag(2*n), d, A, b0, meq=1, factorized=FALSE)
qpsol <- matrix(sol$solution, nrow=2*n)

beta <- rep(0, 2);

beta[1] <- sum(qpsol*y*x[,1])
beta[2] <- sum(qpsol*y*x[,2])

minOfPos <- xpos[1,];
min <- t(xpos[1,]%*%beta)
for(i in c(1:n))
{
  if(t(xpos[i,])%*%beta <= min)
  {
    minOfPos <- xpos[i,];
    min <- t(xpos[i,])%*%beta;
  }
}

maxOfNeg <- xneg[1,];
max <- t(xneg[1,]%*%beta)
for(i in c(1:n))
{
  if(t(xneg[i,])%*%beta >= max)
  {
    maxOfNeg <- xneg[i,];
    max <- t(xneg[i,])%*%beta;
  }
}


beta_0 <- -(min+max)/2

beta
```

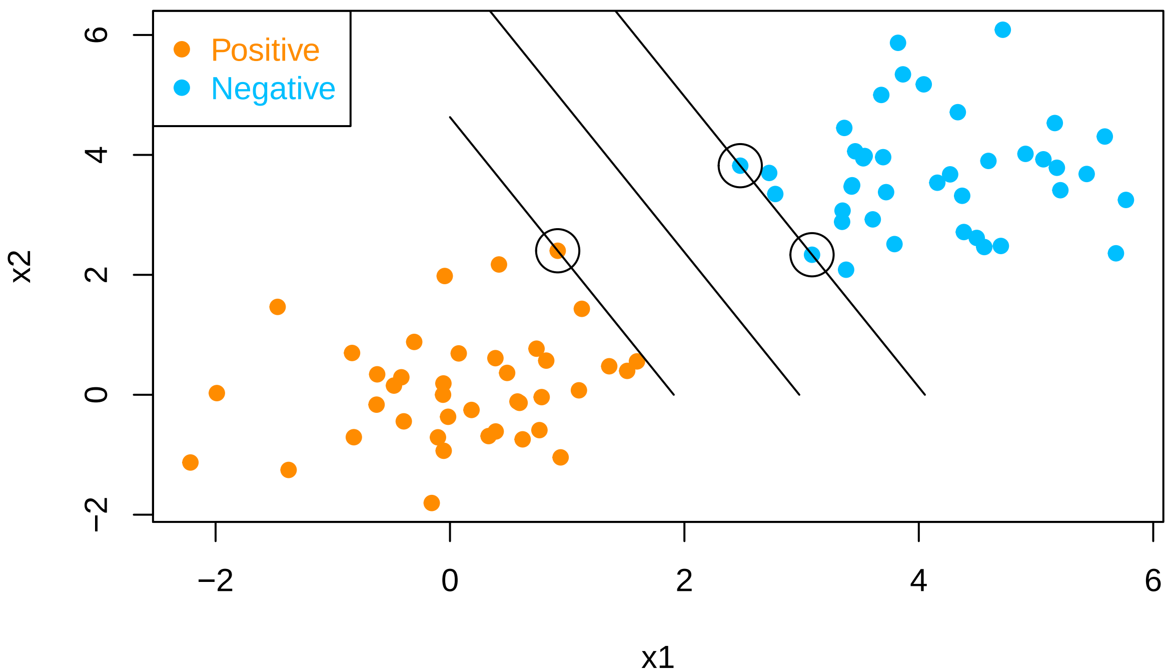
```
## [1] -0.9332568 -0.3848943
```

```r
beta_0
```

```
##          [,1]
## [1,] 2.781813
```

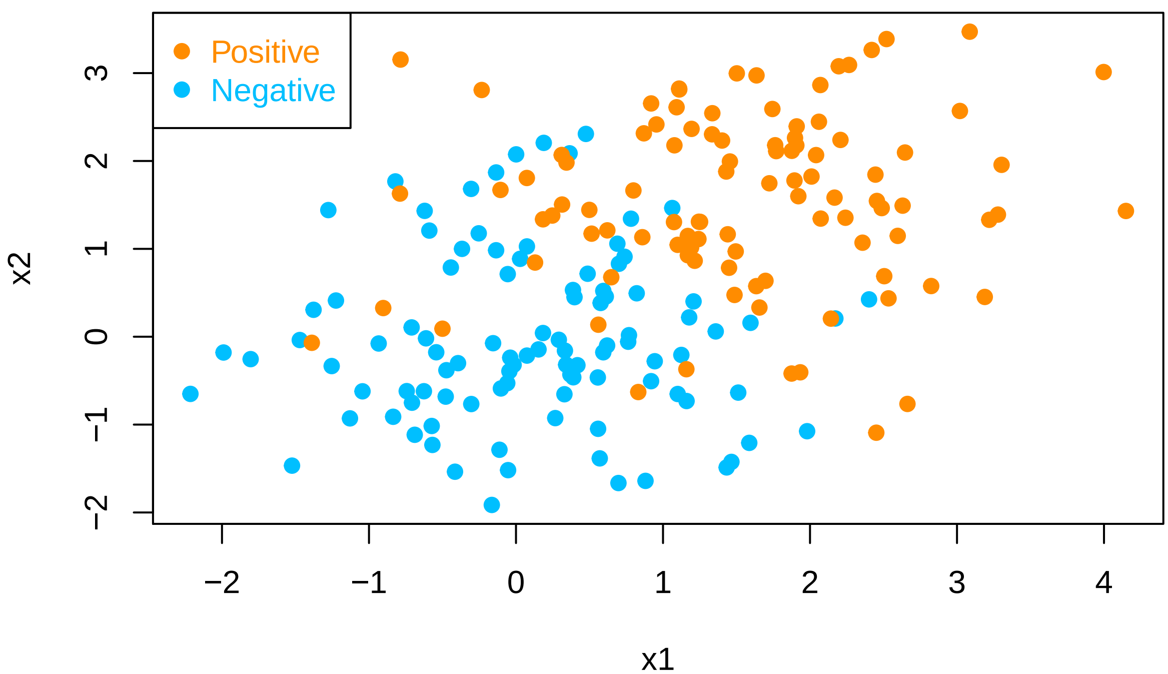
```r
plot(x,col=ifelse(y>0,"darkorange", "deepskyblue"), pch = 19, xlab = "x1", ylab = "x2")
legend("topleft", c("Positive","Negative"),
col=c("darkorange", "deepskyblue"), pch=c(19, 19), text.col=c("darkorange", "deepskyblue"))

svm_x <- c(0, -beta_0/beta[1])
svm_y <- c(-beta_0/beta[2], 0)
```

```r
lines(x=svm_x, y=svm_y)

beta_0 <- -t(minOfPos)%*%beta

svm_x <- c(0, -beta_0/beta[1])
svm_y <- c(-beta_0/beta[2], 0)

lines(x=svm_x, y=svm_y)

beta_0 <- -t(maxOfNeg)%*%beta

svm_x <- c(0, -beta_0/beta[1])
svm_y <- c(-beta_0/beta[2], 0)

lines(x=svm_x, y=svm_y)

for(i in c(1:(2*n)))
{
  if(abs(sol$solution[i])>eps)
  {
    points(x = x[i,1], y=x[i,2], col = "black", cex=3)
  }
}
```



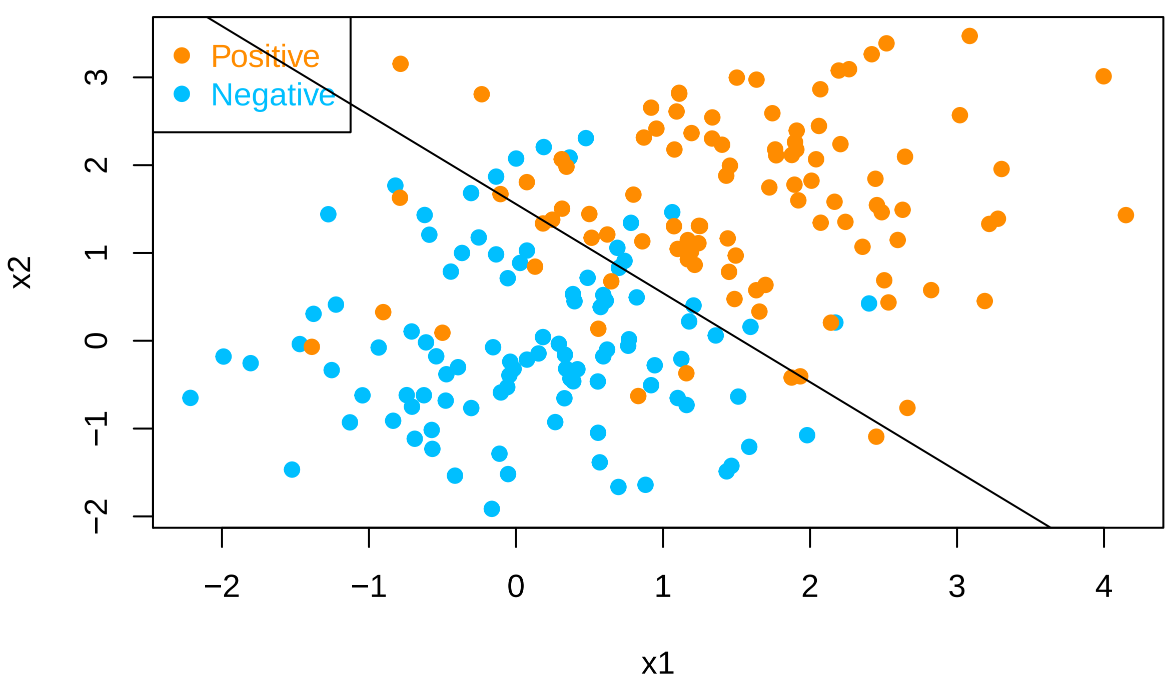
Question 2 Linearly Non-seperable SVM using Penalized Loss

```r
rm(list=ls());

set.seed(1)
  n = 100 # number of data points for each class
  p = 2 # dimension

  # Generate the positive and negative examples
  xpos <- matrix(rnorm(n*p,mean=0,sd=1),n,p)
  xneg <- matrix(rnorm(n*p,mean=1.5,sd=1),n,p)
  x <- rbind(xpos,xneg)
  y <- c(rep(-1, n), rep(1, n))

  plot(x,col=ifelse(y>0,"darkorange", "deepskyblue"), pch = 19, xlab = "x1", ylab = "x2")
  legend("topleft", c("Positive","Negative"), col=c("darkorange", "deepskyblue"),
         pch=c(19, 19), text.col=c("darkorange", "deepskyblue"))
```



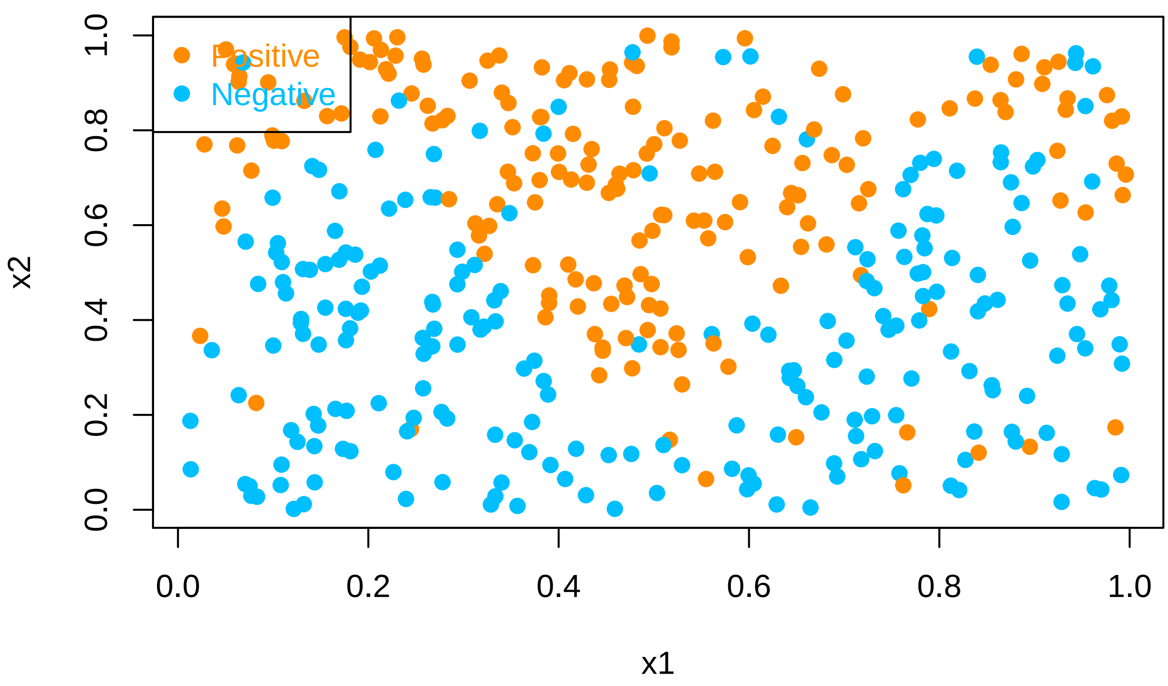

```r
f <- function(beta_c, y, x, l)
{
  beta_0 <- beta_c[1];
  beta <- c(beta_c[2], beta_c[3]);
  sum <- 0
  for(i in c(1: nrow(x)))
```

```
  {
    sum = sum + log(1+exp(-y[i]*(beta_0 + t(x[i,])%*%beta)))
  }
  sum = sum + l * sum((beta*beta))
  return <- sum
}

sol <- optim(rep(0,3), f, y=y, x=x, l=1, method='BFGS');

beta_0 <- sol$par[1]
beta <- c(sol$par[2], sol$par[3])

svm_x <- c(-5, 5)
svm_y <- c((5*beta[1]-beta_0),(-5*beta[1]-beta_0))/beta[2]

plot(x,col=ifelse(y>0,"darkorange", "deepskyblue"), pch = 19, xlab = "x1", ylab = "x2")
legend("topleft", c("Positive","Negative"), col=c("darkorange", "deepskyblue"),
         pch=c(19, 19), text.col=c("darkorange", "deepskyblue"))
lines(x=svm_x, y=svm_y)
```



Question 3: Nonlinear and Non-seperable SVM using Penalized Loss

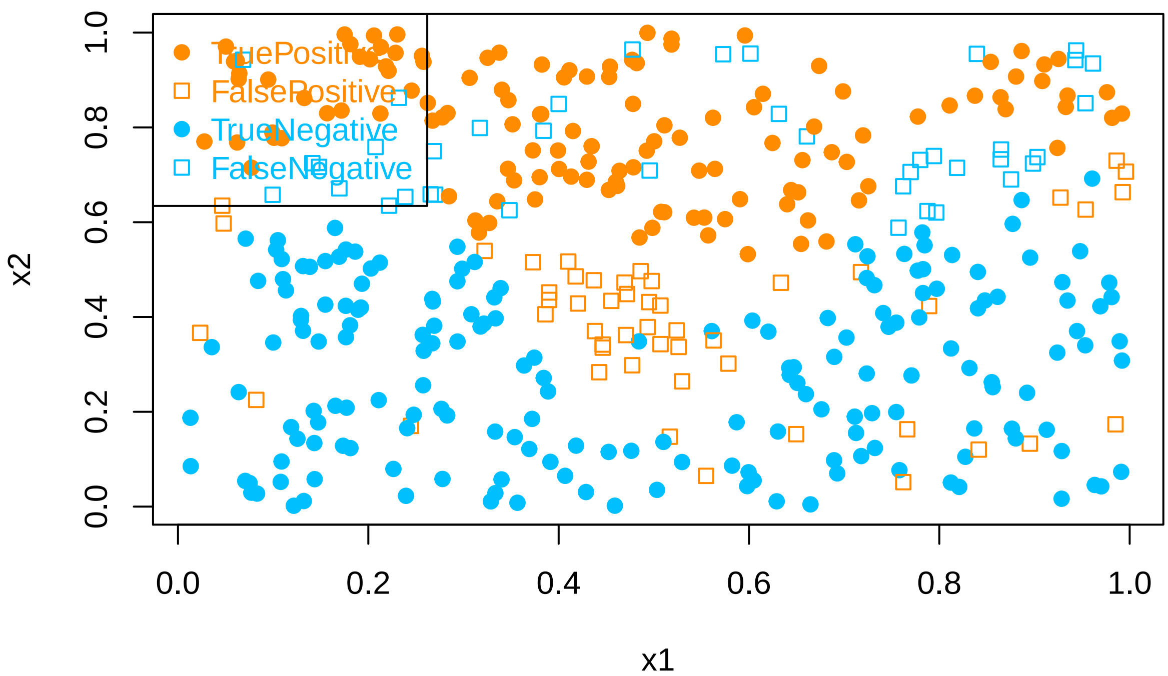
```
rm(list=ls());
set.seed(1)
n = 400
p = 2 # dimension

# Generate the positive and negative examples
x <- matrix(runif(n*p), n, p)
side <- (x[, 2] > 0.5 + 0.3*sin(3*pi*x[, 1]))
y <- sample(c(1, -1), n, TRUE, c(0.9, 0.1))*(side == 1) + sample(c(1, -1), n, TRUE, c(0.1, 0.9))*(side

plot(x,col=ifelse(y>0,"darkorange", "deepskyblue"), pch = 19, xlab = "x1", ylab = "x2")
legend("topleft", c("Positive","Negative"),
       col=c("darkorange", "deepskyblue"), pch=c(19, 19), text.col=c("darkorange", "deepskyblue"))
```




```
K <- matrix(rep(0, n*n), n, n)

for(i in c(1:n))
{
  for(j in c(i:n))
  {
    k <- exp(-sum((x[i,]-x[j,])*(x[i,]-x[j,]))/2)
```

```r
    K[i,j] <- k
    K[j,i] <- k
  }
}



f <- function(beta, y, K, lambda)
{
  sum <- 0;
  for(i in c(1:nrow(K)))
  {
    sum = sum + log(1+exp(-y[i]*(t(K[i,])%*%beta)))
  }
  sum = sum+ lambda*t(beta)%*%K%*%beta;
  return <- sum;
}

sol <- optim(rep(0,n), f, y=y, K=K, lambda=0.1, method='BFGS')
```

```r
#sol

beta <- sol$par

result <- K%*%beta

predictions <- result/abs(result)

error <- sum(predictions==y)/n

plot(x,col=ifelse(y>0,"darkorange", "deepskyblue"), pch = ifelse(predictions==y, 19, 0), xlab = "x1", y
legend("topleft", c("TruePositive","FalsePositive", "TrueNegative", "FalseNegative"),
       col=c("darkorange", "darkorange", "deepskyblue", "deepskyblue"), pch=c(19, 0, 19, 0), text.col=c
```

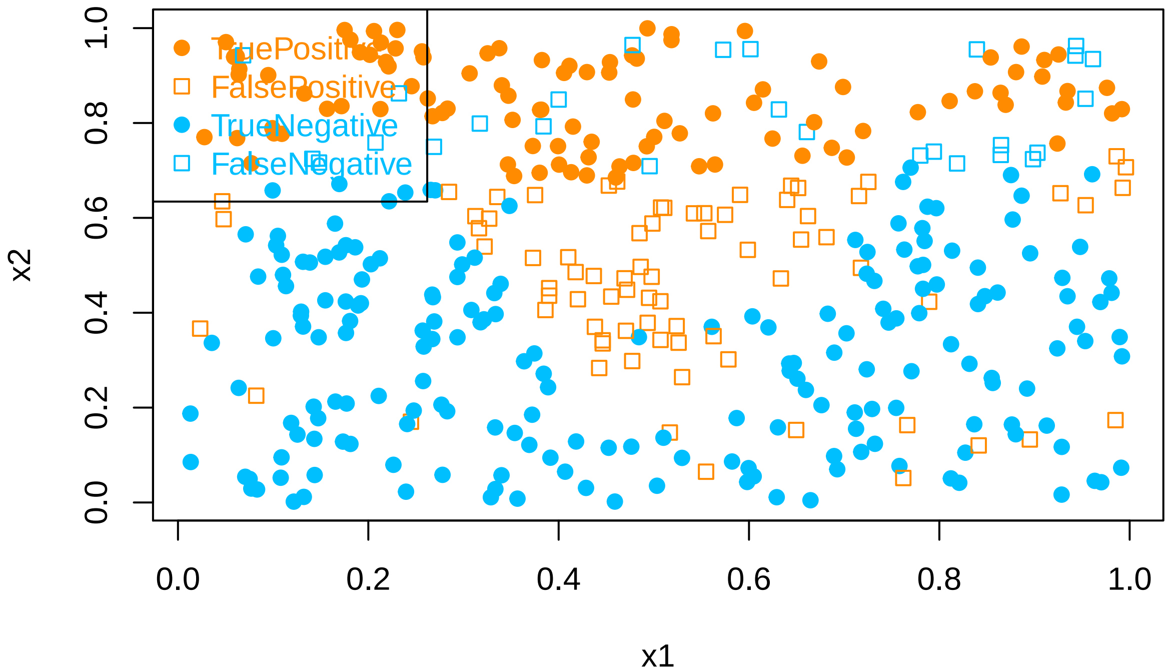



```r
cat("Error: ", error)
```

```
## Error:  0.775
```

```r
sol <- optim(rep(0,n), f, y=y, K=K, lambda=10, method='BFGS')
```

```r
#sol

beta <- sol$par

result <- K%*%beta

predictions <- result/abs(result)

error <- sum(predictions==y)/n

plot(x,col=ifelse(y>0,"darkorange", "deepskyblue"), pch = ifelse(predictions==y, 19, 0), xlab = "x1", yl
legend("topleft", c("TruePositive","FalsePositive", "TrueNegative", "FalseNegative"),
       col=c("darkorange", "darkorange", "deepskyblue", "deepskyblue"), pch=c(19, 0, 19, 0), text.col=c
```




```
cat("Error: ", error)
```

```
## Error:  0.7425
```