STAT 716 - Class 11 2016-11-26

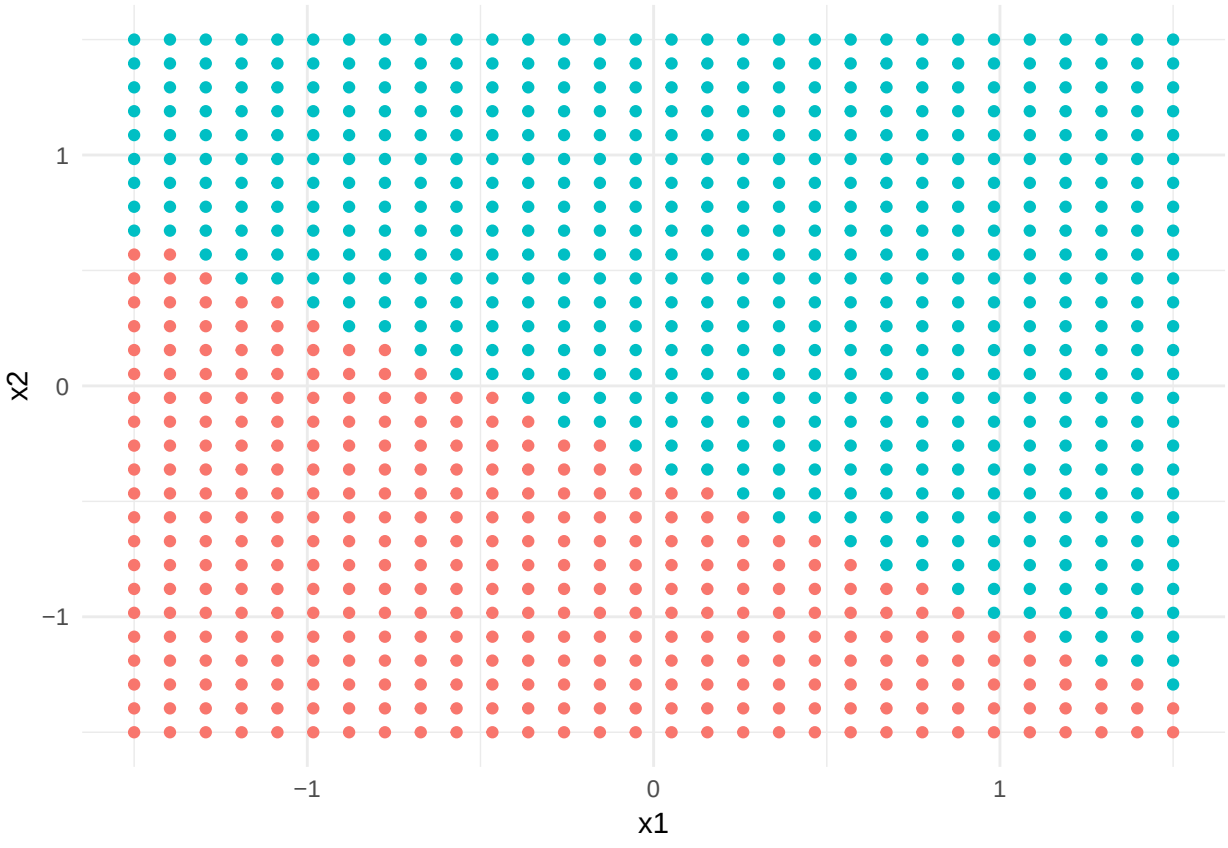
Vitaly Druker

Support Vector Machines

Maximal Margin Classifier

Setting up the problem

```
hyper_plane <- function(x1, x2){
  1 + 2*x1 + 3*x2 > 0
}

xs <- seq(-1.5, 1.5, length.out = 30)

full_data <- expand.grid(x1  = xs,
  x2 = xs) %>%
  mutate(val = hyper_plane(x1, x2))

full_data %>%
  ggplot(aes(x = x1, y = x2, color = val)) +
  geom_point() +
  theme(legend.position = "none")
```

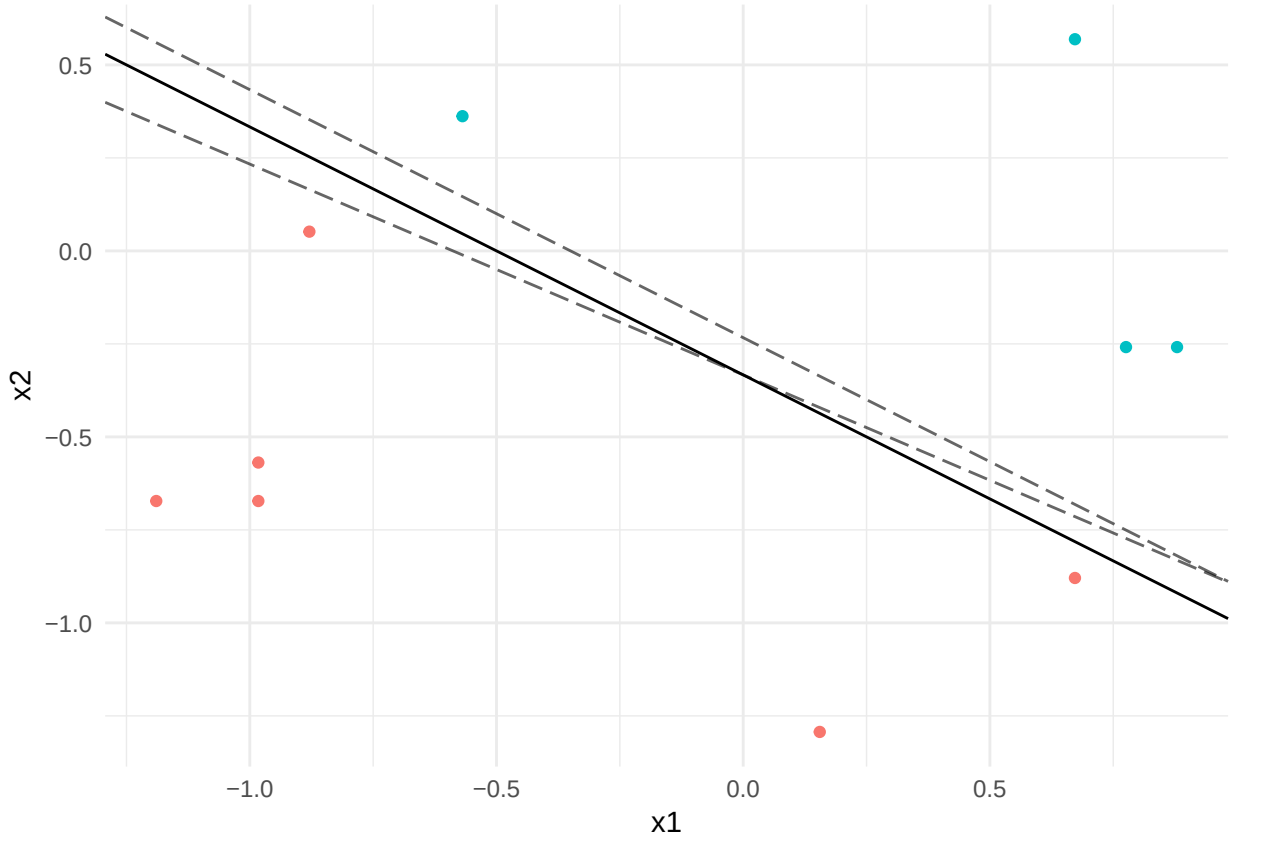


Now how does this work with real data?

The hyperplane is divided by a line

$$X_2 = -\frac{1}{3} + -\frac{2}{3}X_1$$

```
set.seed(10)
full_data %>%
  sample_n(10) %>%
  ggplot(aes(x = x1, y = x2, color = val)) +
  geom_point() +
  theme(legend.position = "none") +
  geom_abline(slope = -2/3, intercept = -1/3) +
  geom_abline(slope = -2/3+.1, intercept = -1/3, linetype = "longdash", alpha = 0.6) +
  geom_abline(slope = -2/3, intercept = -1/3+.1, linetype = "longdash", alpha = 0.6)
```



The solid line shows the true boundary while the dotted lines show possible bounderies.

This hyper plane classifier can also be written as:

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

for multiple predictors. If this predictor is > 0 then it is one class, if it is < 0 then it is a different class.

The Maximal Margin Classifier

The best choice we can make is the the maximal margin hyperplane such that the sum of distances for each point is at a maximum.

The lines that make up the distance from the closest points are called the support vectors because the hyperplane *only* depends on them. It could be just a few points of 100's

The classifier can be seen as an optimization to maximize M given the following statement must be true:

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

where y_i is either -1 or 1 depending on the class. The equation in parentheseis is either negative or positive given the side of the hyper plane it lies on.

An additional restriction is added such that:

$$\sum_{j=1}^{p} \beta_j^2 = 1$$

This does not mess up the overall hyperplane because everything is just scaled. It allows for easier calculation of distance from the hyperplane:

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip})$$

Which means M is a physical property (the margin)

Now what happenes if there is no hyperplane? Or what if the hyperplane overfits the training data?

Support Vector Classifier

Add a 'cheat' term.

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

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

Where C is a tuning parameter that represents the amount of slack available to the classifier.If ϵ_i is 0 then it is on the correct side of the Margin. If $0 < \epsilon_i \leq 1$ then it has violated the margin but is on the correct side of the hyperplane. and if $\epsilon_i > 1$ then is on the wrong side of the hyperplane.

Only observations within margin change anything.

Support Vector Machines

We can expand this to non-linear boundaries using non linear (as we saw in the previous chapter) transformations of X

However we can also use kernals

Eg polynomial kernal and radial kernal which modify the way that the distance between two points is calculated. Standard:

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

Polynomials

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

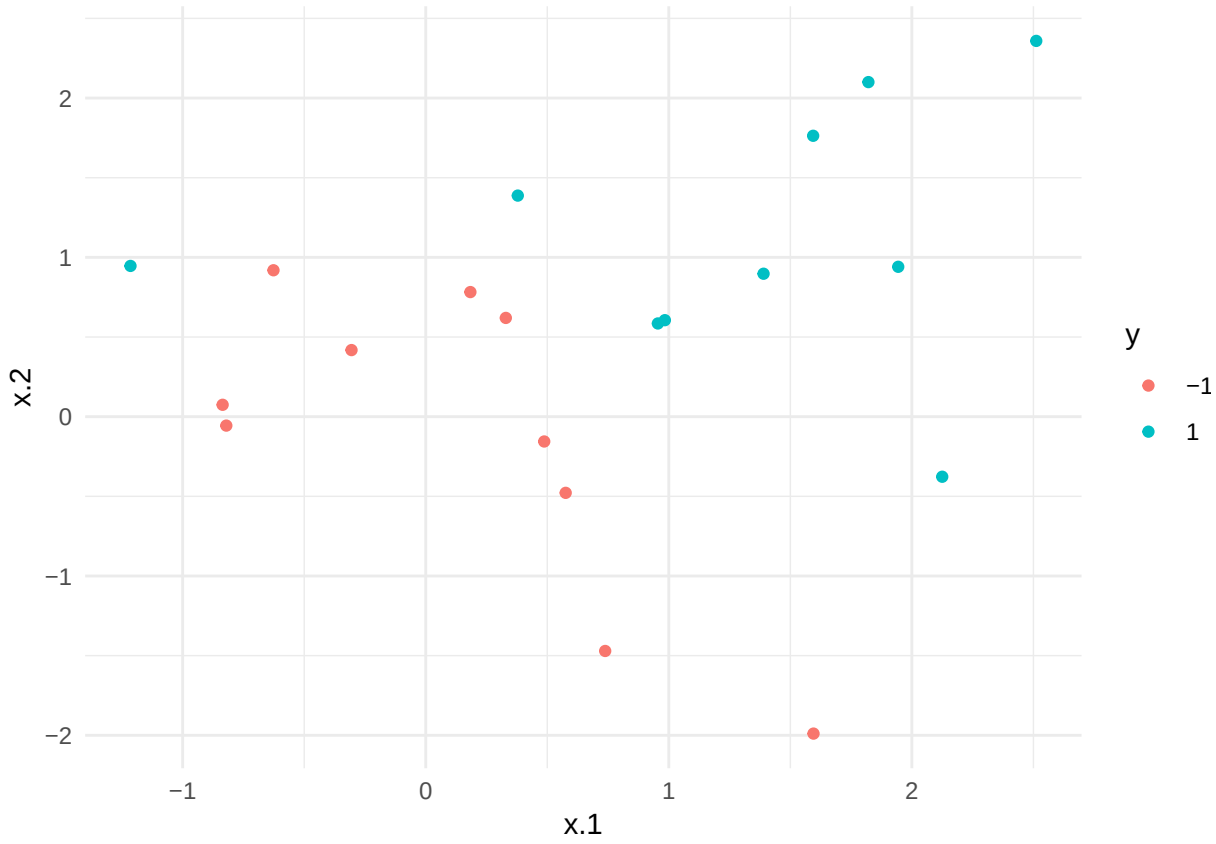
Radial

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

SVMs are actually thought to be similar to logistic regession with penality term (e.g. ridge regression)

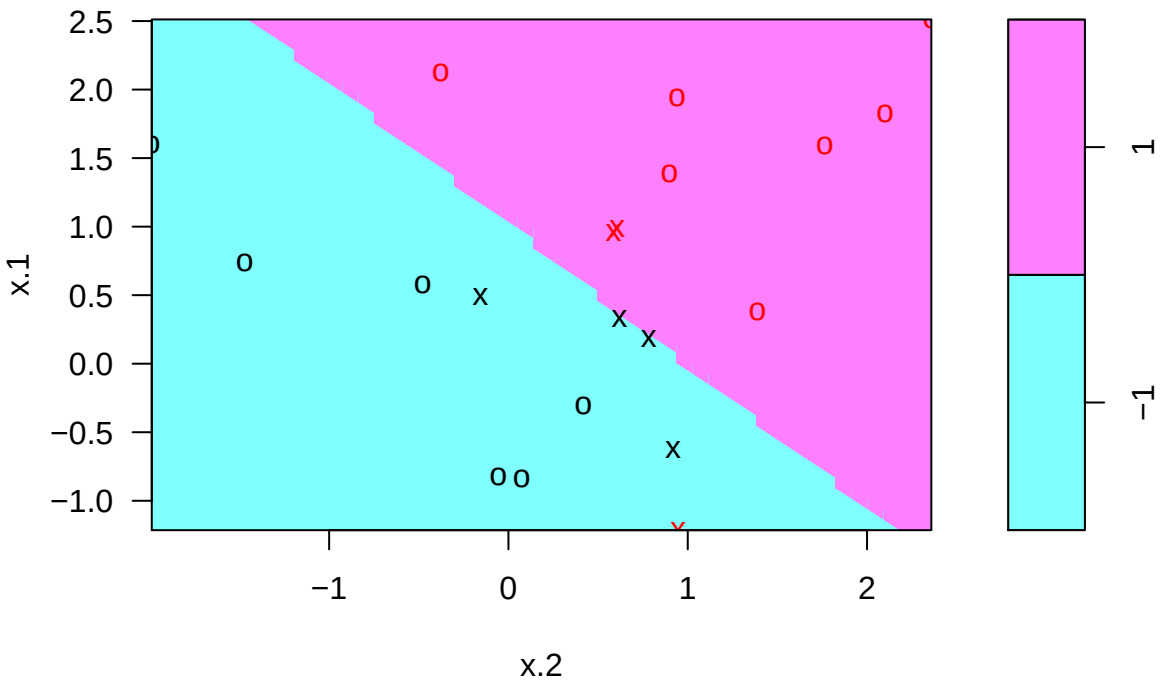
Lab

```
set.seed(1)
x <- matrix(rnorm(20*2), ncol = 2)
y <- c(rep(-1,10), rep(1,10))
x[y==1, ] <- x[y==1,] + 1

dat <- data.frame(x=x, y=as.factor (y))


ggplot(dat, aes(x = x.1, y = x.2, color = y)) +
  geom_point()
```

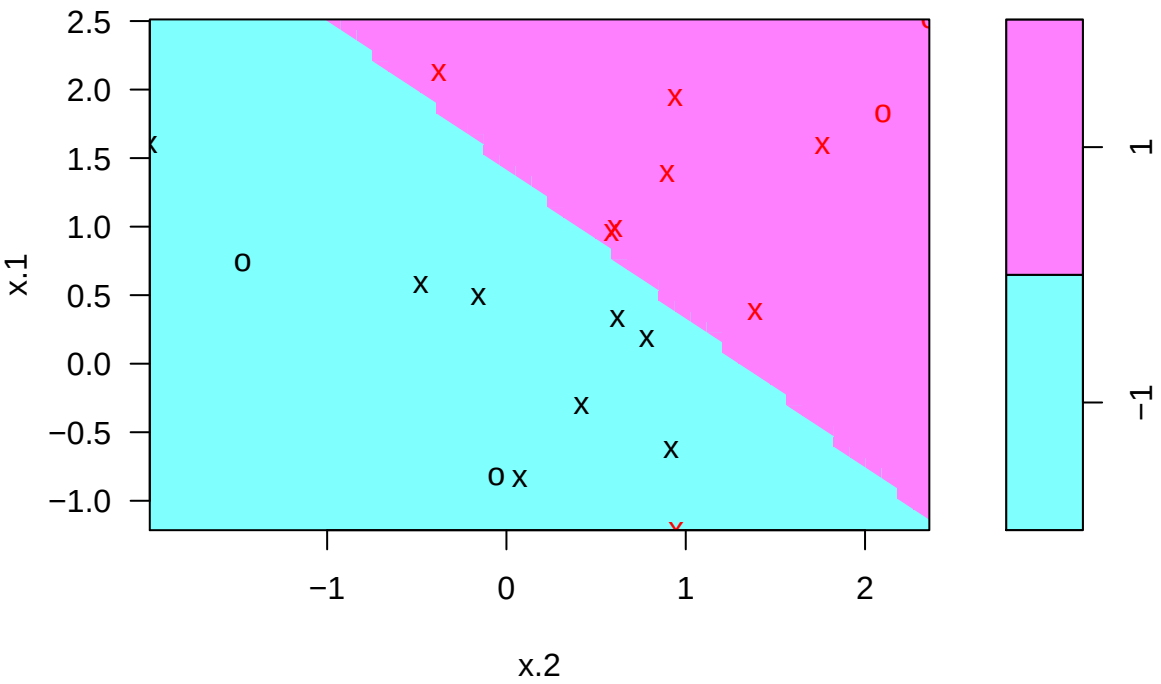


Fit the svm

```
library(e1071)

# thin magin
svm_fit <- svm(y ~ ., data = dat,
               kernel = "linear", cost = 10, scale = FALSE)
plot(svm_fit, data = dat)
```



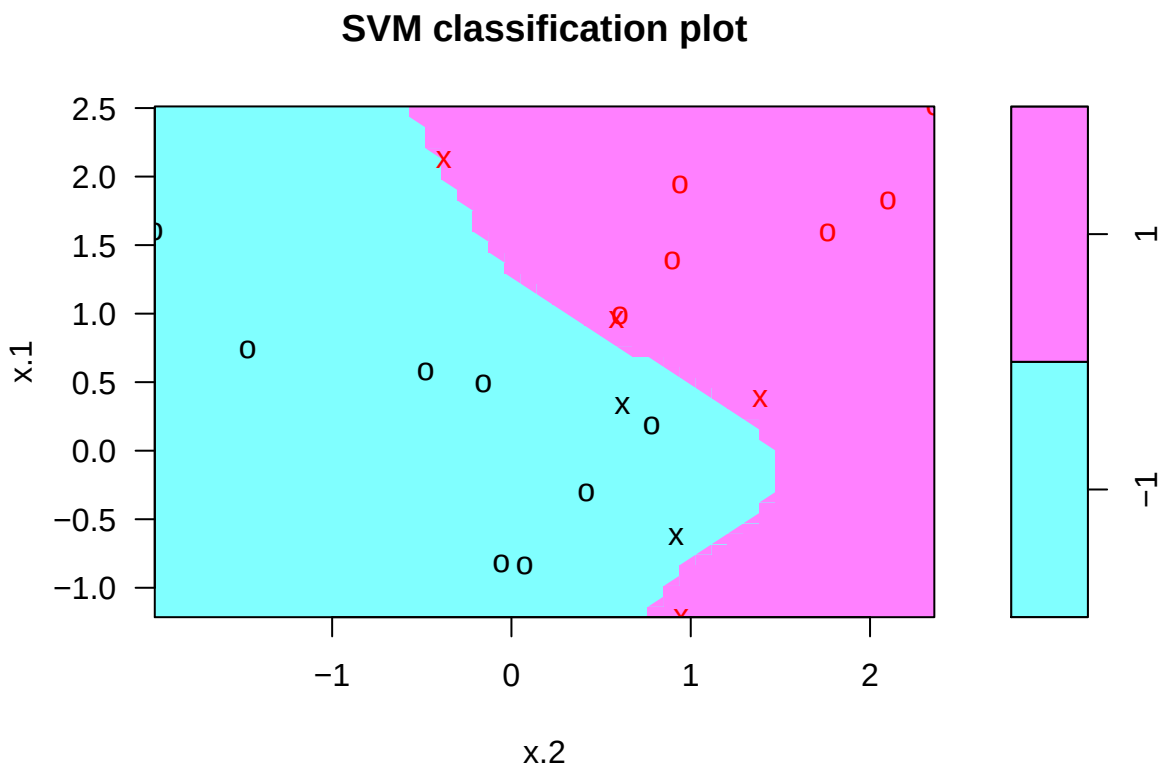

```
# wider margin
svm_fit <- svm(y ~ ., data = dat,
               kernel = "linear", cost = .1, scale = FALSE)

# note x's and o's

plot(svm_fit, data = dat)
```




```
summary(svm_fit)
```

```
##
## Call:
## svm(formula = y ~ ., data = dat, kernel = "linear", cost = 0.1,
##     scale = FALSE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.1
##       gamma:  0.5
##
## Number of Support Vectors:  16
##
##  ( 8 8 )
##
##
## Number of Classes:  2
##
## Levels:
##  -1 1
```

```
svm(y ~ ., data = dat,
               kernel = "polynomial", cost = 10, scale = FALSE) %>%
  plot(data = dat)
```




```
svm(y ~ ., data = dat,
                kernel = "radial", cost = 10, scale = FALSE) %>%
  plot(data = dat)
```