Support Vector Machine

Jaeyoon Han

2016-08-09

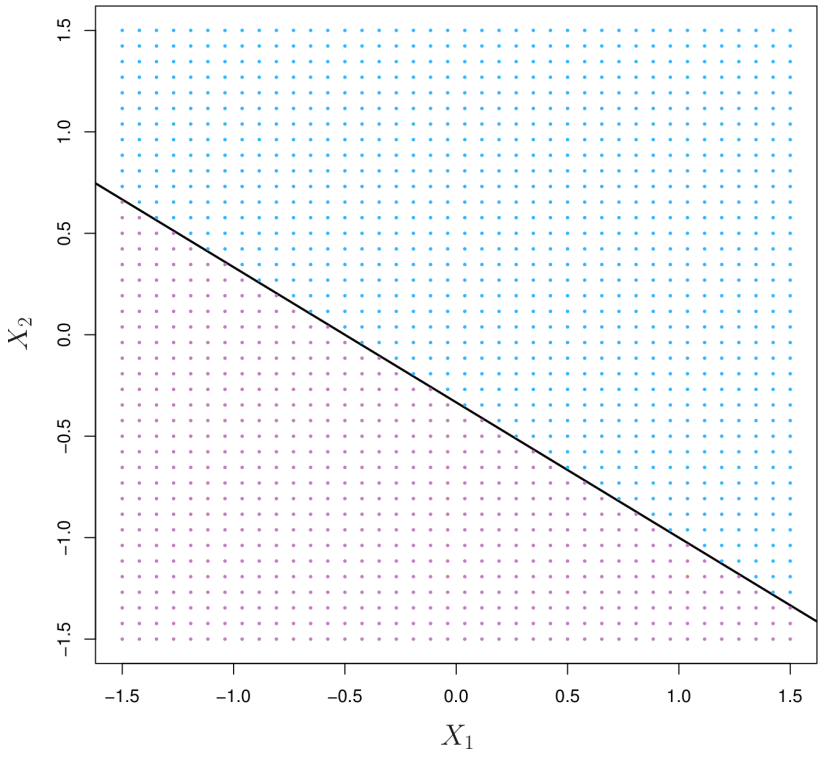
Support Vector Machine

서포트 벡터 머신(Support Vector Machine, SVM)은 1990년대에 개발된 분류 기법입니다. 현재 가장 일반적인 상황에서 좋은 성능을 보이는 비선형 분류기 중 하나입니다. 하지만, 그 기본 개념이 굉장히 어렵기 때문에 본 강의에서는 서포트 벡터 머신이 나오게 된 역사를 차근차근 되짚어보도록 하겠습니다.

History of SVM

1962, Maximal Margin Classifier

일반적인 유클리드 공간에 대해서 **초평면(Hyperplane)**을 정의해봅시다. 간단하게 원점(Origin)이 어디인지 알 수 없는 평면을 초평면이라고 합니다. 2차원 공간에서의 초평면은 선이 되죠. 이 초평면은 기존의 공간을 반으로 나눕니다.



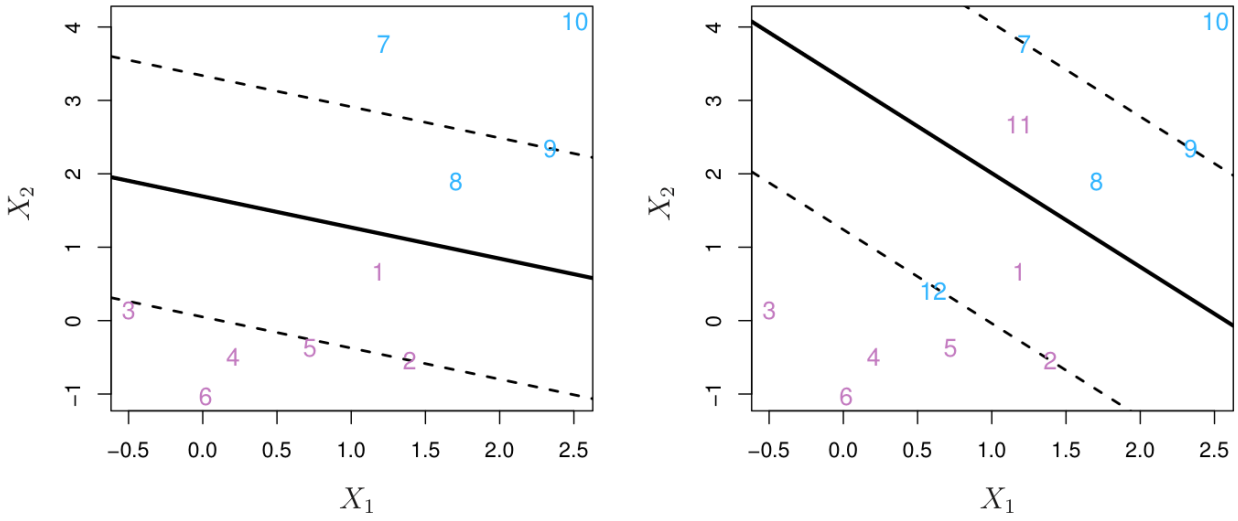
위 이미지는 초평면 $1 + 2X_1 + 3X_2 = 0$을 나타내고 있습니다. 윗쪽 파란색 영역은 해당 초평면에 대해서 $1 + 2X_1 + 3X_2 > 0$인 점들의 집합이고, 보라색 영역은 $1 + 2X_1 + 3X_2 < 0$인 점들의 집합입니다.

두 개의 클래스로 나뉘어진 어떤 데이터 포인트들이 유클리드 공간에 놓여져 있다고 생각해봅시다. 이 때, 해당 데이터 포인트들에 대해서 클래스를 완벽하게 나눌 수 있는 초평면이 있다고 가정해보죠. 이 초평면을 **분리 초평면(Sepration Hyperplane)**이라고 합니다. 이러한 분리 초평민이 존재한다면, 무한히 많은 분리 초평면이 존재할겁니다. 이미 존재하는 분리 초평면에서 1mm만 움직여도 분리 초평면이 되기 때문이죠. 이 중에서, 가장 적절한 분리 초평면은 데이터로부터 가장 거

리가 먼 분리 초평면일겁니다. 앞으로 이 성질을 갖는 초평면을 **최대 마진 초평면(Maximal Margin Hyperplane)**이라고 부를겁니다. 이 최대 마진 초평면을 이용한 분류기가 바로 **최대 마진 분류기(Maximal Margin Classifier)**입니다. 최대 마진 초평면과 관련된 여러 가지 용어는 PPT 자료에 첨부하였습니다.

최대 마진 분류기의 문제는 일반적인 상황에서 모든 데이터를 올바르게 나눌 수 없다는 점입니다. 모든 데이터에 대해서 완벽하게 분리할 수 있는 분리 초평면이 존재할 때만 최대 마진 분류기가 존재하죠. 이제 여기서, 조금의 타협을 보도록 하겠습니다.

1996, Soft Margin Classifier



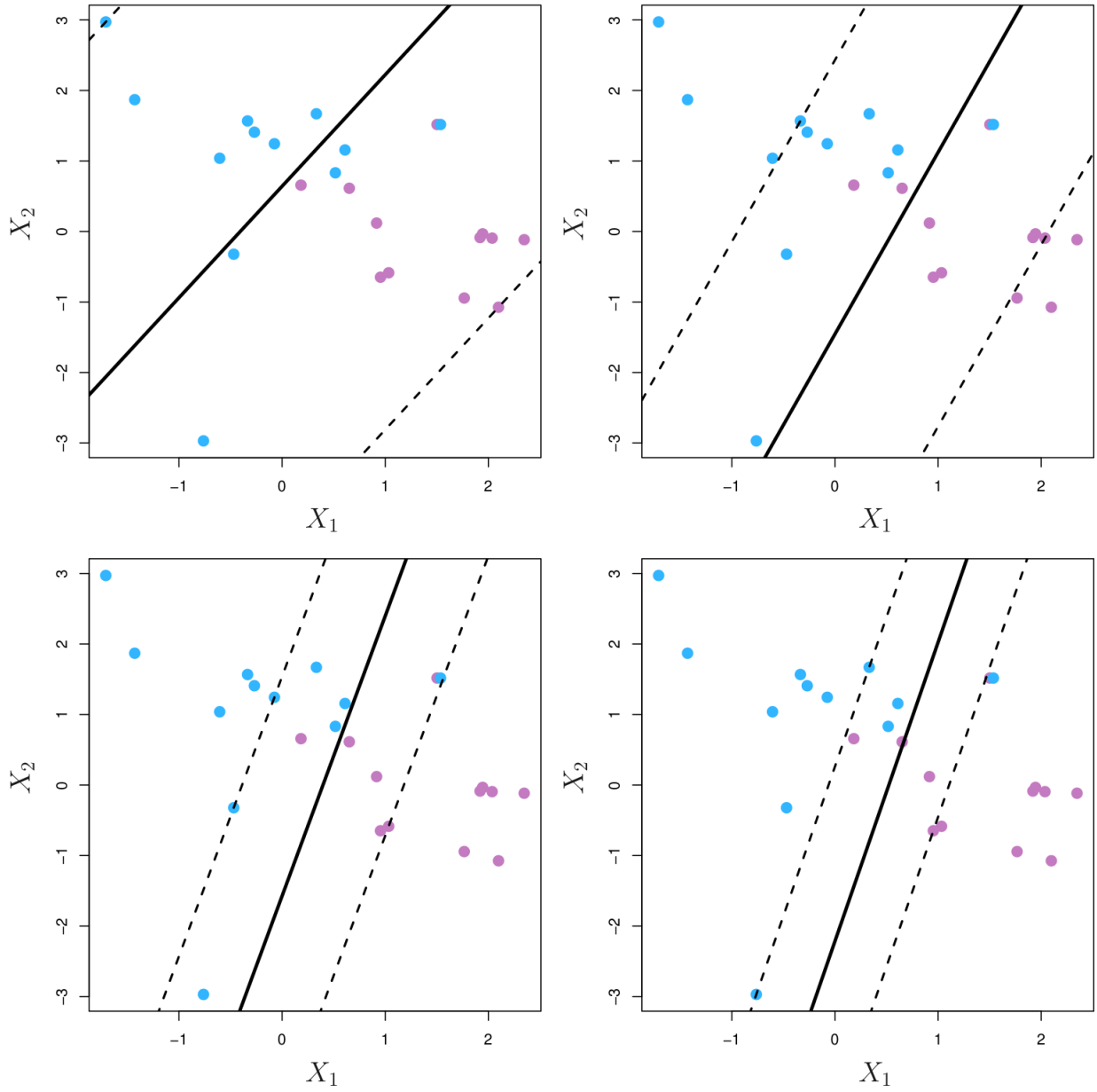
오분류된 데이터 일부를 허용할 수 있는 모델을 생각해보겠습니다. 이 때, '일부'의 정의를 잘 내리면 단단한(Robust) 모델을 만들 수 있을 것 같습니다. 그렇게 나온 개념이 바로 **슬랙 변수(Slack variable)**입니다. 슬랙 변수는 데이터 포인트의 위치를 나타냅니다.

- $e = 0$: 올바르게 분류되었다.
- $1 > e > 0$: 올바르게 분류되었으나, 마진을 넘어서는 범위에 있다.
- $e > 1$: 올바르게 분류되지 않았다.

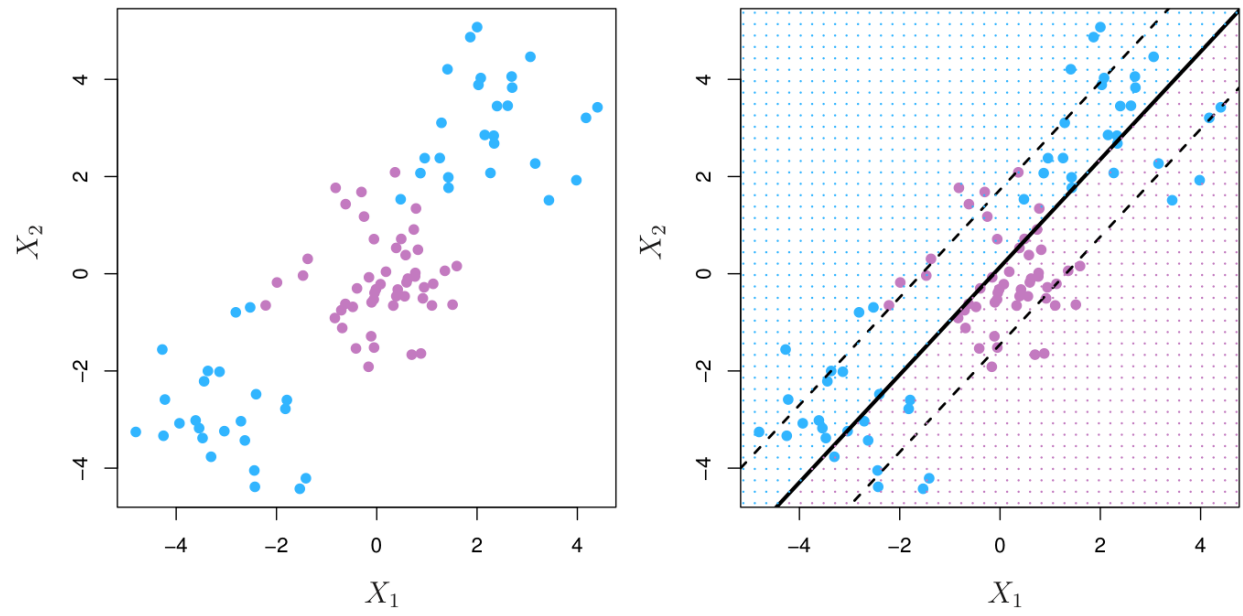
이제 각각의 데이터 포인트에 대해서 슬랙 변수값을 구한 후 모두 더합니다. 이 더한 값이 얼마나 많은 변수들이 오분류되었는지 말해줄겁니다.

하지만 이 값이 너무 커버리면 제대로 분류가 되지 않는 모델이 나올겁니다. 따라서 이 값에 한계를 주는 최대 비용(Maximum Cost)를 설정해줍니다. 서포트 벡터 머신에서 가장 중요한 Tuning Parameter인 C입니다. 이 값에 따라서 모델이 굉장히 많이 바뀌게 됩니다.

- C값이 크면 : 과소적합(Underfitting)
- C값이 작으면 : 과적합(Overfitting)



이 모델을 우리는 **소프트 마진 분류기(Soft Margin Classifier)** 또는 **서포트 벡터 분류기(Support Vector Classifier)** 라고 부릅니다. 이렇게 유연한 모델을 만들었지만, 결국 이런 문제는 언제나 발생합니다. 어떻게 해결해야 할까요?



1992, Kernel Trick

많은 머신러닝 연구자들은 위 이미지와 같은 문제를 해결하기 위해서 여러 가지 방법을 고안했습니다. 그 중에서 지금까지도 널리 쓰이는 해결법은 수학에서 나왔습니다. 바로 **커널 트릭(Kernel Trick)**입니다.

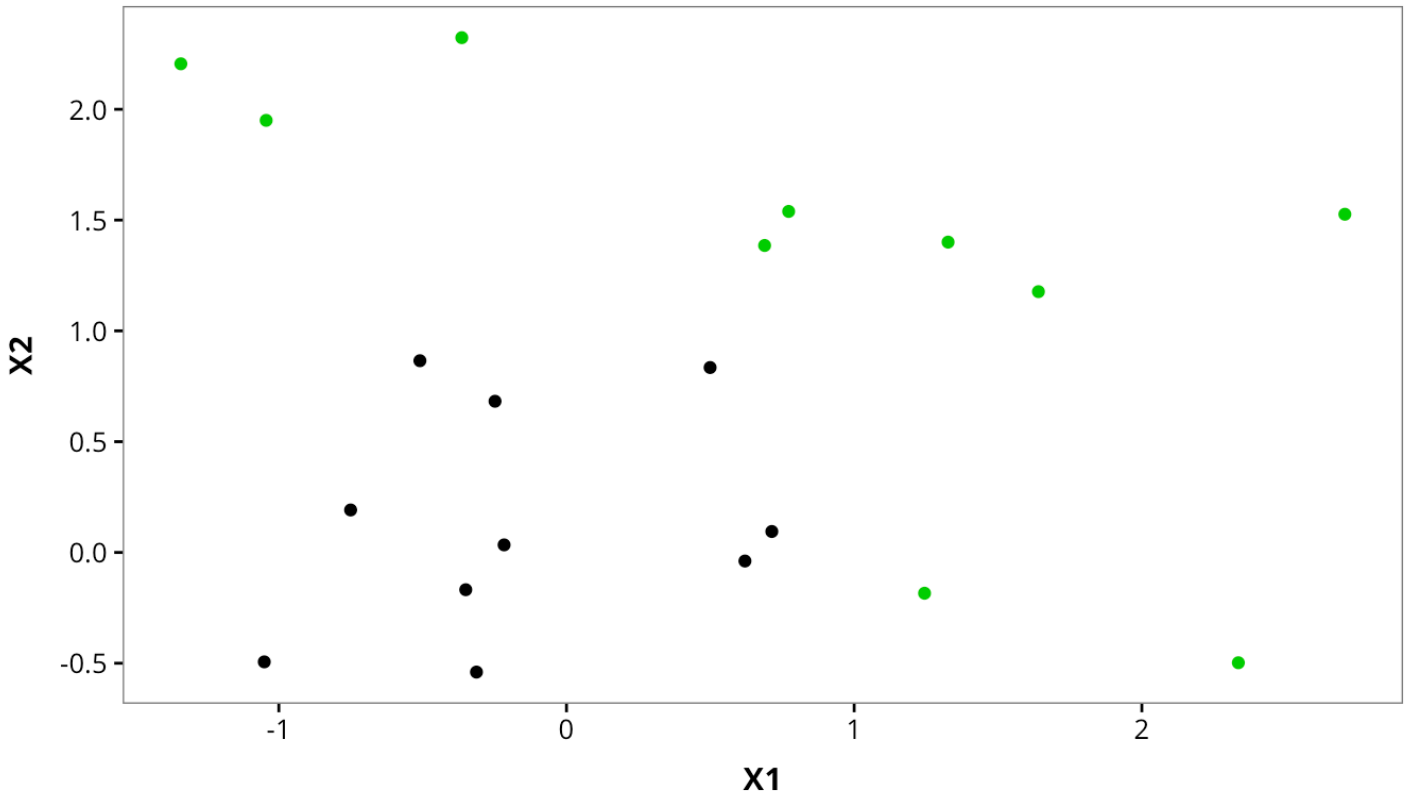
커널 트릭이란 어떤 변수 공간에 **커널 함수(Kernel function)**을 사용하여 그 공간을 확장시키는 방법입니다. 복잡하게 말하자면 커널 함수를 사용해서 데이터의 경계를 결정하는 결정경계면을 계산하고 클래스를 분류합니다. 커널 트릭을 사용하면 복잡한 형태의 데이터라도 결정경계면을 계산하여 클래스를 올바르게 분류할 수 있습니다. 다양한 커널 함수가 있지만 자주 사용하는 네 가지만 알아보도록 하겠습니다.

1. Linear Function : $K_l(x_i, x_j) = \langle x_i, x_j \rangle$
2. Polynomial Kernel : $K_p(x_i, x_j) = (\gamma \cdot \langle x_i, x_j \rangle + r)^d$
3. Gaussian Radial Basis Kernel : $K_r(x_i, x_j) = e^{-\gamma \cdot |x_i - x_j|^2}$, where $\gamma > 0$
4. Sigmoid Kernel : $K_s(x_i, x_j) = \tanh(\gamma \cdot \langle x_i, x_j \rangle + r)$

여기서 γ, r, d 값은 정해줘야 하는 파라미터로 이 값들에 따라서 결정경계면에 큰 차이가 생깁니다.

Toy Data

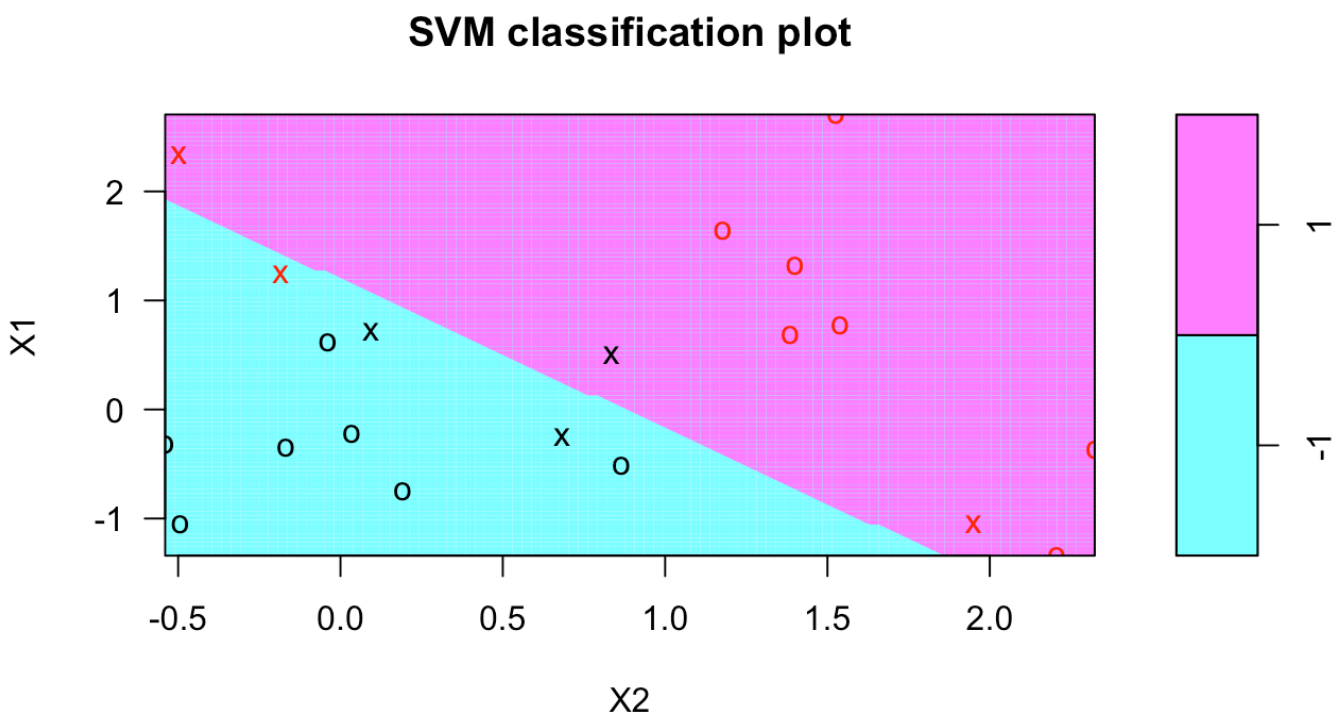
```
library(e1071)
set.seed(10111)
x = matrix(rnorm(40), 20, 2)
y = rep(c(-1, 1), c(10, 10))
x[y == 1, ] = x[y == 1, ] + 1

toy <- data.frame(x, y = factor(y))
ggplot(toy, aes(x = X1, y = X2)) +
        geom_point(color = y + 2)
```



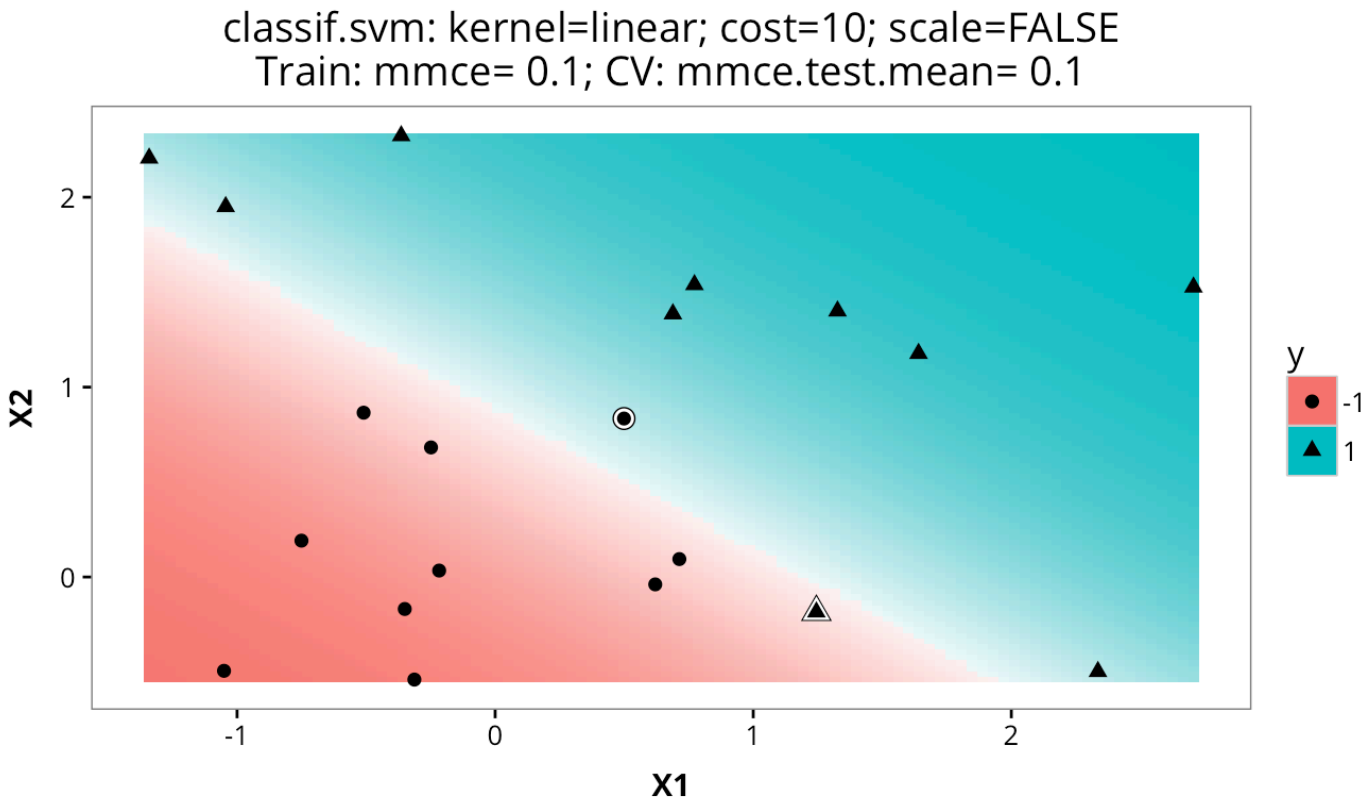
```
svm.fit <- svm(y ~ ., data = toy, kernel = 'linear', cost = 10, scale = FALSE)
summary(svm.fit)
```

```
Call:
svm(formula = y ~ ., data = toy, kernel = "linear", cost = 10,
    scale = FALSE)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  10
      gamma:  0.5

Number of Support Vectors:  6

 ( 3 3 )


Number of Classes:  2

Levels:
 -1 1
```

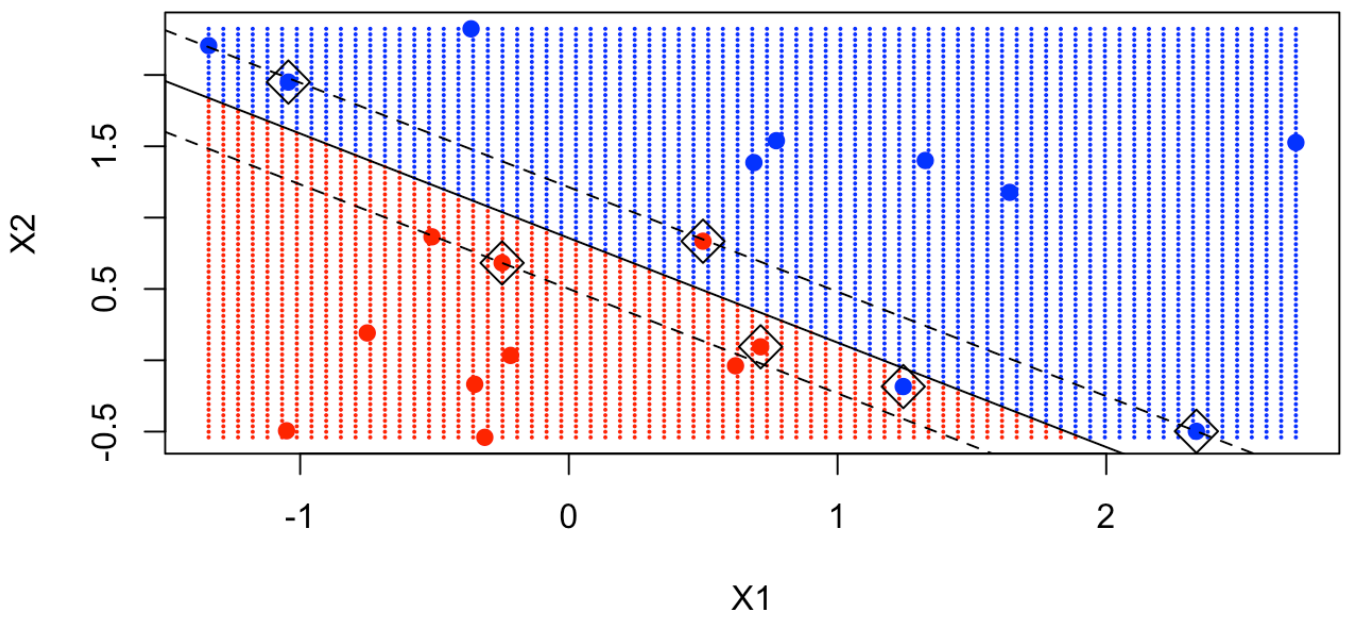
```
plot(svm.fit, toy, grid = 100)
```



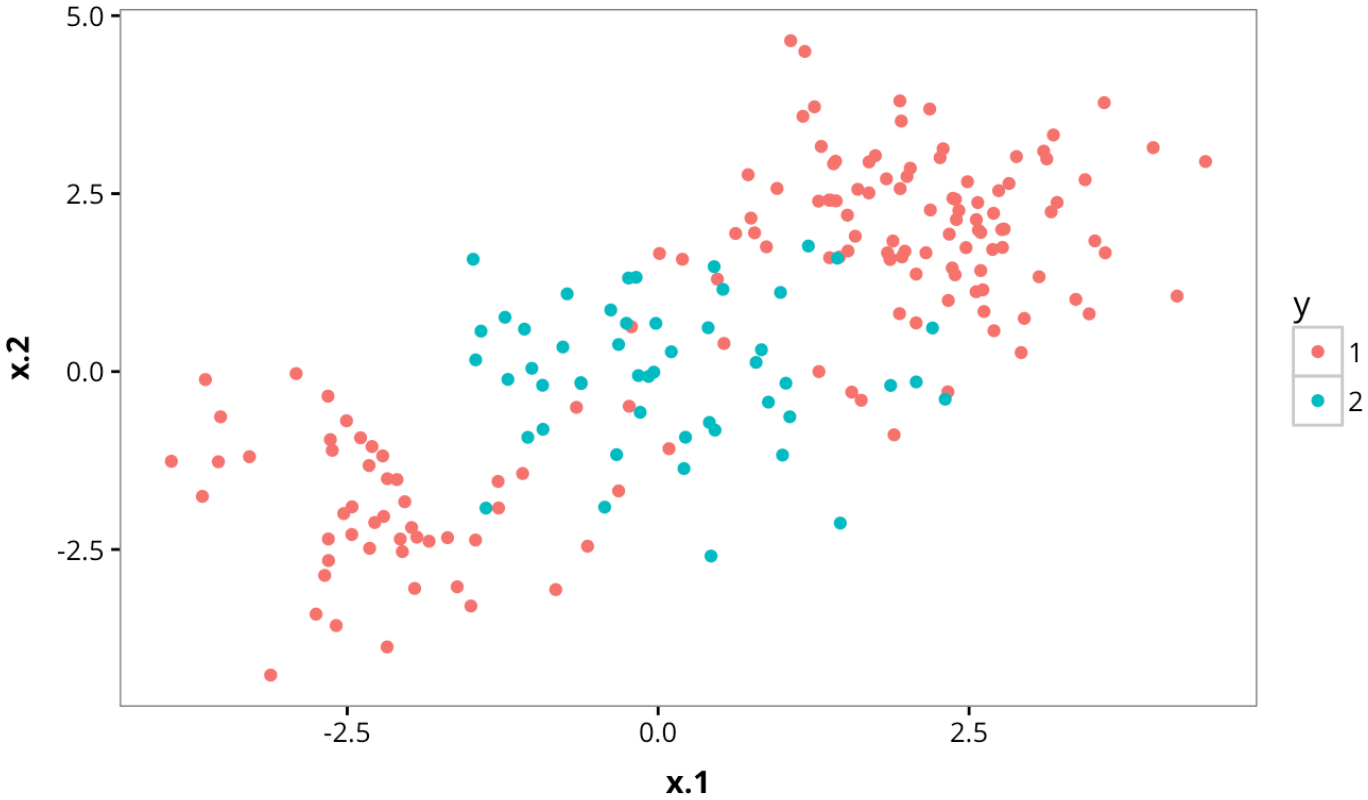
```
library(mlr)
toy.task <- makeClassifTask(id = "Toy", data = toy, target = "y")
learner <- makeLearner("classif.svm", kernel = "linear", cost = 10, scale = FALSE)
print(plotLearnerPrediction(learner, toy.task))
```



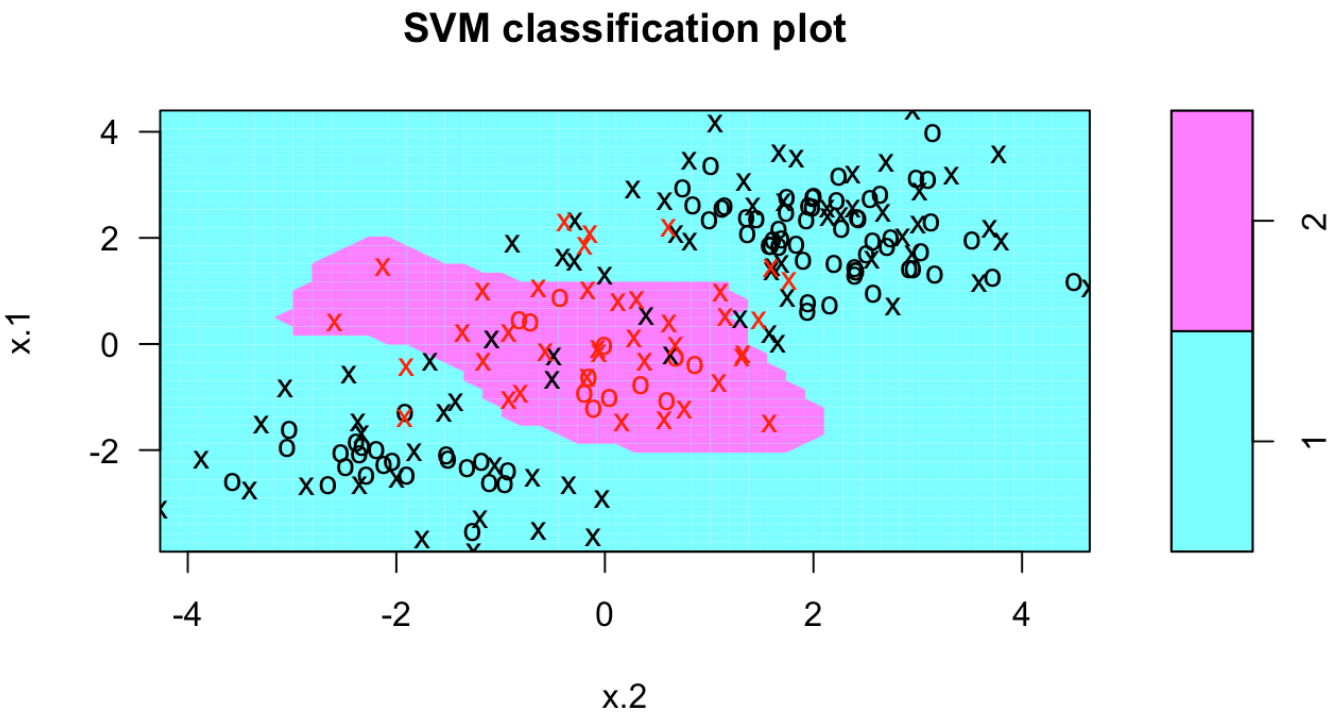

```
make.grid = function(x, n = 75) {
        grange = apply(x, 2, range)
        x1 = seq(from = grange[1, 1], to = grange[2, 1], length = n)
        x2 = seq(from = grange[1, 2], to = grange[2, 2], length = n)
        expand.grid(X1 = x1, X2 = x2)
}
xgrid = make.grid(x)
ygrid = predict(svm.fit, xgrid)
beta = drop(t(svm.fit$coefs) %*% x[svm.fit$index, ])
beta0 = svm.fit$rho
plot(xgrid, col = c("red", "blue")[as.numeric(ygrid)], pch = 20, cex = 0.2)
points(x, col = y + 3, pch = 19)
points(x[svm.fit$index, ], pch = 5, cex = 2)
abline(beta0/beta[2], -beta[1]/beta[2])
abline((beta0 - 1)/beta[2], -beta[1]/beta[2], lty = 2)
abline((beta0 + 1)/beta[2], -beta[1]/beta[2], lty = 2)
```



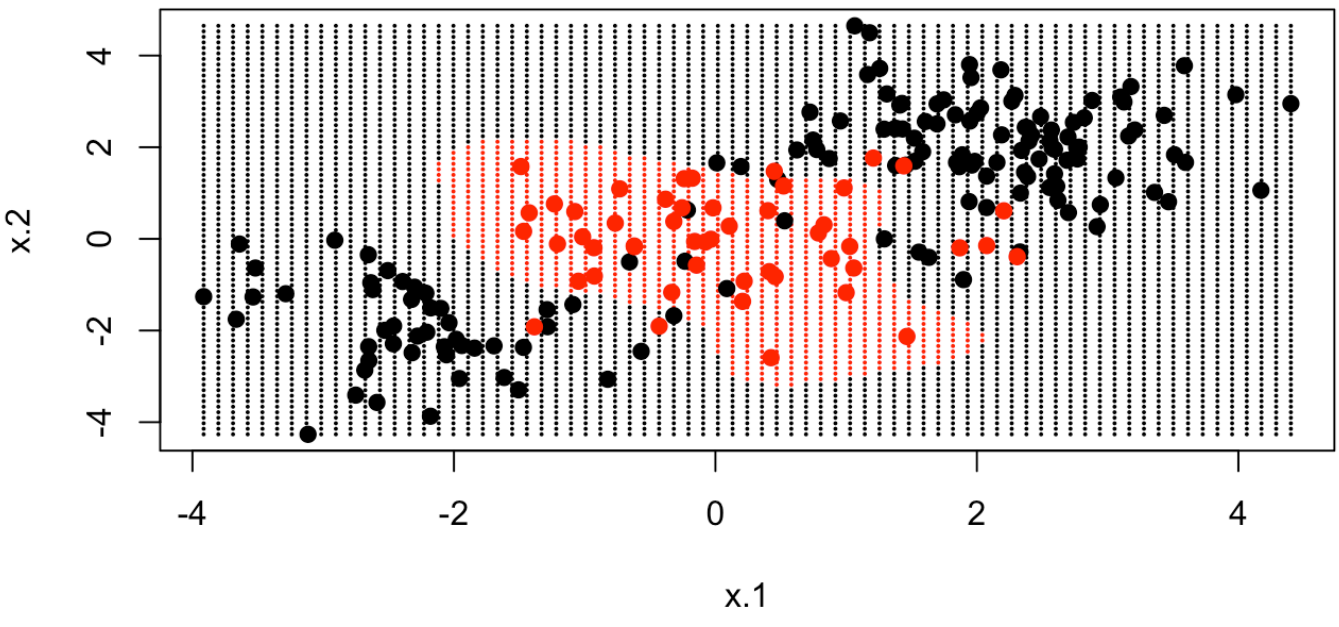

```
set.seed(1)
x <- matrix(rnorm(200 * 2), ncol = 2)
x[1:100, ] <- x[1:100, ] + 2
x[101:150, ] = x[101:150, ] - 2
y <- c(rep(1, 150), rep(2, 50))
toy2 <- data.frame(x=x,y=as.factor(y))

ggplot(data = toy2, aes(x = x.1, y = x.2, color = y)) +
        geom_point()
```



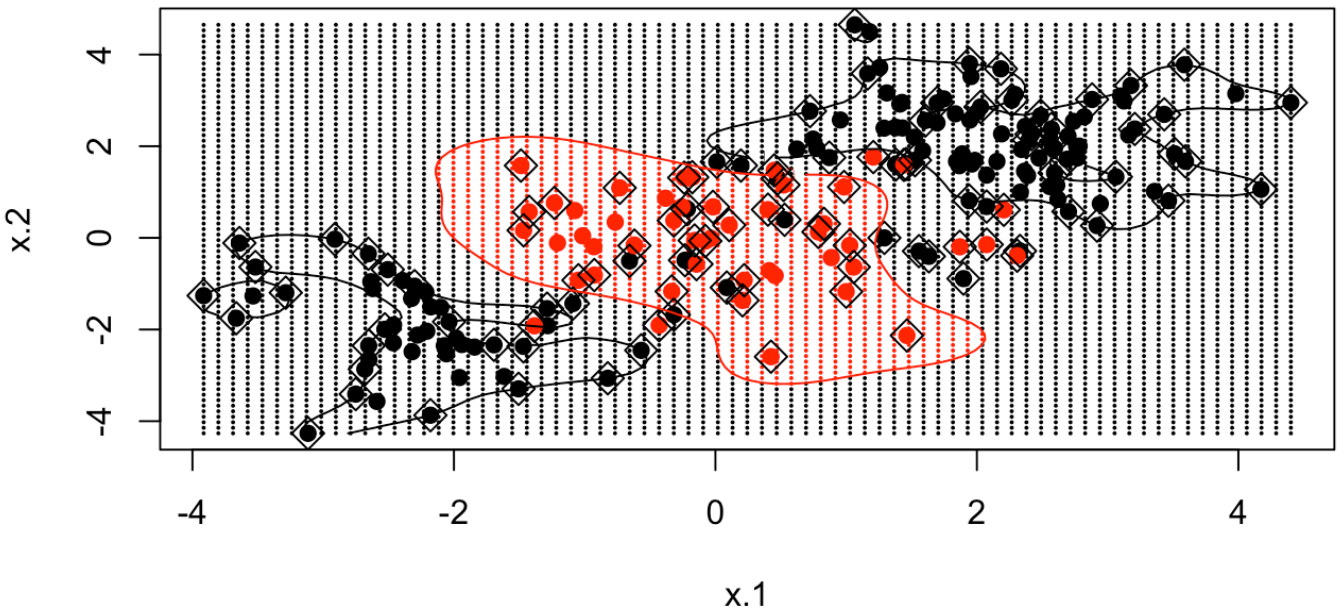
```
svm2 <- svm(y ~ ., data = toy2, kernel = 'radial', gamma = 2, cost = 1, scale = FALSE)
plot(svm2, toy2)
```



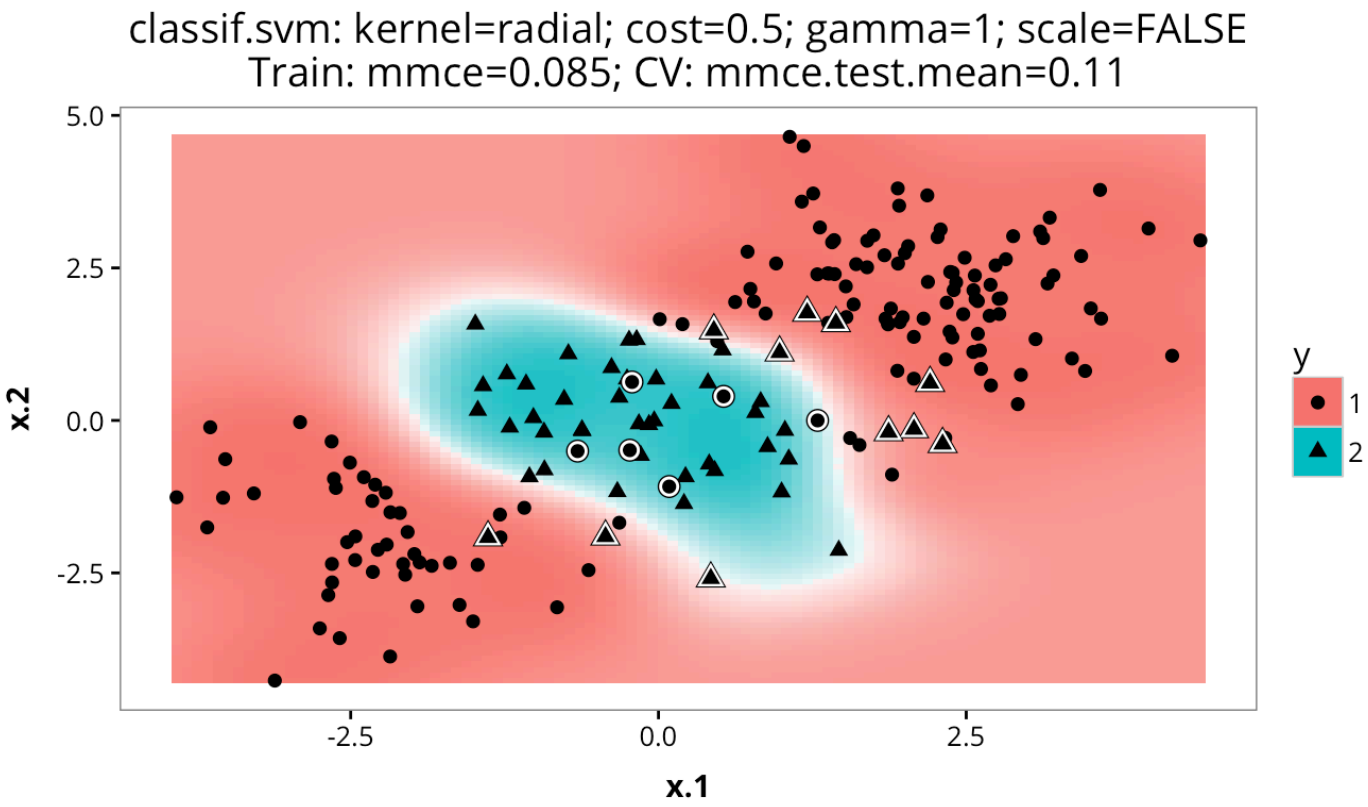

```
xgrid = make.grid(x)
names(xgrid) <- c("x.1", "x.2")
ygrid <- predict(svm2, xgrid)
plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = 0.2)
points(x, col = y, pch = 19)
```



```
func = predict(svm2, xgrid, decision.values = TRUE)
func = attributes(func)$decision
xgrid = make.grid(x)
names(xgrid) <- c("x.1", "x.2")
ygrid <- predict(svm2, xgrid)
plot(xgrid, col = as.numeric(ygrid), pch = 20, cex = 0.2)
points(x, col = y, pch = 19)
points(svm2$SV, pch = 5, cex = 1.5)

px1 <- unique(xgrid$x.1)
px2 <- unique(xgrid$x.2)
contour(px1, px2, matrix(func, 75, 75), level = 0, add = TRUE, col = "red")
contour(px1, px2, matrix(func, 75, 75), level = 1, add = TRUE, col = "black")
```




```
toy.task <- makeClassifTask(id = "Toy", data = toy2, target = "y")
learner <- makeLearner("classif.svm", kernel = "radial", cost = .5, gamma = 1, scale = FALSE)
print(plotLearnerPrediction(learner, toy.task))
```



Diabetes Diagnostics

```
library(e1071)
library(rpart)
library(mlbench)
library(MASS)
library(dplyr)
```

```
data(Pima.tr)
data(Pima.te)
str(Pima.tr)
```

```
'data.frame':   200 obs. of  8 variables:
 $ npreg: int  5 7 5 0 0 5 3 1 3 2 ...
 $ glu  : int  86 195 77 165 107 97 83 193 142 128 ...
 $ bp   : int  68 70 82 76 60 76 58 50 80 78 ...
 $ skin : int  28 33 41 43 25 27 31 16 15 37 ...
 $ bmi  : num  30.2 25.1 35.8 47.9 26.4 35.6 34.3 25.9 32.4 43.3 ...
 $ ped  : num  0.364 0.163 0.156 0.259 0.133 ...
 $ age  : int  24 55 35 26 23 52 25 24 63 31 ...
 $ type : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 2 1 1 1 2 ...
```

```
str(Pima.te)
```

```
'data.frame':   332 obs. of  8 variables:
 $ npreg: int  6 1 1 3 2 5 0 1 3 9 ...
 $ glu  : int  148 85 89 78 197 166 118 103 126 119 ...
 $ bp   : int  72 66 66 50 70 72 84 30 88 80 ...
 $ skin : int  35 29 23 32 45 19 47 38 41 35 ...
 $ bmi  : num  33.6 26.6 28.1 31 30.5 25.8 45.8 43.3 39.3 29 ...
 $ ped  : num  0.627 0.351 0.167 0.248 0.158 0.587 0.551 0.183 0.704 0.263 ...
 $ age  : int  50 31 21 26 53 51 31 33 27 29 ...
 $ type : Factor w/ 2 levels "No","Yes": 2 1 1 2 2 2 2 1 1 2 ...
```

```
summary(Pima.tr)
```

```
     npreg            glu              bp              skin      
 Min.   : 0.00   Min.   : 56.0   Min.   : 38.00   Min.   : 7.00  
 1st Qu.: 1.00   1st Qu.:100.0   1st Qu.: 64.00   1st Qu.:20.75  
 Median : 2.00   Median :120.5   Median : 70.00   Median :29.00  
 Mean   : 3.57   Mean   :124.0   Mean   : 71.26   Mean   :29.21  
 3rd Qu.: 6.00   3rd Qu.:144.0   3rd Qu.: 78.00   3rd Qu.:36.00  
 Max.   :14.00   Max.   :199.0   Max.   :110.00   Max.   :99.00  
      bmi             ped              age         type    
 Min.   :18.20   Min.   :0.0850   Min.   :21.00   No :132  
 1st Qu.:27.57   1st Qu.:0.2535   1st Qu.:23.00   Yes: 68  
 Median :32.80   Median :0.3725   Median :28.00            
 Mean   :32.31   Mean   :0.4608   Mean   :32.11            
 3rd Qu.:36.50   3rd Qu.:0.6160   3rd Qu.:39.25            
 Max.   :47.90   Max.   :2.2880   Max.   :63.00            
```

```
summary(Pima.te)
```

```
     npreg             glu              bp              skin      
 Min.   : 0.000   Min.   : 65.0   Min.   : 24.00   Min.   : 7.00  
 1st Qu.: 1.000   1st Qu.: 96.0   1st Qu.: 64.00   1st Qu.:22.00  
 Median : 2.000   Median :112.0   Median : 72.00   Median :29.00  
 Mean   : 3.485   Mean   :119.3   Mean   : 71.65   Mean   :29.16  
 3rd Qu.: 5.000   3rd Qu.:136.2   3rd Qu.: 80.00   3rd Qu.:36.00  
 Max.   :17.000   Max.   :197.0   Max.   :110.00   Max.   :63.00  
      bmi             ped              age         type    
 Min.   :19.40   Min.   :0.0850   Min.   :21.00   No :223  
 1st Qu.:28.18   1st Qu.:0.2660   1st Qu.:23.00   Yes:109  
 Median :32.90   Median :0.4400   Median :27.00            
 Mean   :33.24   Mean   :0.5284   Mean   :31.32            
 3rd Qu.:37.20   3rd Qu.:0.6793   3rd Qu.:37.00            
 Max.   :67.10   Max.   :2.4200   Max.   :81.00            
```

```
test_labels <- Pima.te$type
```

```
library(caret)
linear.tune <- tune.svm(type ~ ., data = Pima.tr, kernel = "linear", cost = c(0.001, 0.001, 0.1, 1,
5, 10))
summary(linear.tune)

best.linear <- linear.tune$best.model
linear.test <- predict(best.linear, Pima.te)
confusionMatrix(linear.test, test_labels)
```

```
poly.tune <- tune.svm(type ~., data = Pima.tr, kernel = "polynomial", degree = c(2, 3, 4, 5),
                      coef0 = c(0.1, 0.5, 1, 2, 3, 4), cost = c(0.001, 0.001, 0.1, 1, 5, 10))
summary(poly.tune)
best.poly <- poly.tune$best.model
poly.test <- predict(best.poly, Pima.te)
confusionMatrix(poly.test, test_labels)
```

```
radial.tune <- tune.svm(type ~., data = Pima.tr, kernel = "radial", gamma = c(0.1, 0.5, 1, 2, 3, 4)
, cost = c(0.001, 0.001, 0.1, 1, 5, 10))
summary(radial.tune)
best.radial <- radial.tune$best.model
radial.test <- predict(best.radial , Pima.te)
confusionMatrix(radial.test, test_labels)
```

```
sig.tune <- tune.svm(type ~., data = Pima.tr, kernel = "sigmoid", gamma = c(0.1, 0.5, 1, 2, 3, 4),
                     coef0 = c(0.1, 0.5, 1, 2, 3, 4), cost = c(0.001, 0.001, 0.1, 1, 5, 10))
summary(sig.tune)
best.sig <- sig.tune$best.model
sig.test <- predict(best.sig, Pima.te)
confusionMatrix(sig.test, test_labels)
```

With Scaling

```
train_n <- data.frame(apply(Pima.tr[, -8], 2, scale), type = Pima.tr$type)
test_n <- data.frame(apply(Pima.te[, -8], 2, scale), type = Pima.te$type)
```

```
linear.tune <- tune.svm(type ~ ., data = train_n, kernel = "linear", cost = c(0.001, 0.001, 0.1, 1,
5, 10))
summary(linear.tune)

best.linear <- linear.tune$best.model
linear.test <- predict(best.linear, test_n)
confusionMatrix(linear.test, test_labels)
```

```
poly.tune <- tune.svm(type ~., data = train_n, kernel = "polynomial", degree = c(2, 3, 4, 5),
                      coef0 = c(0.1, 0.5, 1, 2, 3, 4), cost = c(0.001, 0.001, 0.1, 1, 5, 10))
summary(poly.tune)
best.poly <- poly.tune$best.model
poly.test <- predict(best.poly, test_n)
confusionMatrix(poly.test, test_labels)
```

```
radial.tune <- tune.svm(type ~., data = train_n, kernel = "radial", gamma = c(0.1, 0.5, 1, 2, 3, 4)
, cost = c(0.001, 0.001, 0.1, 1, 5, 10))
summary(radial.tune)
best.radial <- radial.tune$best.model
radial.test <- predict(best.radial , test_n)
confusionMatrix(radial.test, test_labels)
```

```
sig.tune <- tune.svm(type ~., data = train_n, kernel = "sigmoid", gamma = c(0.1, 0.5, 1, 2, 3, 4),
                     coef0 = c(0.1, 0.5, 1, 2, 3, 4), cost = c(0.001, 0.001, 0.1, 1, 5, 10))
summary(sig.tune)
best.sig <- sig.tune$best.model
sig.test <- predict(best.sig, test_n)
confusionMatrix(sig.test, test_labels)
```