Machine Learning 2019

袁欣

2019 年 3 月 29 日

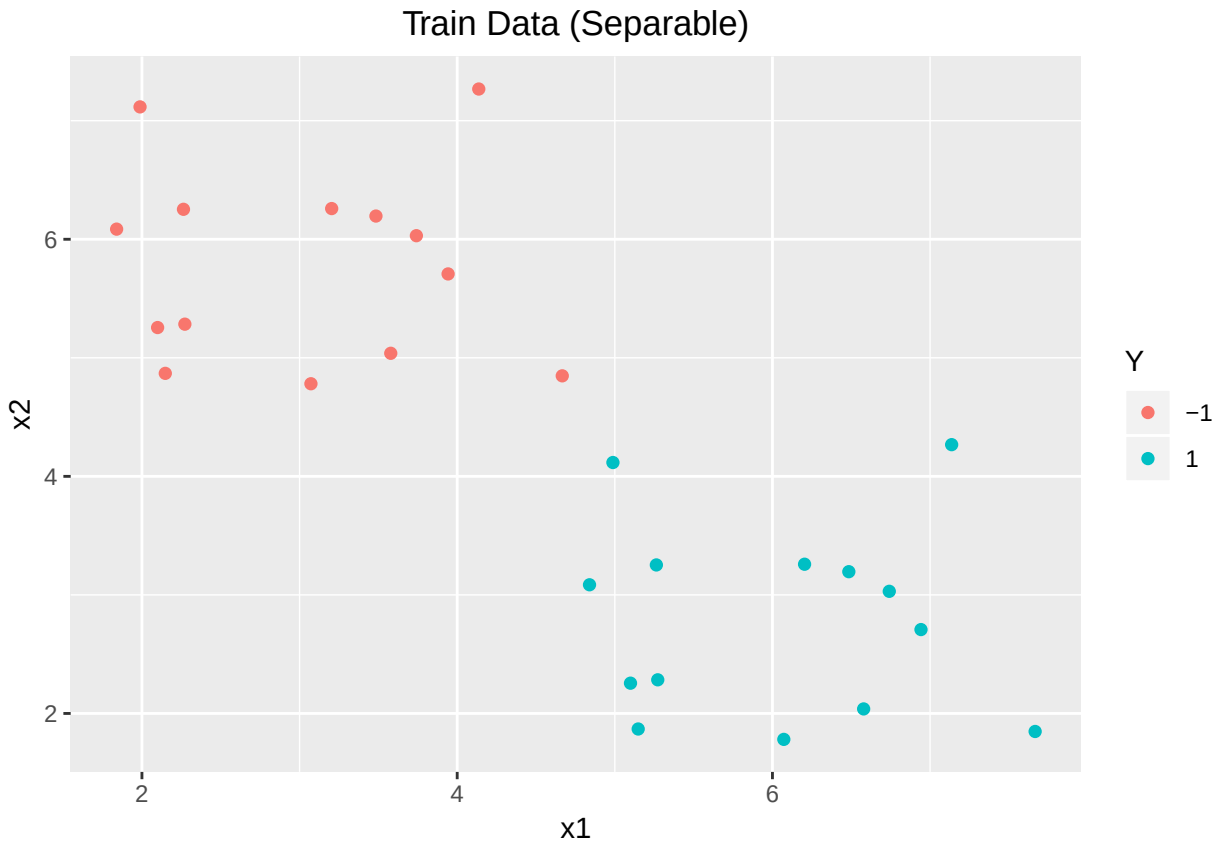
1 支持向量机

1.1 构建数据集

随机生成服从二元正态分布的 40 个样本，其中 20 个样本均值为 $[3,6]^T$，协方差阵为单位矩阵，设置标签为-1；20 个样本均值为 $[6,3]^T$，协方差阵仍为单位矩阵，设置标签为-1（确保两类样本线性可分）。然后按照 7：3 的比例将样本分为训练集与测试集，并画出训练集图形。

```
CreatData <- function(n, mu1, mu2, Sigma, seed = 3, alha = 0.7){
  set.seed(seed)
  X1 <- mvrnorm(n, mu1, Sigma)
  X1 <- data.frame(x1 = X1[, 1], x2 = X1[, 2])
  Y1 <- rep(-1, n)
  set.seed(seed)
  X2 <- mvrnorm(n, mu2, Sigma)
  X2 <- data.frame(x1 = X2[, 1], x2 = X2[, 2])
  Y2 <- rep(1, n)
  X1.train <- X1[1:floor(n*alha), ]
  X1.test <- X1[(floor(n*alha) + 1):n, ]
  X2.train <- X2[1:floor(n*alha), ]
  X2.test <- X2[(floor(n*alha) + 1):n, ]
  Y1.train <- Y1[1:floor(n*alha)]
  Y1.test <- Y1[(floor(n*alha) + 1):n]
  Y2.train <- Y2[1:floor(n*alha)]
  Y2.test <- Y2[(floor(n*alha) + 1):n]
  X.train <- rbind(X1.train, X2.train)
  X.test <- rbind(X1.test, X2.test)
  Y.train <- data.frame(Y = as.factor(c(Y1.train, Y2.train)))
  Y.test <- data.frame(Y = as.factor(c(Y1.test, Y2.test)))
  return(list(data.train = cbind(X.train, Y.train),
              data.test = cbind(X.test, Y.test)))
```

```
}
data1 <- CreatData(20, c(3, 6), c(6, 3), diag(2))
data1.train <- data1$data.train
data1.test <- data1$data.test
ggplot(data = data1.train, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  labs(title="Train Data (Separable)") + theme(plot.title = element_text(hjust = 0.5))
```



随机生成服从二元正态分布的 200 个样本，其中 100 个样本均值为 $[3,6]^T$，协方差阵为单位矩阵，设置标签为-1；100 个样本均值为 $[6,3]^T$，协方差阵仍为单位矩阵，设置标签为-1（确保两类样本线性不可分）。然后按照 7：3 的比例将样本分为训练集与测试集，并画出训练集图形。

```
data2 <- CreatData(100, c(3, 6), c(6, 3), diag(2), seed = 2)
data2.train <- data2$data.train
data2.test <- data2$data.test
ggplot(data = data2.train, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  labs(title="Train Data (Inseparable)") + theme(plot.title = element_text(hjust = 0.5))
```

1.2 支持向量机简介

1.2.1 基本形式

支持向量机就是找最优超平面去划分样本。超平面可以通过线性方程 $w^T + b = 0$ 来描述。其中 $w = (w_1; w_2; ...; w_d)$ 为法向量，b 为位移项。

样本空间中任意一点 x 到超平面 (w, b) 的距离可以写为

$$r = \frac{|w^T x + b|}{||w||}$$

最优的超平面就是使得距离超平面最近的点离超平面的距离最大。从上式可以看到当 w 改变时，分子与分母同时变化，不宜计算，我们把问题转化为以下形式, 令

$$\begin{cases} w^T x_i + b \geqslant 1, y_i = 1 \\ w^T x_i + b \leqslant -1, y_i = -1 \end{cases}$$

距离超平面最近的几个训练样本点使得上式的等号成立，他们被称为“支持向量”（support vector），两个异类支持向量到超平面的距离之和为

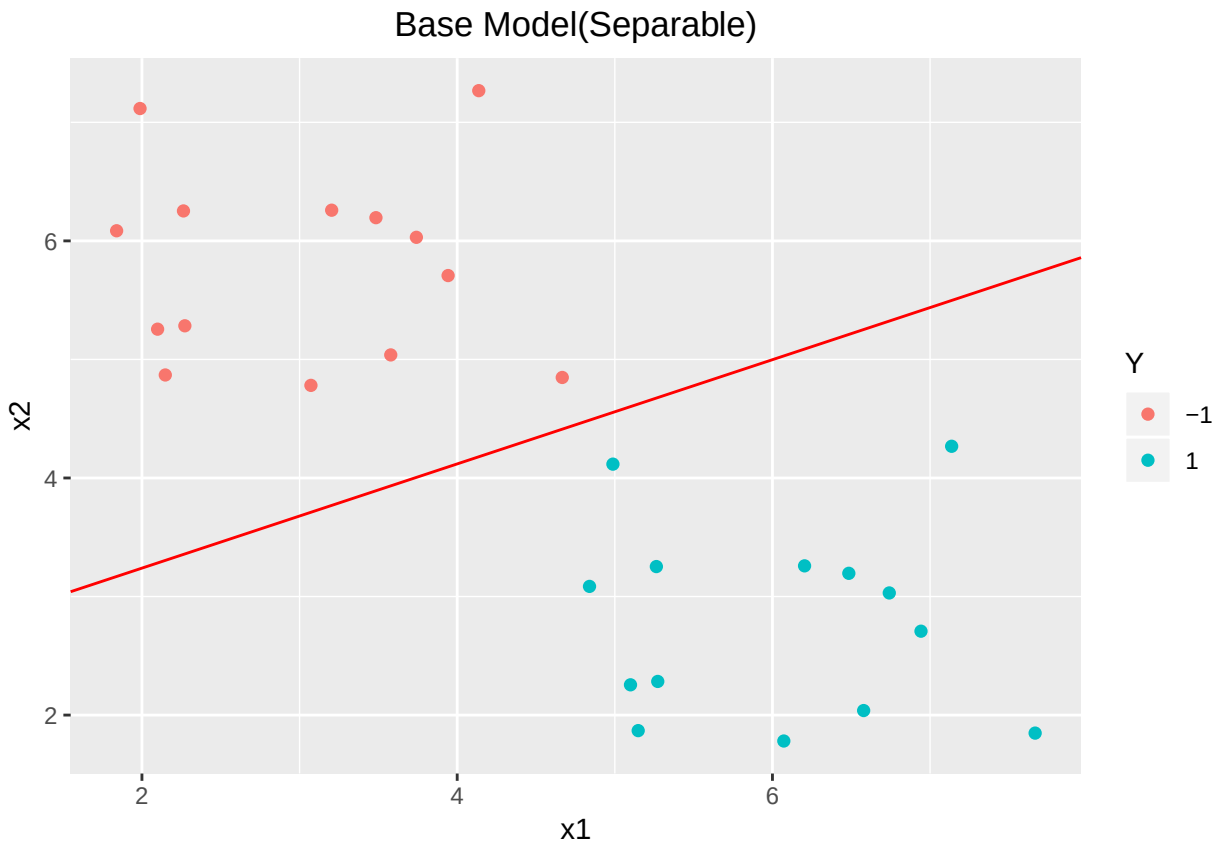
$$\gamma = \frac{2}{||w||}$$

于是支持向量机的基本模型就可以表示为

$$\min_{w,b} \frac{1}{2}||w||^2$$
$$s.t.\, y_i(w^T + b) \geqslant 1,\ i = 1, 2, ..., m$$

- 编程实现

```
X <- as.matrix(data1.train[, 1:2])
Y <- as.matrix(as.numeric(as.character(data1.train[, 3])))

SVMbase <- function(X, Y){
  #
  n <- ncol(X)
  m <- nrow(Y)
  w <- Variable(n)
  b <- Variable(1)
  objective <- Minimize(1/2 * norm(w, "F")^2)
  constraints <- list(Y * (X %*% w + b) >= 1)
  prob <- Problem(objective, constraints)

  # Problem solution
  solution <- solve(prob)
  w_res <- solution$getValue(w)
  b_res <- solution$getValue(b)
  return(list(w_res = w_res, b_res = b_res))
}
modelbase <- SVMbase(X, Y)
w_res <- modelbase$w_res
b_res <- modelbase$b_res

ggplot(data = data1.train, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  geom_abline(slope = -w_res[1]/w_res[2], intercept = -b_res/w_res[2], colour = "red") +
  labs(title="Base Model(Separable)") + theme(plot.title = element_text(hjust = 0.5))
```

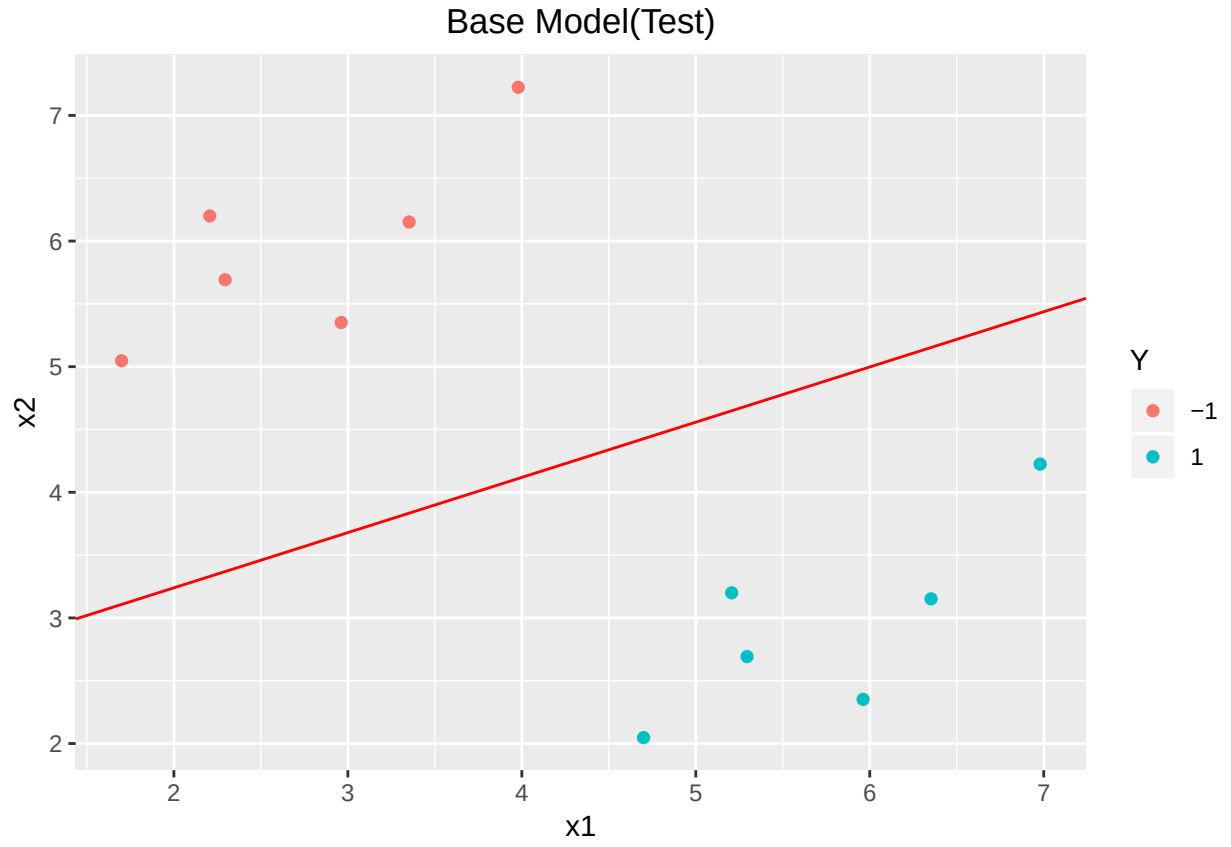


- 测试集检验

```
predictbase <- function(da, w_res, b_res){
  n <- ncol(da)
  m <- nrow(da)
  X <- as.matrix(da[, 1:(n-1)])
  preY <- X %*% w_res + b_res
  preY <- ifelse(preY >= 0, 1, -1)
  # print cof matrix
  TY <- da[, 3]
  print(table(preY, TY))

  return(preY)
}
preY <- predictbase(data1.test, w_res, b_res)
```

```
##     TY
## preY -1 1
##   -1  6 0
##   1   0 6
```

```
ggplot(data = data1.test, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  geom_abline(slope = -w_res[1]/w_res[2], intercept = -b_res/w_res[2], colour = "red") +
  labs(title="Base Model(Test)") + theme(plot.title = element_text(hjust = 0.5))
```



1.2.2 对偶问题

引入拉格朗日乘子 $\alpha_i \geqslant 0, i = 1, 2, ..., m$，定义拉格朗日函数：

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i(w^T x_i + b))$$

根据拉格朗日对偶性，原始问题的对偶问题是极大极小问题：

$$\max_{\alpha} \min_{w,b} L(w, b, \alpha)$$

令 $L(w, b, \alpha)$ 对 w 和 b 的偏导数为零可得

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i$$

$$0 = \sum_{i=1}^{m} \alpha_i y_i$$

将上式代入拉格朗日函数得：

$$L(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

最终得到对偶问题为：

$$\begin{aligned}\min_{\alpha}\quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j-\sum_{i=1}^{m}\alpha_i\\ s.t.\quad & \sum_{i=1}^{m}\alpha_i y_i=0\\ & \alpha_i\geqslant 0,\ \ i=1,2,...,m\end{aligned}$$

对于线性可分的数据集，假设对偶问题的解为 $\alpha^\star$，则原始问题的解为

$$w^\star=\sum_{i=1}^{m}\alpha_i^\star y_i x_i$$

$$b^\star=y_j-\sum_{i=1}^{m}\alpha_i^\star y_i(x_i^T x_j)$$

$b^\star$ 可通过任一支持向量求解（j 表示 α 不等于 0 的样本）。

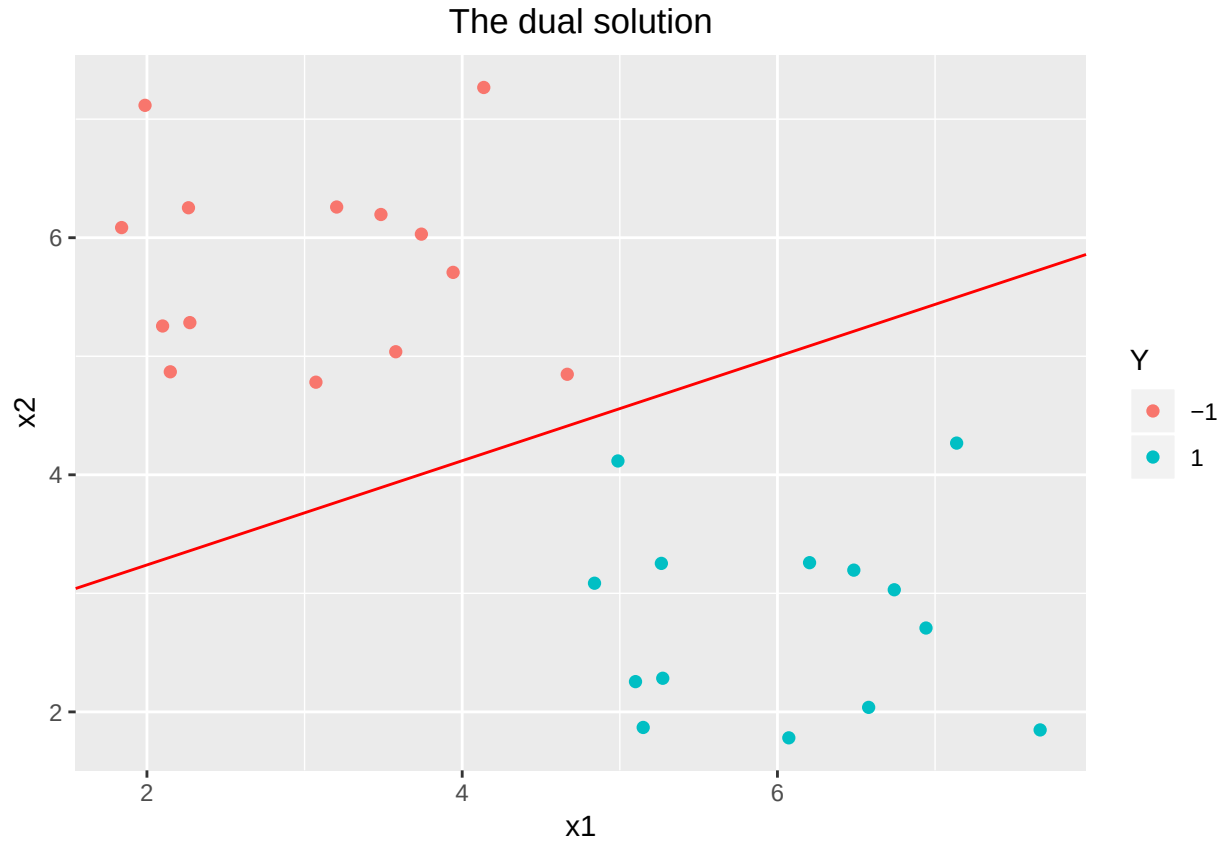
最终原始超平面可以表示为：

$$\sum_{i=1}^{m}\alpha_i y_i x_i^T x+b=0$$

- 编程实现

```
X <- as.matrix(data1.train[, 1:2])
Y <- as.matrix(as.numeric(as.character(data1.train[, 3])))

SVMdual <- function(X, Y){
  n <- ncol(X)
  m <- nrow(Y)
  A <- Variable(m)

  objective <- Minimize(-sum(A) + 1/2 * quad_form(Y * A, X %*%t(X)))
  constraints <- list(A >= 0, sum(Y * A) == 0)
  prob <- Problem(objective, constraints)

  # Problem solution
  solution <- solve(prob)
  A_res <- solution$getValue(A)

  w_res <- colSums(cbind(Y * A_res, Y * A_res) * X)
```

```
  # cal b_res(according to support vector)
  ind <- which(A_res > 0.00001)[1]
  b_res <- (1 - (Y[ind, 1])*(t(X[ind, ])%*%w_res)) / Y[ind, 1]

  # b_res <- Y[ind, 1] - sum(A_res * Y *  X %*% t(X[ind, , drop = FALSE]))
  return(list(w_res = w_res, b_res = b_res))
}
modeldual <- SVMdual(X, Y)
w_res <- modelbase$w_res
b_res <- modelbase$b_res

ggplot(data = data1.train, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  geom_abline(slope = -w_res[1]/w_res[2], intercept = -b_res/w_res[2], colour = "red") +
  labs(title="The dual solution") + theme(plot.title = element_text(hjust = 0.5))
```



1.2.3 核函数

当原始样本空间不存在能正确划分两类样本的超平面时，我们需要将样本从原始空间映射到一个更高维的特征空间，使得样本在这个特征空间内线性可分。理论已经证明，如果原始空间是有限维的，那么一定存在

一个高维特征空间使样本可分。

此时模型就可以写为

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \phi(x_i)^T\phi(x_j) - \sum_{i=1}^{m}\alpha_i \\ s.t. \quad & \sum_{i=1}^{m}\alpha_i y_i = 0 \\ & \alpha_i \geqslant 0, \ \ i = 1, 2, ..., m \end{aligned}$$

直接计算 $\phi(x_i)^T\phi(x_j)$ 通常是困难的。为了避开这个障碍，可以设想一个函数：

$$k(x_i, x_j) = \phi(x_i)^T\phi(x_j)$$

常用核函数有线性核与高斯核函数，其中高斯核函数的表达式为：

$$k(x_i, x_j) = exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$$

其中 $\sigma > 0$ 为高斯核的带宽。

- 编程实现

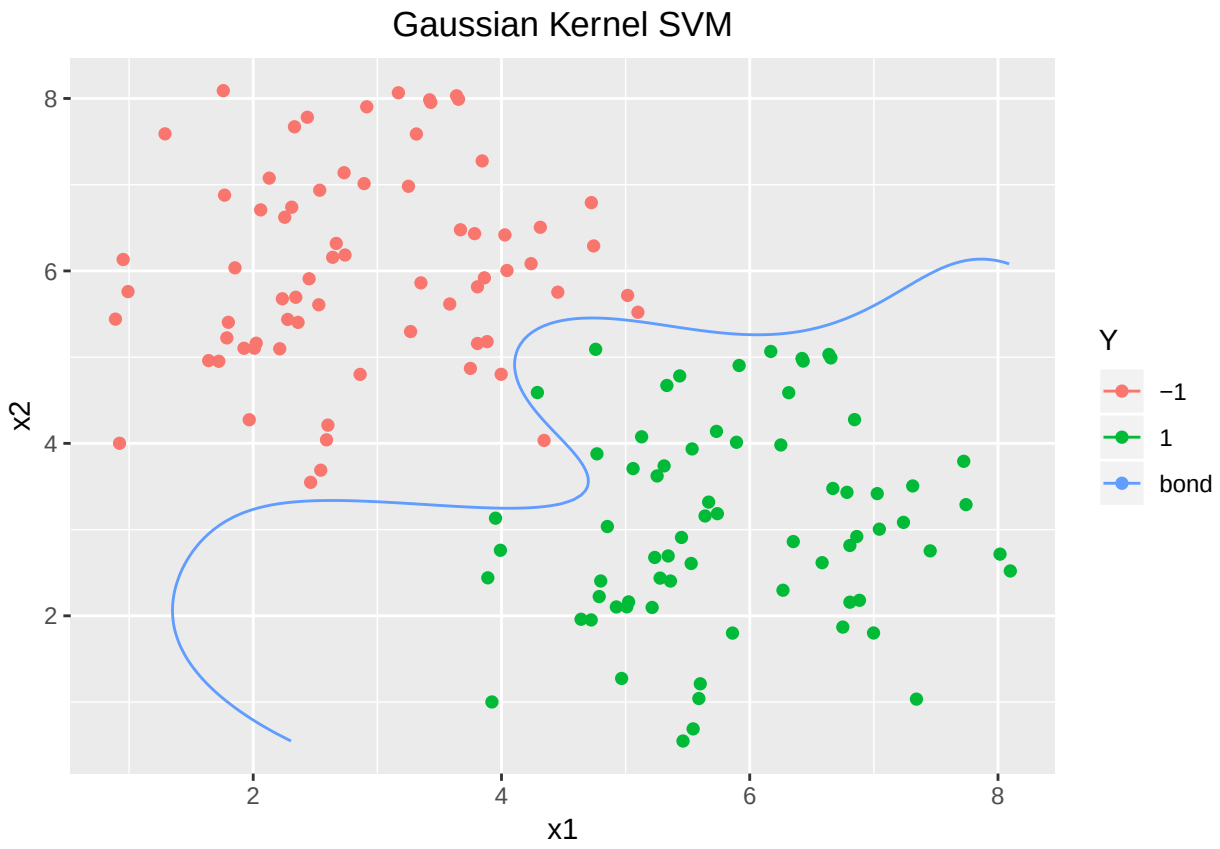
```
X <- as.matrix(data2.train[, 1:2])
Y <- as.matrix(as.numeric(as.character(data2.train[, 3])))

GaussianK <- function(xi, xj, Sigma = 1){
  temp <- exp(-sum((xi - xj)^2)/(2 * Sigma^2))
  return(temp)
}
SVMGaussiank <- function(X, Y, Sigma = 1){

  n <- ncol(X)
  m <- nrow(Y)
  A <- Variable(m)

  KernelM <- matrix(0, m, m)
  for(i in 1:m){
    for(j in 1:m){
      KernelM[i, j] <- GaussianK(X[i, ], X[j, ], Sigma)
    }
  }
  #
  objective <- Minimize(-sum(A) + 1/2 * quad_form(Y * A, KernelM))
```

```
  constraints <- list(A >= 0, sum(Y * A) == 0)
  prob <- Problem(objective, constraints)

  # Problem solution
  solution <- solve(prob)
  A_res <- solution$getValue(A)

  # cal b_res(according to support vector)
  ind <- which(A_res > 0.00001)[1]
  b_res <- Y[ind, 1] - sum(A_res * Y * KernelM[, ind])
  return(list(A_res = A_res, b_res = b_res, X = X, Y = Y, Sigma = Sigma))
}

modelGauss <- SVMGaussiank(X, Y, 1)

# predict

PredictGauss <- function(newda, modelGauss){
  #
  A_res <- modelGauss$A_res
  b_res <- modelGauss$b_res
  X <- modelGauss$X
  Y <- modelGauss$Y
  Sigma <- modelGauss$Sigma
  #
  newX <- as.matrix(newda[, 1:2])
  newY <- newda[, 3]
  newm <- nrow(newda)
  m <- nrow(X)
  KernelM <- matrix(0, m, newm)
  for(i in 1:m){
    for(j in 1:newm){
      KernelM[i, j] <- GaussianK(X[i, ], newX[j, ], Sigma)
    }
  }
  #
  preYvalue <- colSums(matrix(rep(A_res * Y, newm), ncol = newm) * KernelM) + b_res
  preSign <- ifelse(preYvalue >= 0, 1, -1)
  tb <- table(preSign, newY)
```

```
  return(list(preYvalue = preYvalue, preSign = preSign, tb = tb))
}

PreGauss.train <- PredictGauss(data2.train, modelGauss)
# The Train confusion matrix
PreGauss.train$tb
```

```
##        newY
## preSign -1  1
##      -1 70  0
##      1   0 70
```

```
Wherefx0 <- function(modelGauss){
  #
  X <- modelGauss$X
  Sigma <- modelGauss$Sigma
  #
  x1 <- seq(min(X[, 1]), max(X[, 1]),0.05)
  x2 <- seq(min(X[, 2]), max(X[, 2]),0.05)
  newda <- outer(x1, x2)
  colnames(newda) <- x2
  rownames(newda) <- x1
  newda <- melt(newda)
  newda$value <- 0
  colnames(newda) <- c("x1", "x2", "Y")
  Pred <- PredictGauss(newda, modelGauss)
  newda$Y <- Pred$preYvalue
  return(newda)
}

fx0 <- Wherefx0(modelGauss)

ggplot(data = data2.train, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  labs(title="Gaussian Kernel SVM") +
  theme(plot.title = element_text(hjust = 0.5)) +
  stat_contour(data = fx0, aes(x = x1, y = x2, z = Y, colour = "bond"), breaks=c(0))
```

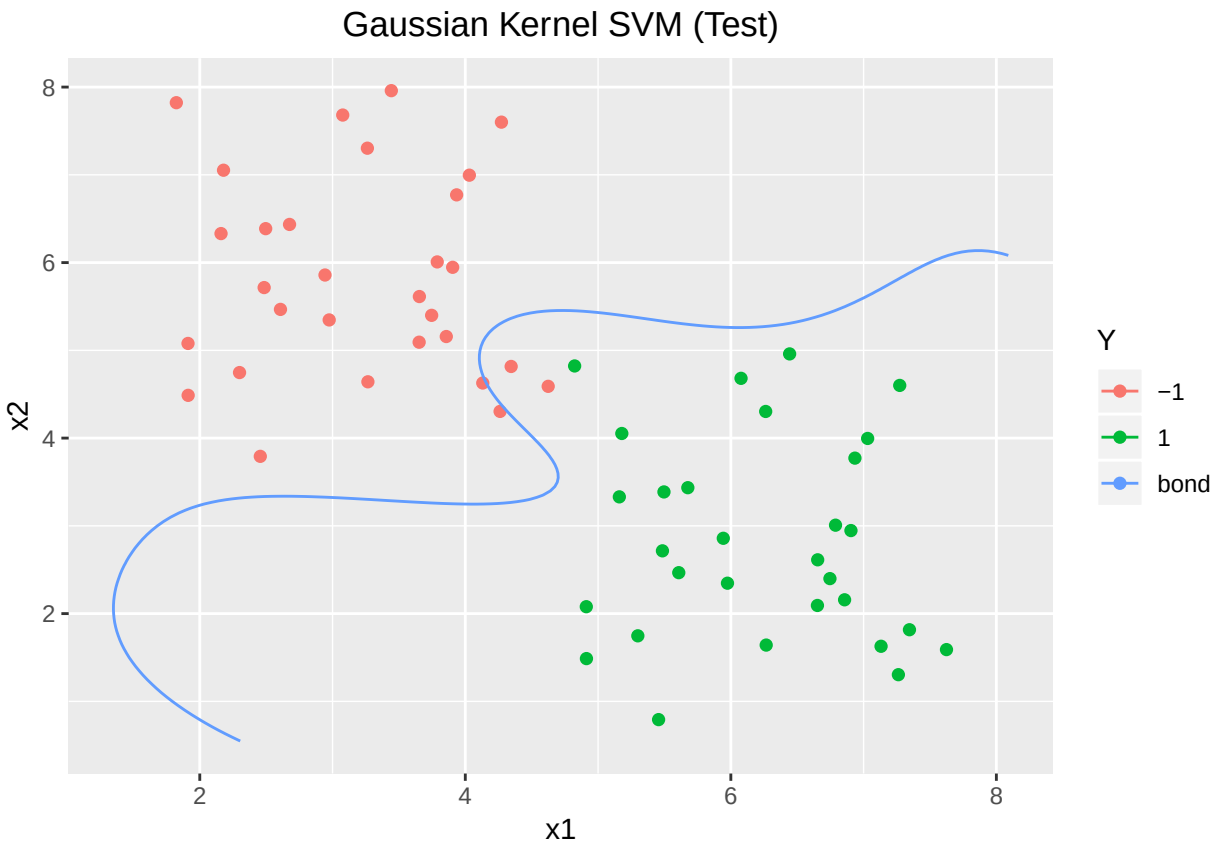


- 测试集检验

```
PreGauss.test <- PredictGauss(data2.test, modelGauss)
# The Test confusion matrix
PreGauss.test$tb
```

```
##         newY
## preSign -1  1
##      -1 28  0
##      1   2 30
```

```
ggplot(data = data2.test, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  labs(title="Gaussian Kernel SVM (Test)") +
  theme(plot.title = element_text(hjust = 0.5)) +
  stat_contour(data = fx0, aes(x = x1, y = x2, z = Y, colour = "bond"), breaks=c(0))
```



1.2.4 软间隔与正则化

当样本空间线性不可分时，我们除了引入核函数的方法以外，还可以利用软间隔与正则化。软间隔（soft margin）的思想是允许支持向量机在一些样本上出错。软间隔允许某些样本不满足约束 $y_i(w^T x_i + b) \geqslant 1$。当然，在最大化间隔的同时，不满足约束的样本因该尽可能少，于是优化目标可写为

$$min\, w,\, b \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} l_{0/1}(y_i(w^T x_i + b) - 1)$$

其中 $C > 0$ 是一个常数，$l_{0/1}$ 是损失函数。

$$l_{0/1} = \begin{cases} 1, & if \;\; z < 0 \\ 0, & otherwise \end{cases}$$

然而，$l_{0/1}$ 非凸、非连续，不宜求解。于是人们提出了很多替代函数，比如 hinge 损失函数 $l_{hinge}(z) = max(0,\, 1 - z)$，那么优化目标变为

$$min\, w,\, b \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} max(0,\, 1 - y_i(w^T x_i + b))$$

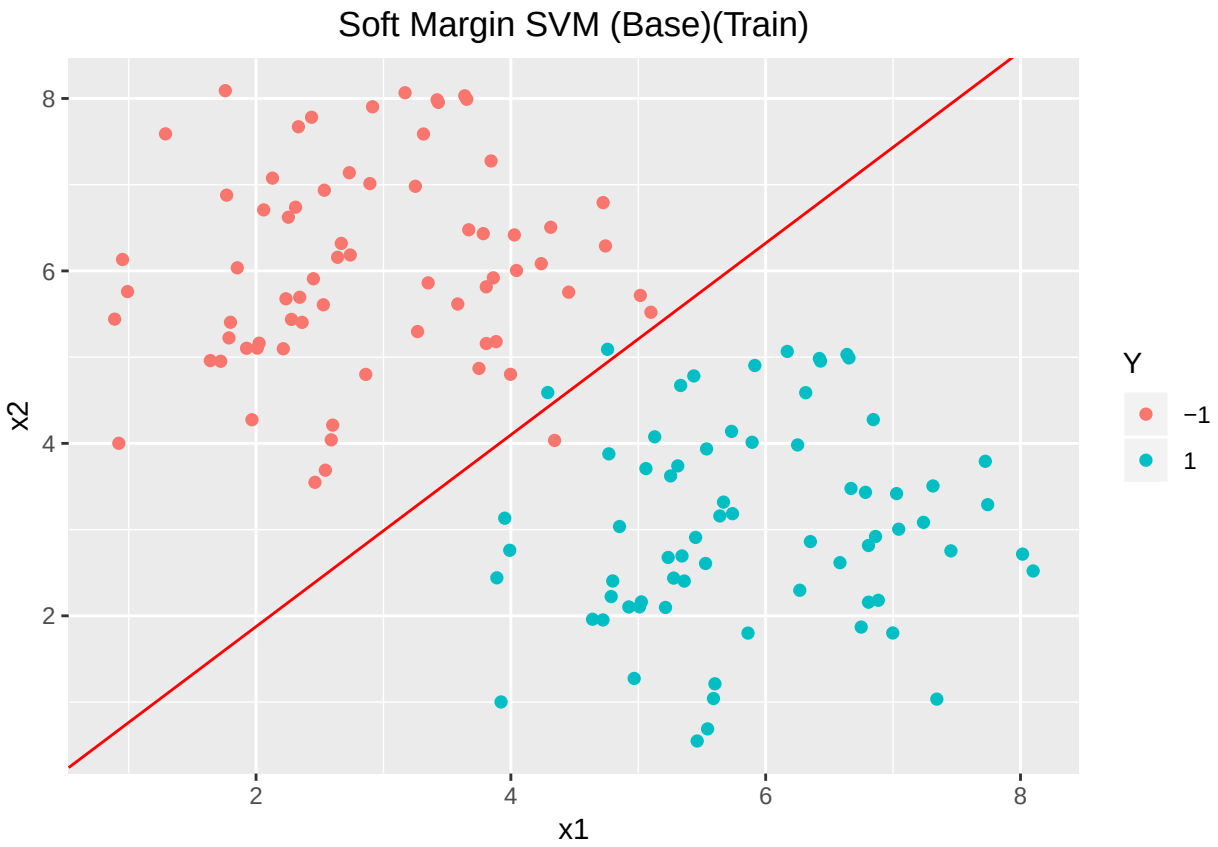
引入 “松弛变量”（slack variables）$\xi_i \geqslant 0$，可将上式重写为

$$
\begin{aligned}
min\, w,\, b \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\xi_i \\
s.t. \quad & y_i(w^T x_i + b) \geqslant 1 - \xi_i \\
& \xi_i \geqslant 0, \;\; i = 1, 2, ..., m
\end{aligned}
$$

这就是常见的“软间隔支持向量机”。

- 编程实现

```
X <- as.matrix(data2.train[, 1:2])
Y <- as.matrix(as.numeric(as.character(data2.train[, 3])))

SVMSoftMargin <- function(X, Y, C = 1){
  #
  n <- ncol(X)
  m <- nrow(Y)
  w <- Variable(n)
  b <- Variable(1)
  xi <- Variable(m)
  objective <- Minimize(1/2 * norm(w, "F")^2 + C*sum(xi))
  constraints <- list(xi >= 0, Y * (X %*% w + b) >= 1-xi)
  prob <- Problem(objective, constraints)

  # Problem solution
  solution <- solve(prob)
  w_res <- solution$getValue(w)
  b_res <- solution$getValue(b)
  return(list(w_res = w_res, b_res = b_res))
}
modelsoftmargin <- SVMSoftMargin(X, Y, C = 1)
w_res <- modelsoftmargin$w_res
b_res <- modelsoftmargin$b_res

ggplot(data = data2.train, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  geom_abline(slope = -w_res[1]/w_res[2], intercept = -b_res/w_res[2], colour = "red") +
  labs(title="") + theme(plot.title = element_text(hjust = 0.5)) +
  labs(title="Soft Margin SVM (Base)(Train)") +
  theme(plot.title = element_text(hjust = 0.5))
```

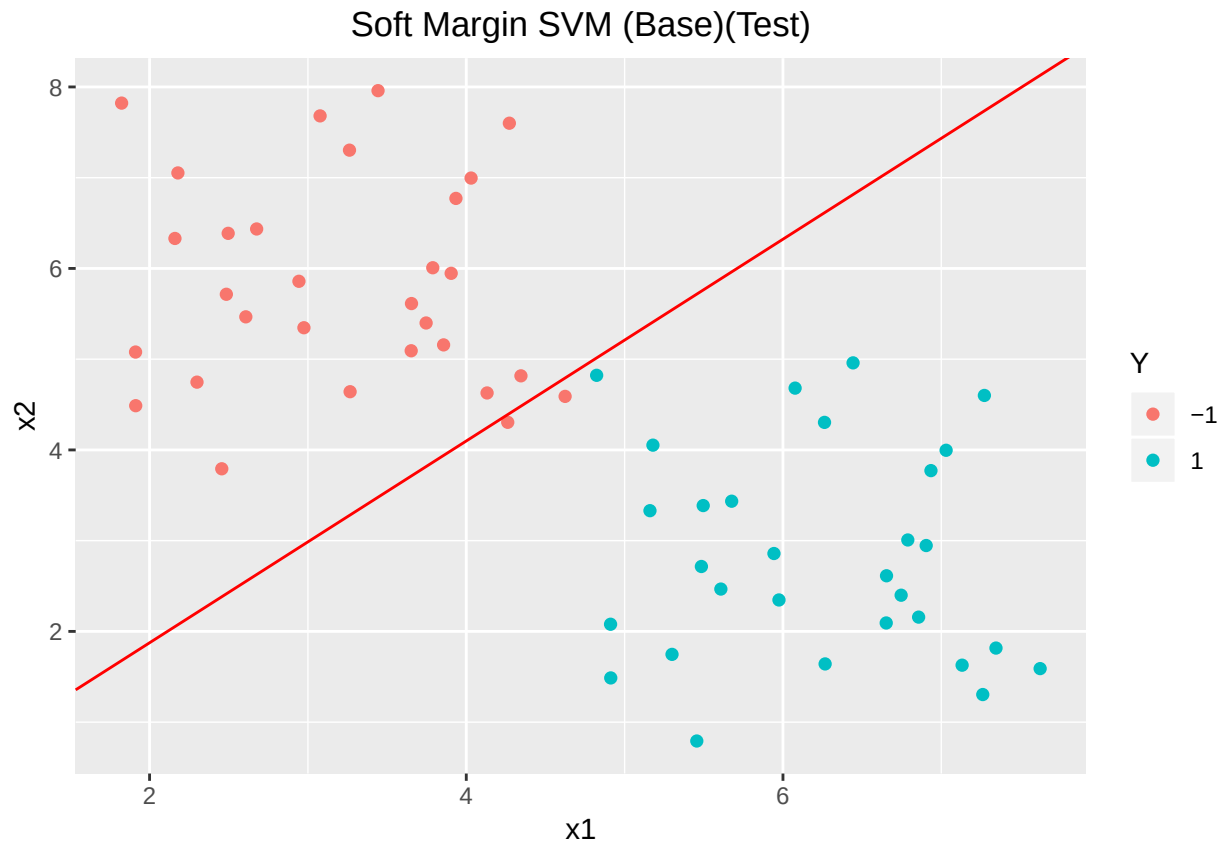


- 测试集检验

```
preY <- predictbase(data2.test, w_res, b_res)
```

```
##     TY
## preY -1  1
##   -1 28  0
##   1   2 30
```

```
ggplot(data = data2.test, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  geom_abline(slope = -w_res[1]/w_res[2], intercept = -b_res/w_res[2], colour = "red") +
  labs(title="Soft Margin SVM (Base)(Test)") + theme(plot.title = element_text(hjust = 0.5))
```



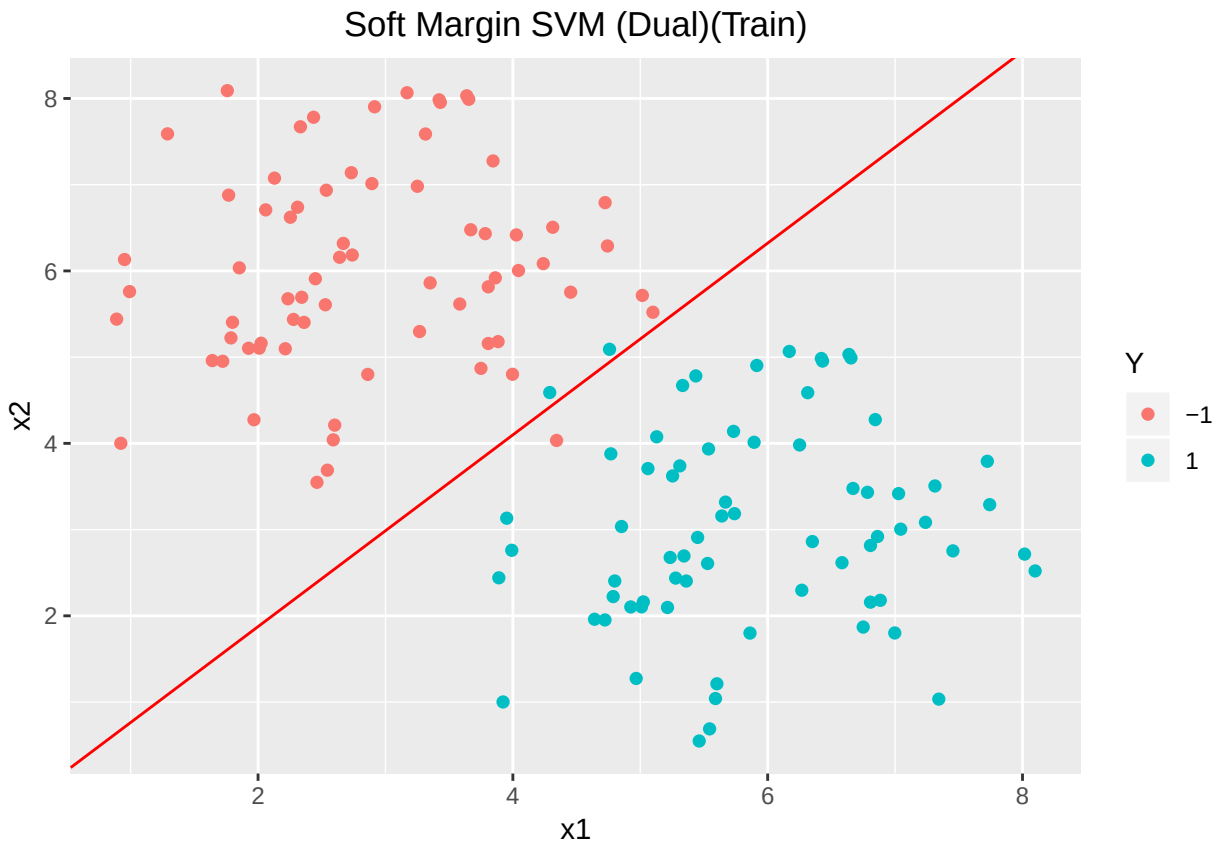
通过拉格朗日乘子法可将此问题转化为对偶问题

$$
\begin{aligned}
\underset{\alpha}{min} \quad & \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{m}\alpha_i \\
s.t. \quad & \sum_{i=1}^{m}\alpha_i y_i = 0 \\
& 0 \leqslant \alpha_i \leqslant C, \ \ i = 1, 2, ..., m
\end{aligned}
$$

- 编程实现

```
X <- as.matrix(data2.train[, 1:2])
Y <- as.matrix(as.numeric(as.character(data2.train[, 3])))

SVMSoftMarginDual <- function(X, Y, C = 1){
  n <- ncol(X)
  m <- nrow(Y)
  A <- Variable(m)

  objective <- Minimize(-sum(A) + 1/2 * quad_form(Y * A, X %*%t(X)))
  constraints <- list(A >= 0, A <= C, sum(Y * A) == 0)
```

```
  prob <- Problem(objective, constraints)

  # Problem solution
  solution <- solve(prob)
  A_res <- solution$getValue(A)

  w_res <- colSums(cbind(Y * A_res, Y * A_res) * X)
  # cal b_res(according to support vector)

  ind <- which(A_res > 0.00001)
  b_res <- min(1 - (Y[ind, 1])*(X[ind, ]%*%w_res))

  return(list(w_res = w_res, b_res = b_res))
}
modelSoftMDual <- SVMSoftMarginDual(X, Y)
w_res <- modelSoftMDual$w_res
b_res <- modelSoftMDual$b_res

ggplot(data = data2.train, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  geom_abline(slope = -w_res[1]/w_res[2], intercept = -b_res/w_res[2], colour = "red") +
  labs(title="Soft Margin SVM (Dual)(Train)") + theme(plot.title = element_text(hjust = 0.5))
```



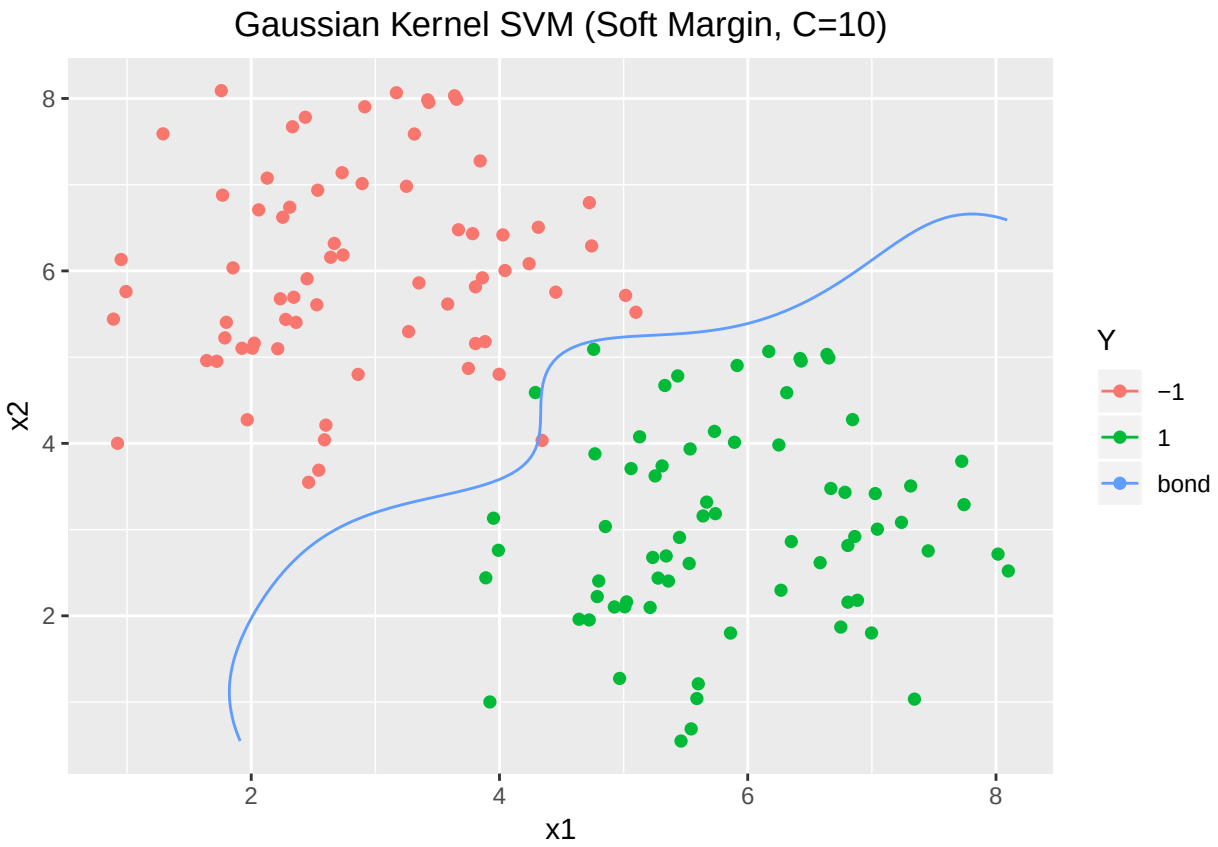
当然基于高斯核函数的支持向量机也可以加入软间隔降低模型过拟合的可能。

```
X <- as.matrix(data2.train[, 1:2])
Y <- as.matrix(as.numeric(as.character(data2.train[, 3])))

SVMGaussiankSoftMargin <- function(X, Y, Sigma = 1, C = 1){

  n <- ncol(X)
  m <- nrow(Y)
  A <- Variable(m)

  KernelM <- matrix(0, m, m)
  for(i in 1:m){
    for(j in 1:m){
      KernelM[i, j] <- GaussianK(X[i, ], X[j, ], Sigma)
    }
  }
  #
  objective <- Minimize(-sum(A) + 1/2 * quad_form(Y * A, KernelM))
  constraints <- list(A >= 0, A <= C, sum(Y * A) == 0)
```

```
  prob <- Problem(objective, constraints)

  # Problem solution
  solution <- solve(prob)
  A_res <- solution$getValue(A)

  # cal b_res(according to support vector)
  ind <- which(A_res > 0.00001)
  b_res <- min(1 - (Y[ind, 1])*
                colSums(matrix(rep(A_res * Y, length(ind)), ncol = length(ind))
                        * KernelM[, ind]))

  return(list(A_res = A_res, b_res = b_res, X = X, Y = Y, Sigma = Sigma))
}

modelGauss <- SVMGaussiankSoftMargin(X, Y, 1, 10)

PreGauss.train <- PredictGauss(data2.train, modelGauss)
# The Train confusion matrix
PreGauss.train$tb
```

```
##        newY
## preSign -1  1
##      -1 69  1
##      1   1 69
```

```
fx0 <- Wherefx0(modelGauss)

ggplot(data = data2.train, aes(x = x1, y = x2, colour = Y)) +
  geom_point(size = 2.0, shape = 16) +
  labs(title="Gaussian Kernel SVM (Soft Margin, C=10)") +
  theme(plot.title = element_text(hjust = 0.5)) +
  stat_contour(data = fx0, aes(x = x1, y = x2, z = Y, colour = "bond"), breaks=c(0))
```

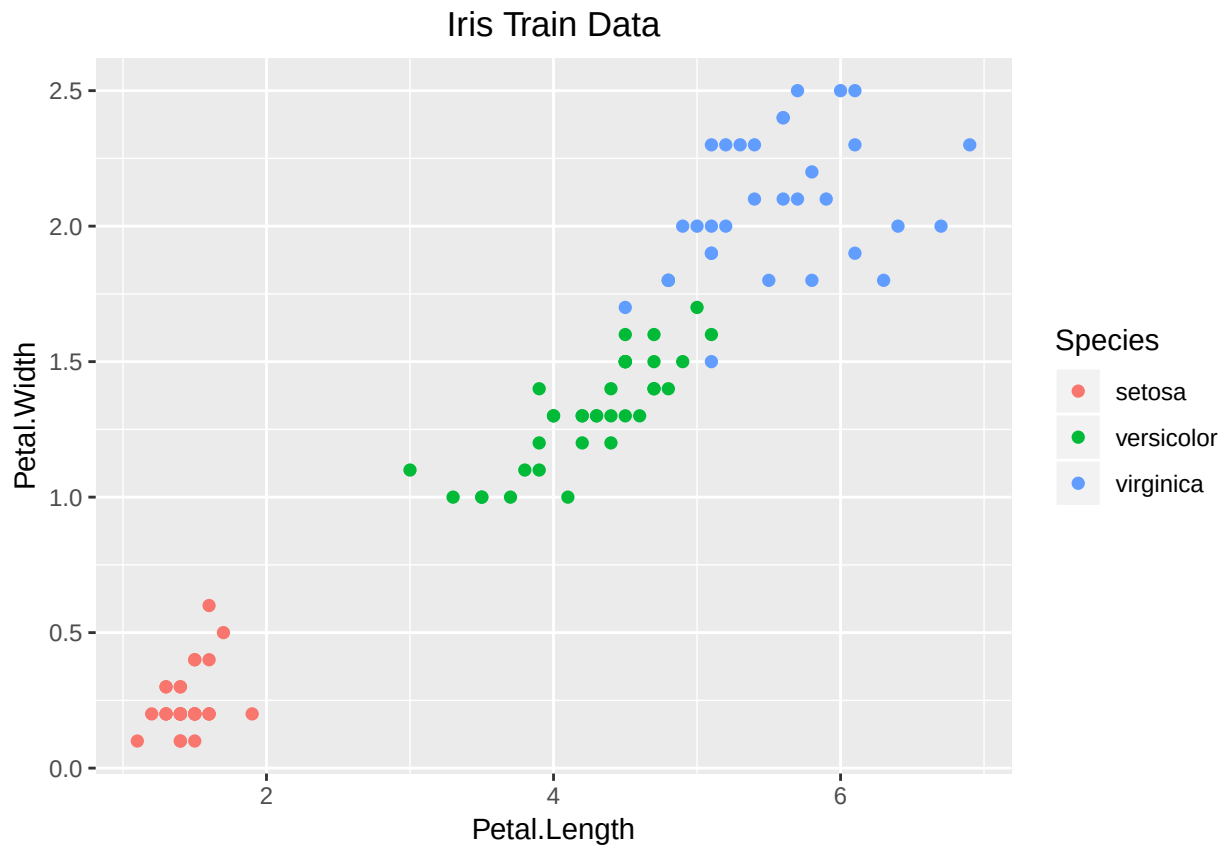


1.2.5 支持向量多分类

我们利用 OvR 训练 N 个分类器解决支持向量机多分类问题。OvR 每次将一个类的样本作为正例、所有其他类的样本作为反例来进行训练，训练完成后分别计算各分类器的 $f(x)$，将样本划分到 $f(x)$ 取最大值的类别。

- Iris 数据集

```
set.seed(1)
ind <- sample(1:150, 100)
iris.train <- iris[ind, c(3:5)]
iris.test <- iris[c(1:150)[-ind], c(3:5)]
ggplot(data = iris.train, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
  geom_point(size = 2.0, shape = 16) + labs(title="Iris Train Data") +
  theme(plot.title = element_text(hjust = 0.5))
```



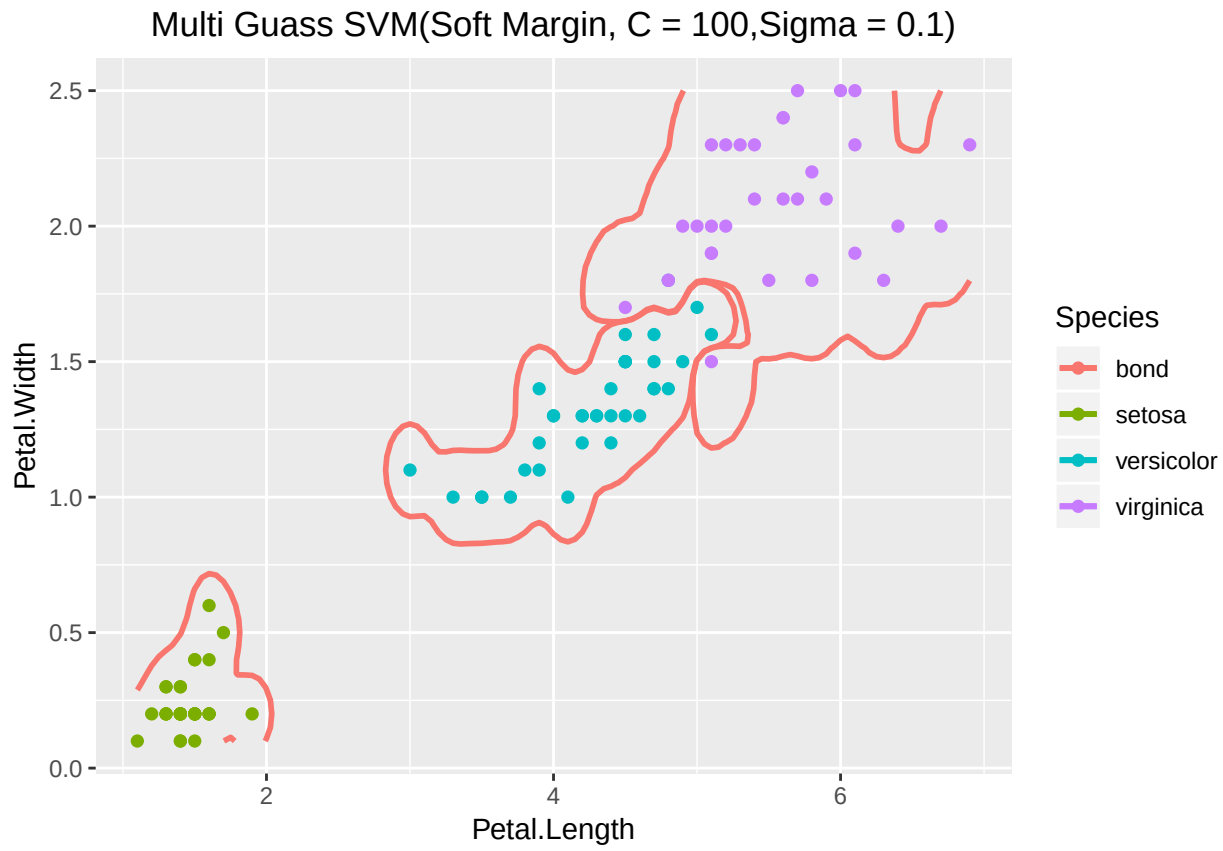
- 编程实现

```
MultiGuassSVM <- function(da, Sigma = 1, C = 10000){
  #
  m <- nrow(da)
  n <- ncol(da)
  #
  label <- levels(da[, ncol(da)])
  N <- length(label)
  X <- as.matrix(da[, 1:(n-1)])
  #
  fxdata <- data.frame(matrix(0, ncol = N, nrow = m))
  modellist <- list()
  for(i in 1:N){
    #
    labeli <- label[i]
    Yi <- ifelse(da[, ncol(da)] == labeli, 1, -1)
    Yi <- matrix(Yi, ncol = 1)
    dai <- da
```

```
    dai[, ncol(dai)] <- Yi

    modeli <- SVMGaussiankSoftMargin(X, Yi, Sigma, C)
    PreGaussi <- PredictGauss(dai, modeli)
    fxdata[, i] <- PreGaussi$preYvalue
    modellist[[i]] <- modeli
  }
  Presign <- apply(fxdata, 1, function(x){order(x, decreasing=T)[1]})
  Presign <- label[Presign]
  return(list(modellist = modellist, Presign = Presign, fxdata = fxdata,
              label = label, tb = table(Presign, True = da[, ncol(da)])))
}
modelMultSVM <- MultiGuassSVM(iris.train, Sigma = 0.1, C = 100)
modelMultSVM$tb
```

```
##             True
## Presign      setosa versicolor virginica
##   setosa         32          0         0
##   versicolor      0         35         0
##   virginica       0          1        32
```

```
# plot
fx01 <- Wherefx0(modelMultSVM$modellist[[1]])
fx02 <- Wherefx0(modelMultSVM$modellist[[2]])
fx03 <- Wherefx0(modelMultSVM$modellist[[3]])

ggplot(data = iris.train, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
  geom_point(size = 2.0, shape = 16) +
  labs(title="Multi Guass SVM(Soft Margin, C = 100,Sigma = 0.1)") +
  theme(plot.title = element_text(hjust = 0.5)) +
  stat_contour(data = fx01, aes(x = x1, y = x2, z = Y, colour = "bond"),
               breaks=c(0), size = 1) +
  stat_contour(data = fx02, aes(x = x1, y = x2, z = Y, colour = "bond"),
               breaks=c(0), size = 1) +
  stat_contour(data = fx03, aes(x = x1, y = x2, z = Y, colour = "bond"),
               breaks=c(0), size = 1)
```



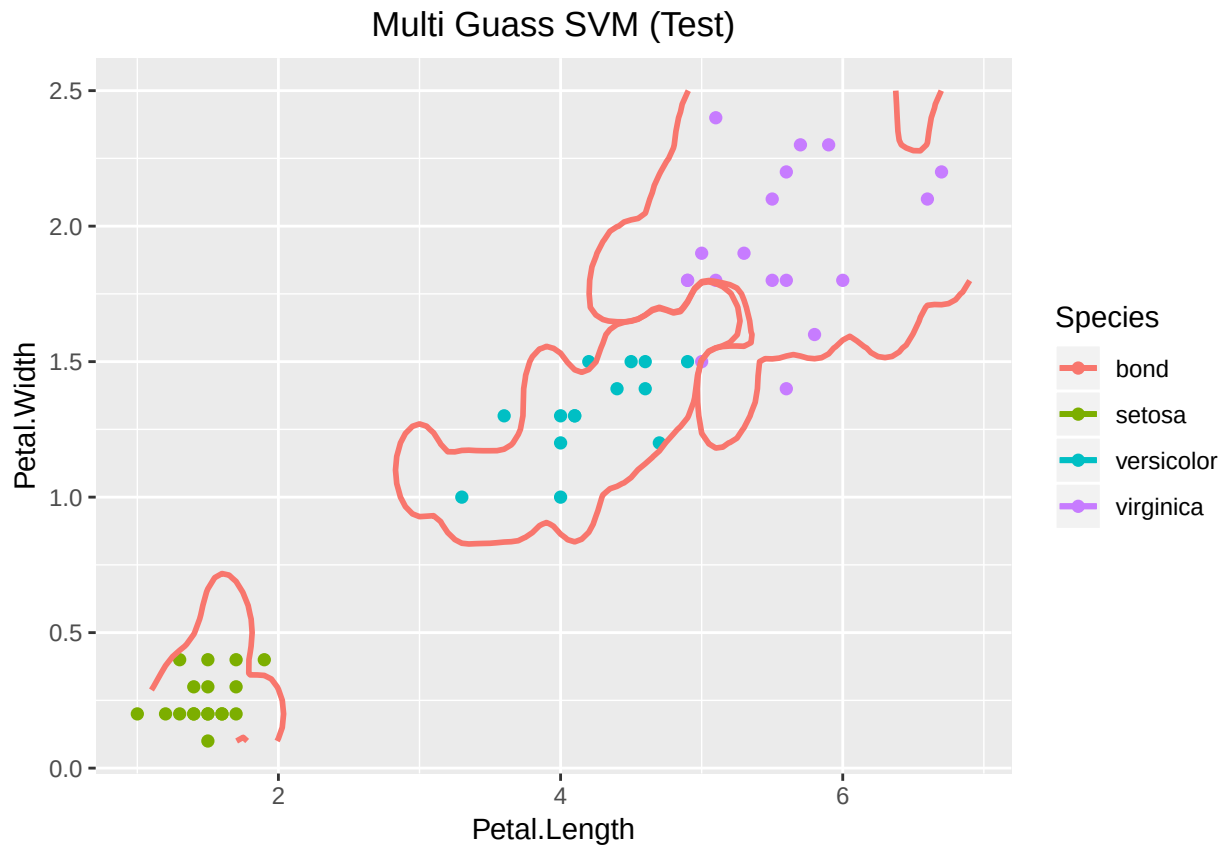
- 测试集检验

```
PredictMultiGuass <- function(da, modellist, label){
  #
    m <- nrow(da)
    n <- ncol(da)
  #
    N <- length(modellist)
    X <- as.matrix(da[, 1:(n-1)])
  #
    fxdata <- data.frame(matrix(0, ncol = N, nrow = m))
    for(i in 1:N){
      #
      modeli <- modellist[[i]]
      PreGaussi <- PredictGauss(da, modeli)
      fxdata[, i] <- PreGaussi$preYvalue
    }
    Presign <- apply(fxdata, 1, function(x){order(x, decreasing=T)[1]})
    Presign <- label[Presign]
```

```
  return(list(Presign = Presign, tb = table(Presign, True = da[, ncol(da)])))
}

PredMultiGUass <- PredictMultiGuass(iris.test, modelMultSVM$modellist, modelMultSVM$label)
PredMultiGUass$tb
```

```
##             True
## Presign      setosa versicolor virginica
##   setosa         17          0         0
##   versicolor      0         13         0
##   virginica       1          1        18
```

```
# plot
ggplot(data = iris.test, aes(x = Petal.Length, y = Petal.Width, colour = Species)) +
  geom_point(size = 2.0, shape = 16) +
  labs(title="Multi Guass SVM (Test)") +
  theme(plot.title = element_text(hjust = 0.5)) +
  stat_contour(data = fx01, aes(x = x1, y = x2, z = Y, colour = "bond"),
               breaks=c(0), size = 1) +
  stat_contour(data = fx02, aes(x = x1, y = x2, z = Y, colour = "bond"),
               breaks=c(0), size = 1) +
  stat_contour(data = fx03, aes(x = x1, y = x2, z = Y, colour = "bond"),
               breaks=c(0), size = 1)
```



1.2.6 支持向量机回归

支持向量回归（Support Vector Regression，简称 SVR）与传统模型不同的是，它假设我们能容忍 $f(x)$ 与 y 之间最多有 ϵ 的偏差，即仅当 $f(x)$ 与 y 之间的绝对偏差大于 ϵ 时才计算损失。

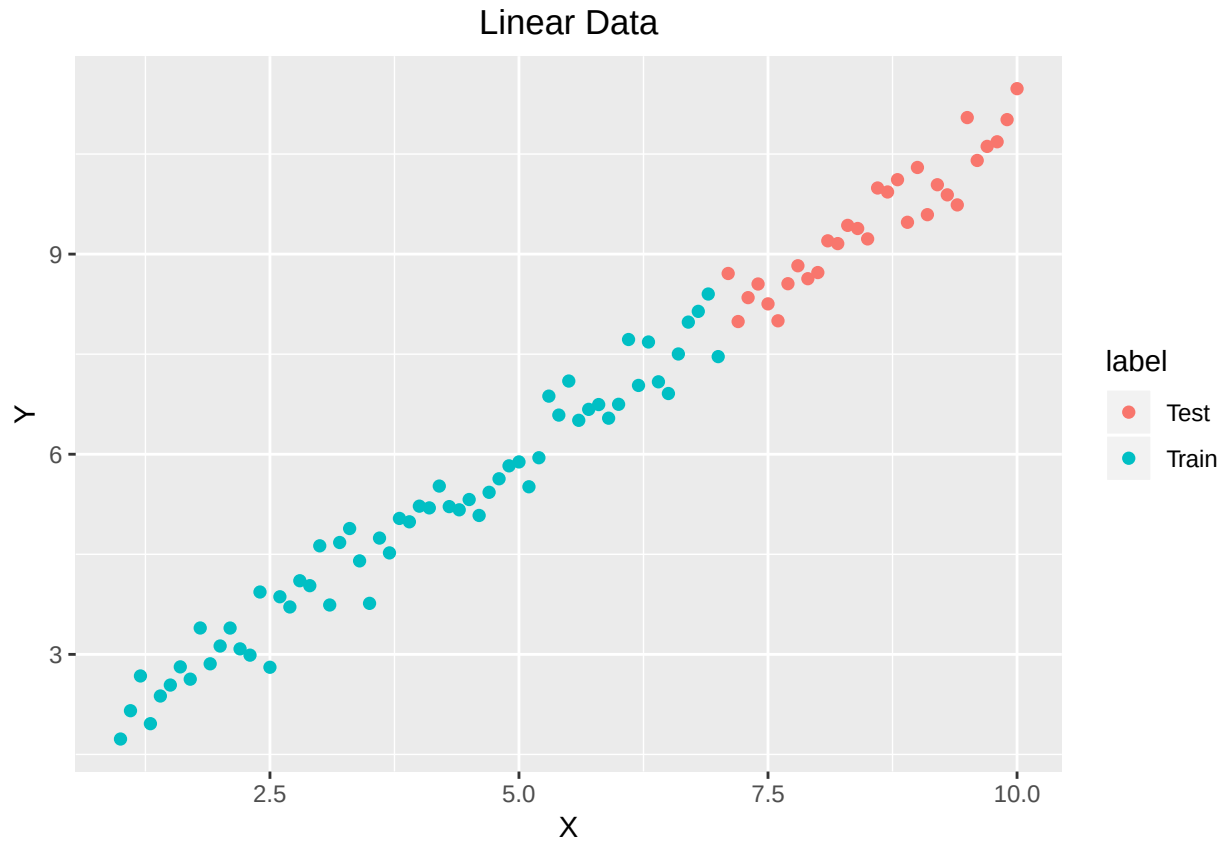
支持向量机回归的优化问题可以用下式表达

$$
\begin{aligned}
\underset{w,b,\xi_i^1,\xi_i^2}{min} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}(\xi_i^1 + \xi_i^2) \\
s.t. \quad & f(x_i) - y_i \leqslant \epsilon + \xi_i^1 \\
& y_i - f(x_i) \leqslant \epsilon + \xi_i^2 \\
& \xi_i^1 \geqslant 0, \xi_i^2 \geqslant 0, \ \ i = 1, 2, ..., m
\end{aligned}
$$

- 编程实现

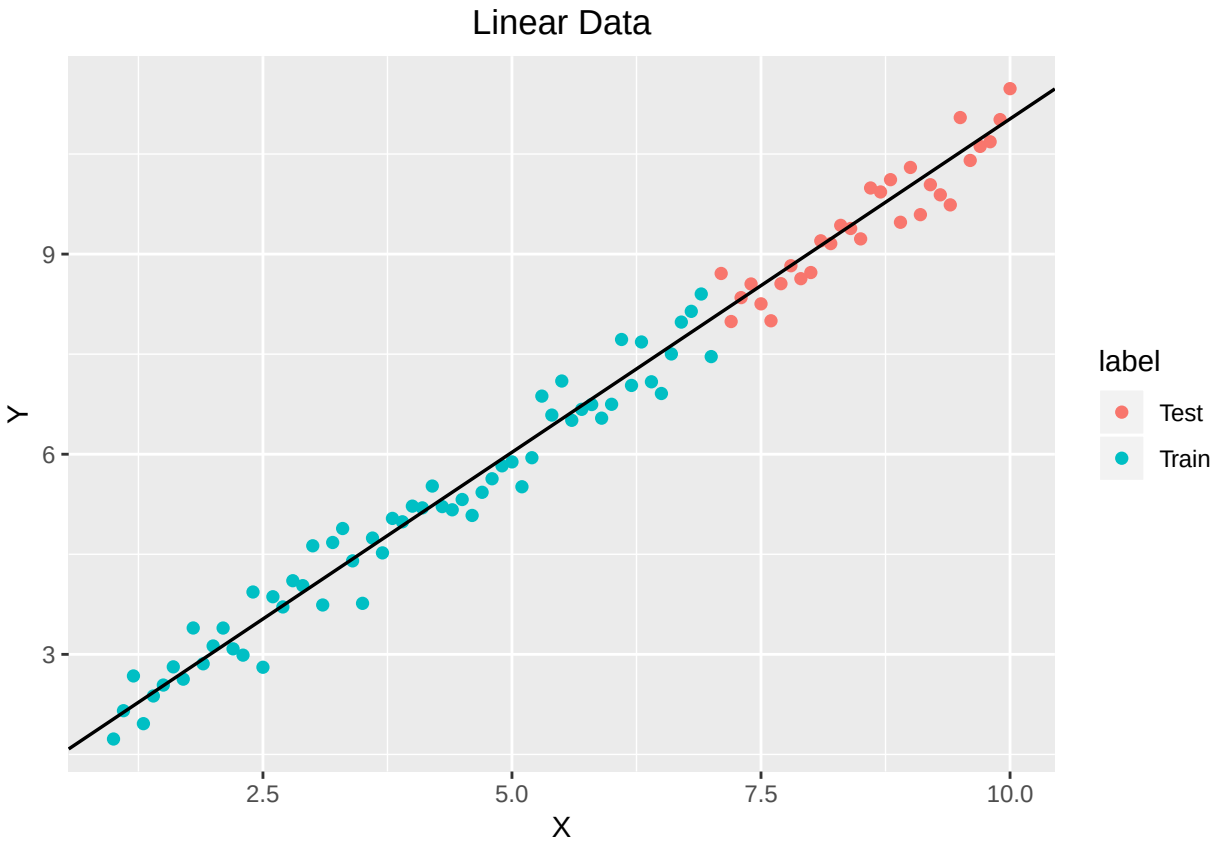
```
X <- seq(1, 10, 0.1)
set.seed(2)
Y <- X + 1 + rnorm(length(X), 0, 0.3)
linerda <- data.frame(X = X, Y = Y, label = c(rep("Train", 61), rep("Test", 30)))
linerda.train <- linerda[1:61, 1:2]
```

```
linerda.test <- linerda[62:91, 1:2]

ggplot(data = linerda, aes(x = X, y = Y, colour = label)) +
  geom_point(size = 2, shape = 16) + labs(title = "Linear Data") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
SVRbase <- function(da, epsilon = 0.3, C = 1){
  #
  n <- ncol(da)
  m <- nrow(da)
  X <- as.matrix(da[, 1:(n-1)], nrow = m)
  Y <- as.matrix(da[, n], nrow = m)
  #
  w <- Variable(n-1)
  b <- Variable(1)
  kexi1 <- Variable(m)
  kexi2 <- Variable(m)
  #
  objective <- Minimize(0.5 * norm(w, "F")^2 + C * sum(kexi1 + kexi2))
```

```
  constraints <- list(X %*% w + b - Y <= epsilon + kexi1,
                      Y - (X %*% w + b) <= epsilon + kexi2,
                      kexi1 >= 0, kexi2 >= 0)
  prob <- Problem(objective, constraints)

  # Problem solution
  solution <- solve(prob)
  w_res <- solution$getValue(w)
  b_res <- solution$getValue(b)
  return(list(w_res = w_res, b_res = b_res))
}

modelSVRbase <- SVRbase(linerda.train)
w_res <- modelSVRbase$w_res
b_res <- modelSVRbase$b_res

ggplot(data = linerda, aes(x = X, y = Y, colour = label)) +
  geom_point(size = 2, shape = 16) + labs(title = "Linear Data") +
  theme(plot.title = element_text(hjust = 0.5)) +
  geom_abline(slope = w_res, intercept = b_res, size = 0.6)
```



- 预测

```
predictbaseSVR <- function(da, w_res, b_res){
  #
  n <- ncol(da)
  m <- nrow(da)
  X <- as.matrix(da[, 1:(n-1)], nrow = m)
  Y <- as.matrix(da[, n], nrow = m)
  #
  PreY <- X %*% w_res + b_res
  # sum of squared errors
  SSE <- sum((PreY - Y)^2)
  return(list(PreY = PreY, SSE = SSE))
}

Prelinertest <- predictbaseSVR(linerda.test, w_res, b_res)
print(paste("SSE = ", round(Prelinertest$SSE,3)))
```

```
## [1] "SSE =  3.292"
```

1.2.7 小结

本次大部分内容都是代码实现，并没有太多理论的推到。原因是我对于对偶问题的理解还不是很深刻，对于 KKT 条件更是知之甚少。我打算在后续学完凸优化课程后再重新推导 SVM 的理论部分。

在编程实现的过程中有两点经验值得记录：

- 偏移量 b 的计算

当求解原始问题时，b 是优化参数，直接利用 `CVXR` 中的函数 `getValue(b)` 就可以求得；当求解对偶问题时，利用 α_i 不等于 0 判断出一个支持向量，既可以利用公式求解；当求解软间隔对偶问题时，$\alpha_i \neq 0$ 也能是错分样本，这个时候没有办法先判断哪些是支持向量，需计算 `min(1 - (Y[ind, 1])*(X[ind, ]%*%w_res))` 求得 b（ind 代表 α_i 不为 0 的索引）。

- 高斯核画图

利用等高线 0 表示分类边界，`stat_contour` 可实现此功能。