서포트벡터머신(Support Vector Machine)

http://contents.kocw.net/KOCW/document/2016/chungbuk/leegeonmyeong/11.pdf

서포트벡터머신(Support Vector Machine)는 Corres와 Vapnik에 의해서 1995년에 제안되었다.
서포트벡터머신은 서포트 벡터 분류기를 확장하여 비선형 클래스 경계를 수용할 수 있도록 개발한 분류 방법.

초평면(Hyperplane)
최대마진분류기(Maximum Margin Classifier): 데이터가 있을 때, 이것을 곡선이 아닌 직선이나 평면으로 구별하는 방법
초평면(Hyperplane): 최대 마진 분류기가 경계로 사용하는 선이나 면

분리 초평면(Separting Hyperplane): 데이터를 완벽하게 분리하는 초평면

마진(Margin): 데이터와 초평면의 수직 거리(가장 짧은 거리)
최대마진 초평면(Maximal Margin Hyperplane): 마진이 가장 큰 초평면
최대마진분류기(Maximum Margin Classifier): 데이터가 초평면에 의해 가장 잘 분류
서포트 벡터(Support Vector): 양쪽 데이터의 경계값을 포함하는 초평면 상에 위치한 데이터

대부분의 경우 분리 초평면이 존재하지 않을 수도 있고, 최대 마진 분류기 또한 존재할 수 없는 경우가 많다. 이런 문제의 해결을 위하여 데이터를 분류할 때, 약간의 오차를 허용한다.
이 때 약간의 오차를 소프트 마진(Soft margin)이라고 한다.

서포트 벡터 분류기(Support Vector Classifier): 소프트마진을 이용하여 데이터를 분류하는 것

서포트 벡터 분류기는 최대 마진 분류기를 확장한 것으로 몇몇 관측치를 희생하더라도 나머지 관측치를 더 잘 분류할 수 있는 방법

코스트(cost): 허용하는 오류의 정도
R에서는 tune.svm함수를 이용하여 코스트 값을 계산

커널 트릭을 이용하여 분류를 할 때 결정되어야 할 파라미터:
코스트(cost): 오차 허용 정도
감마(Gamma): 커널과 관련된 파라미터

실습

```
======================
# SVM
#

credit1 <- read.csv("credit.csv", header=TRUE)
str(credit1)

# Creditability 컬럼의 0과 1의 숫자를 팩터형 1과 2로 바꿈
credit1$Creditability <- as.factor(credit1$Creditability)
str(credit)

# 학습데이터와 테스트데이터 구성
library(caret)

set.seed(1234)

trData <- createDataPartition(y = credit1$Creditability, p=0.7, list=FALSE)
head(trData)
train <- credit1[trData,]
test <- credit1[-trData,]

str(train)

# 파라미터 설정

install.packages("e1071")
library("e1071")

# radial 커널 사용 튜닝
result1 <- tune.svm(Creditability~., data=train, gamma=2^(-5:0), cost = 2^(0:4), kernel="radial")

# linear 커널 사용 튜닝
result2 <- tune.svm(Creditability~., data=train, cost = 2^(0:4), kernel="linear")

# polynomial 커널 사용 튜닝
result3 <- tune.svm(Creditability~., data=train, cost = 2^(0:4), degree=2:4, kernel="polynomial")
```

```
# 튜닝된 파라미터 확인
result1$best.parameters
result2$best.parameters
result3$best.parameters

# SVM 실행
normal_svm1 <- svm(Creditability~., data=train, gamma=0.0625, cost=1, kernel = "radial")
normal_svm2 <- svm(Creditability~., data=train, cost=1, kernel="linear")
normal_svm3 <- svm(Creditability~., data=train, cost=1, degree=3, kernel = "polynomial")

# 결과 확인
summary(normal_svm1)
summary(normal_svm2)
summary(normal_svm3)

# sv index 확인
normal_svm1$index
normal_svm2$index
normal_svm3$index

# SVM으로 예측
normal_svm1_predict <- predict(normal_svm1, test)
str(normal_svm1_predict)

normal_svm2_predict <- predict(normal_svm2, test)
str(normal_svm2_predict)

normal_svm3_predict <- predict(normal_svm3, test)
str(normal_svm3_predict)

# radial kernel 적용 시 Confusion Matrix 구성 및 Statistics
confusionMatrix(normal_svm1_predict, test$Creditability)

# linear kernel 적용 시 Confusion Matrix 구성 및 Statistics
confusionMatrix(normal_svm2_predict, test$Creditability)

# polynomial kernel 적용 시 Confusion Matrix 구성 및 Statistics
confusionMatrix(normal_svm3_predict, test$Creditability)
```

iris 데이터 대상으로 예측

```
install.packages("kernlab")
library(kernlab)
model1 <- ksvm(Species~., data=iris)
iris_predicted <- predict(model1, newdata=iris)
table(iris_predicted, iris$Species)
```

==========

```
> credit1 <- read.csv("credit.csv", header=TRUE)
> credit1$Creditability <- as.factor(credit1$Creditability)
> str(credit)
'data.frame':   1000 obs. of  21 variables:
 $ Creditability                   : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Account.Balance                 : int  1 1 2 1 1 1 1 1 4 2 ...
 $ Duration.of.Credit..month.      : int  18 9 12 12 12 10 8 6 18 24 ...
 $ Payment.Status.of.Previous.Credit: int  4 4 2 4 4 4 4 4 4 2 ...
 $ Purpose                         : int  2 0 9 0 0 0 0 0 3 3 ...
 $ Credit.Amount                   : int  1049 2799 841 2122 2171 2241 3398 1361 1098 3758 ...
 $ Value.Savings.Stocks            : int  1 1 2 1 1 1 1 1 1 3 ...
 $ Length.of.current.employment    : int  2 3 4 3 3 2 4 2 1 1 ...
 $ Instalment.per.cent             : int  4 2 2 3 4 1 1 2 4 1 ...
 $ Sex...Marital.Status            : int  2 3 2 3 3 3 3 3 2 2 ...
 $ Guarantors                      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Duration.in.Current.address     : int  4 2 4 2 4 3 4 4 4 4 ...
 $ Most.valuable.available.asset   : int  2 1 1 1 2 1 1 1 3 4 ...
 $ Age..years.                     : int  21 36 23 39 38 48 39 40 65 23 ...
 $ Concurrent.Credits              : int  3 3 3 3 1 3 3 3 3 3 ...
 $ Type.of.apartment               : int  1 1 1 1 2 1 2 2 2 1 ...
 $ No.of.Credits.at.this.Bank      : int  1 2 1 2 2 2 2 1 2 1 ...
 $ Occupation                      : int  3 3 2 2 2 2 2 2 1 1 ...
 $ No.of.dependents                : int  1 2 1 2 1 2 1 2 1 1 ...
 $ Telephone                       : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Foreign.Worker                  : int  1 1 1 2 2 2 2 2 1 1 ...
>
```

```
> result1$best.parameters
   gamma cost
8 0.0625    2
> result2$best.parameters
  cost
1    1
> result3$best.parameters
  degree cost
2      3    1
>
```

```
> summary(normal_svm1)

Call:
svm(formula = Creditability ~ ., data = train, gamma = 0.0625, cost = 1, kernel = "radial")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  479

 ( 277 202 )


Number of Classes:  2

Levels:
 0 1
```

```
> normal_svm1_predict <- predict(normal_svm1, test)
> str(normal_svm1_predict)
 Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 2 ...
 - attr(*, "names")= chr [1:300] "5" "8" "9" "10" ...
>
> normal_svm2_predict <- predict(normal_svm2, test)
> str(normal_svm1_predict)
 Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 2 ...
 - attr(*, "names")= chr [1:300] "5" "8" "9" "10" ...
>
> normal_svm3_predict <- predict(normal_svm3, test)
> str(normal_svm1_predict)
 Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 1 2 ...
 - attr(*, "names")= chr [1:300] "5" "8" "9" "10" ...
>
```

```
> confusionMatrix(normal_svm1_predict, test$Creditability)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  29  11
         1  61 199

               Accuracy : 0.76
                 95% CI : (0.7076, 0.8072)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.01249

                  Kappa : 0.3208

 Mcnemar's Test P-Value : 7.709e-09

            Sensitivity : 0.32222
            Specificity : 0.94762
         Pos Pred Value : 0.72500
         Neg Pred Value : 0.76538
             Prevalence : 0.30000
         Detection Rate : 0.09667
   Detection Prevalence : 0.13333
      Balanced Accuracy : 0.63492

       'Positive' Class : 0

>
```

Iris 데이터 예측 결과

```
> library(kernlab)
> model1 <- ksvm(Species~., data=iris)
> iris_predicted <- predict(model1, newdata=iris)
> table(iris_predicted, iris$Species)

iris_predicted setosa versicolor virginica
    setosa         50          0         0
    versicolor      0         48         2
    virginica       0          2        48
> |
```