Homework #7 **20180047 김경수**

0. 이론적 배경

(1) Support vector classifier

최대 마진 분류기를 통해 분류할 수 없는 경우에 대해, 두 클래스를 완벽하게 분류하지 않는 hyperplane에 기초한 보다 일반적인 분류기이다. 이는 가장 큰 margin을 찾는 대신에, 일부 관측치들은 margin의 옳지 않는 쪽에 있거나 심지어는 hyperplane의 옳지 않은 쪽에 있을 수 있도록 허용된다. 따라서 이를 soft margin classifier라고 부르기도 한다.

support vector classifier에서 C(cost, 조율 파라미터)의 선택이 중요한데, 이는 k-fold CV를 통해 일반적으로 선택된다. C가 작을 경우, 좁은 margin을 찾으려고 하기에, support vector들의 수가 감소한다. 따라서 bias는 낮지만 variance는 증가한다. 반대로 C가 증가할 경우, 넓은 margin을 찾으려고 하기에, support vector들의 수가 증가한다. 따라서 bias는 크지만, variance는 감소한다.

(2) Support vector machine(SVN)

(1)의 support vector classifier를 비선형적인 결정경계를 제공하는 것으로 확장한 것이 SVN이다. kernel를 이용하여 변수공간을 확장한 결과이다. Kernel은 관측치들의 similarity를 수량화 하는 함수인데, linear kernel, polynomial kernel, radial kernel 등을 data들의 특성에 맞게 선택하여 사용하여야 한다. 본 레포트에서는 10-fold CV를 이용하여 최고의 모델을 선택하고(parameter들을 선택하고), 이를 검정셋에 대해 적용하여 linear kernel, polynomial kernel, radial kernel의 성능을 비교하였다

***데이터 전처리에 관한 코드이다.**

```
clin <- readRDS("clinical.rds")
gex <- readRDS("expression.rds")
#clinical과 expression.rds 데이터들을 위와 같이 implement함.
idx <- intersect(colnames(gex), clin$sample)
clin_p <- clin[clin$sample_id %in% idx, ]
stage <- as.vector(clin_p$stage)
stage_p <- rep(0,length(stage))
for (i in 1:length(stage)){
  if(stage[i]=='stage i'){stage_p[i]=1}
  else if(stage[i]=='stage ii'){stage_p[i]=2}
  else if(stage[i]=='stage iii'){stage_p[i]=3}
  else if(stage[i]=='stage iv'){stage_p[i]=4}
  else if(stage[i]=='stage v'){stage_p[i]=5}
}
gex_p <- gex[, colnames(gex) %in% idx]
```

```
gex_p <- na.omit(gex_p)
```
#Subtype이 NA인 sample을 제외하고, 두 데이터셋이 공통적으로 가지는 sample_id 들에 대해서만 데이터를 남김. 편의를 위해 stage_p 벡터를 정의하여 각 stage를 integer에 매칭시킴.
```
library(preprocessCore)
temp <- normalize.quantiles(gex_p)
colnames(temp) <- colnames(gex_p)
rownames(temp) <- rownames(gex_p)
gex_p <- temp
data <- data.frame(stage_p, t(gex_p))
```
#Quantile Normalization 진행.
```
set.seed(1)
stage2.sam <- sample(rownames(data)[data$stage_p == 2],
size=min(sum(data$stage_p==1)))
stage3.sam <- sample(rownames(data)[data$stage_p == 3],
size=min(sum(data$stage_p==1)))
data <- data[rownames(data) %in% c(rownames(data)[data$stage_p == 1], stage2.sam,
stage3.sam), ]
```
#Undersampling을 진행한다. Stage_1에 속하는 data의 개수가 89이고, 이것이 최소이기 때문에 이에 맞춰 undersampling을 진행한다.
```
feature.size <- 10
aov.gene.pval <- c()
for(i in 2:(ncol(data))){
  aov.gene.pval <- c(aov.gene.pval, summary(aov(data$stage_p ~ data[,
i]))[[1]][["Pr(>F)"]][1])
}
aov.idx <- order(aov.gene.pval, decreasing=FALSE)[1:feature.size]
data.aov <- data[, c(1, 1+aov.idx)]
```
#ANOVA test를 통해 stage에 따른 가장 큰 변화를 보이는 top 10 gene을 select하고, 이에 대해 집중한다.
```
set.seed(1)
data.aov$stage_p <- as.factor(data.aov$stage_p)
partition <- createDataPartition(data.aov$stage_p, p=0.8, list=FALSE)
train <- data.aov[partition,]
test <- data.aov[-partition,]
```
#교차검증을 위해 dataset seperation을 진행한다.

HW 7

1. Build svm model changing kernel function to classify patient stage. And compare their performance.
2. In case of radial kernel model, find its optimal cost & gamma parameter. Write its mathematical description and relation with model's bias-variance.

1. Procedure & R code

(1) linear kernel을 이용한 SVM의 성능을 검사한다. 먼저 best model(best parameter)를 찾는다.

R code :

```
linear.tune <- tune(svm, stage_p~., data = train, kernel = "linear",
                    ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10)))
linear.model <- linear.tune$best.model
summary(linear.model)
```

(2) best model을 사용하여 test set에 대해 분류를 진행하고 그 성능을 확인한다.

R code :

```
linear.pred <- predict(linear.model, test)
confusionMatrix(linear.pred, test$stage_p)
mean(linear.pred == test$stage_p)
```

(3) polynomial kernel, radial kernel에 대해서도 (1)~(2)와 유사하게 SVM의 성능을 검사한다. 코드는 아래와 같다.

R code :

```
poly.tune <- tune(svm, stage_p~., data = train, kernel = "polynomial",
                  ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10), gamma =
c(0.01,0.5,1,2)))
poly.model <- poly.tune$best.model
summary(poly.model)
poly.pred <- predict(poly.model, test)
confusionMatrix(poly.pred, test$stage_p)
mean(poly.pred == test$stage_p)

rad.tune <- tune(svm, stage_p~., data = train, kernel = "radial",
                 ranges = list(cost = c(0.001, 0.01, 0.1, 1, 10, 100), gamma =
c(0.001, 0.005, 0.01 ,0.5))
summary(rad.tune)
rad.model <- rad.tune$best.model
summary(rad.model)
rad.pred <- predict(rad.model, test)
```

```
confusionMatrix(rad.pred, test$stage_p)
mean(rad.pred == test$stage_p)
```

2. Results

(1) Linear kernel

Cost : 1 인 경우에 교차검증 오차율이 가장 낮으며, 이것이 best parameter인 것을 확인할 수 있었다. 전체 51개의 test set에 대하여 41.18 %의 데이터가 올바르게 분류되었으며, 결과는 다음과 같다.

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1
```

Prediction \ Reference	1	2	3
1	12	8	4
2	4	4	8
3	1	5	5

(2) Polynomial kernel

Cost : 0.01, degree : 3 인 경우에 교차검증 오차율이 가장 낮으며, 이것이 best parameter인 것을 확인할 수 있었다. 전체 51개의 test set에 대하여 29.41 %의 데이터가 올바르게 분류되었으며, 결과는 다음과 같다.

```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
       cost:  0.01
     degree:  3
     coef.0:  0
```

Prediction \ Reference	1	2	3
1	6	4	2
2	6	3	9
3	5	10	6

(3) Radial kernel

Cost : 10, gamma : 0.01 인 경우에 교차검증 오차율이 가장 낮으며, 이것이 best parameter인 것을 확인할 수 있었다. 전체 51개의 test set에 대하여 49.02 %의 데이터가 올바르게 분류되었으며, 결과는 다음과 같다.

```
          Reference
Prediction  1  2  3
         1 12  8  4
         2  3  4  4
         3  2  5  9
Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
 cost gamma
   10  0.01
```

따라서 가장 높은 분류율을 보인 radial kernel을 적용한 SVM model을 select하는 것이 나을 것이다. linear, radial과 비교하였을 때 polynomial kernel을 적용하였을 때 performance가 비교적 좋지 않았다. 이는 실제로 선형결계를 가지고 있을 것이라 유추할 수 있다. Radial kernel이 좋은 performance를 보이는 이유는 raidal kernel이 매우 locally하게 작용하기 때문이다.

Radial kernel을 이용한 SVM의 경우 교차검증을 통해 설정된 optimal cost의 값은 10이고, gamma는 0.01이다.

먼저 cost(C)의 대해 논해보자. 다음은 linear SVM model을 selection하는 최적화 문제에 대한 해이다. 단지 radial model의 경우 kernel function만 변화하는 것이기에 C에 관한 내용은 일맥상통하다.

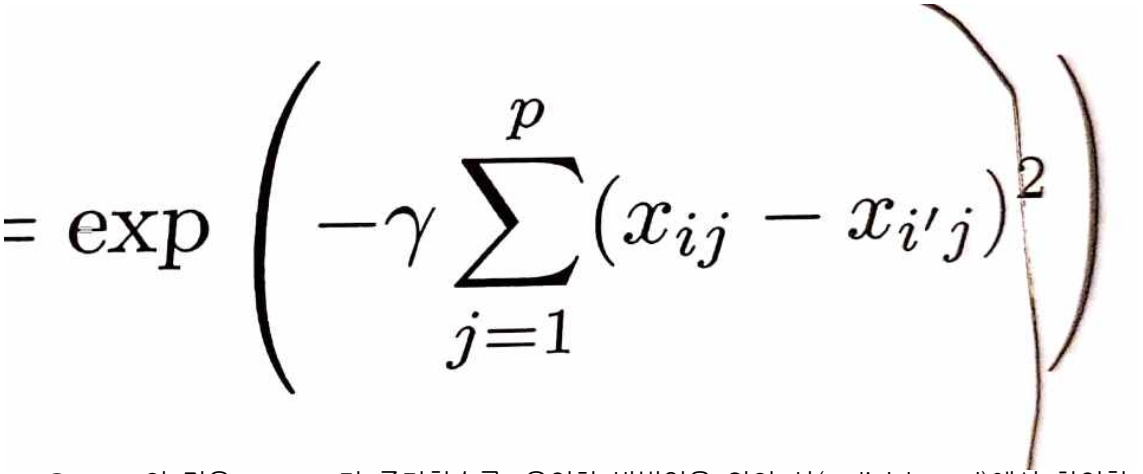
$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

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

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

앞에 말한 것처럼, C가 작을 경우, 좁은 margin(M)을 찾으려고 하기에, support vector들의 수가 감소한다. 따라서 bias는 낮지만 variance는 증가한다. 반대로 C가 증가할 경우, 넓은 margin을 찾으려고 하기에, support vector들의 수가 증가한다. 따라서 bias는 크지만, variance는 감소한다.



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

Gamma의 경우 gamma가 증가할수록, 유연한 방법임을 위의 식(radial kernel)에서 확인할 수 있다. 또한 gamma는 시그마제곱에 반비례한다. 따라서 검정 데이터에 대해, bias가 증가하지만 variance는 감소한다. 반대로, gamma가 감소할수록 bias는 감소하지만 variance는 증가한다. 따라서 cost처럼 gamma value 또한 bias-variance tradeoff 사이에서 절충하여 gamma값을 구하는 것이 중요하다.