# Homework5

*Kai Wang*

*3/21/2018*

## 1 Hoeffding's inequality

*a*

First, we apply Chernoff's Bounds to the random variable $\sum_{i=1}^{n}(X_i - \mu_{X_i})$,

$$Pr(\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_{X_i}) \geq t) \leq \min_{\lambda \geq 0} \mathbb{E}[e^{\lambda(\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_{X_i})}]e^{-\lambda t}$$

Since $X_i$'s are independent, we have

$$\mathbb{E}[e^{\lambda(\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_{X_i}))}]e^{-\lambda t} = e^{-\lambda t}\prod_{i=1}^{n}\mathbb{E}[e^{\frac{\lambda}{n}(X_i - \mu_X)}]$$

By Hoeffding's Lemma, we have

$$Pr(\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_{X_i}) \geq t) \leq \min_{\lambda \geq 0} e^{-\lambda t}e^{\frac{\lambda^2(b-a)^2}{8n^2}} = \min_{\lambda \geq 0} e^{\frac{\lambda^2(b-a)^2 - 8n^2\lambda t}{8n^2}}$$

Since $f(x) = e^x$ is a convex function, we transform this problem to

$$\min_{\lambda \geq 0} \frac{\lambda^2(b-a)^2 - 8n^2\lambda t}{8n^2}$$

.

The minima exists when $\lambda = \frac{4n^2t}{(b-a)^2}$. Hence, we have

$$Pr(\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_{X_i}) \geq t) \leq \exp(\frac{-2n^2t}{(b-a)^2})$$

for all $t \geq 0$.

*b*

$X_i \sim$ Binomial$(n, p)$. Let's take $n = 2, p = 0.4, t = 1$ for simplicity. $\mathbb{E}(X_i) = np = 50$.

$$Pr(\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu_{X_i}) \geq t) \leq \exp(\frac{-2n^2t}{(b-a)^2}) = e^{-2}$$

The real distribution is $Pr(\frac{1}{2}\sum_{i=1}^{2}(X_i - 0.8) \geq 1) = P(X_1 = X_2 = 2) = (C_2^2 0.4^2)^2 = 0.0256 < e^{-2}$. So this bound is tighter than hoeffdings bound.

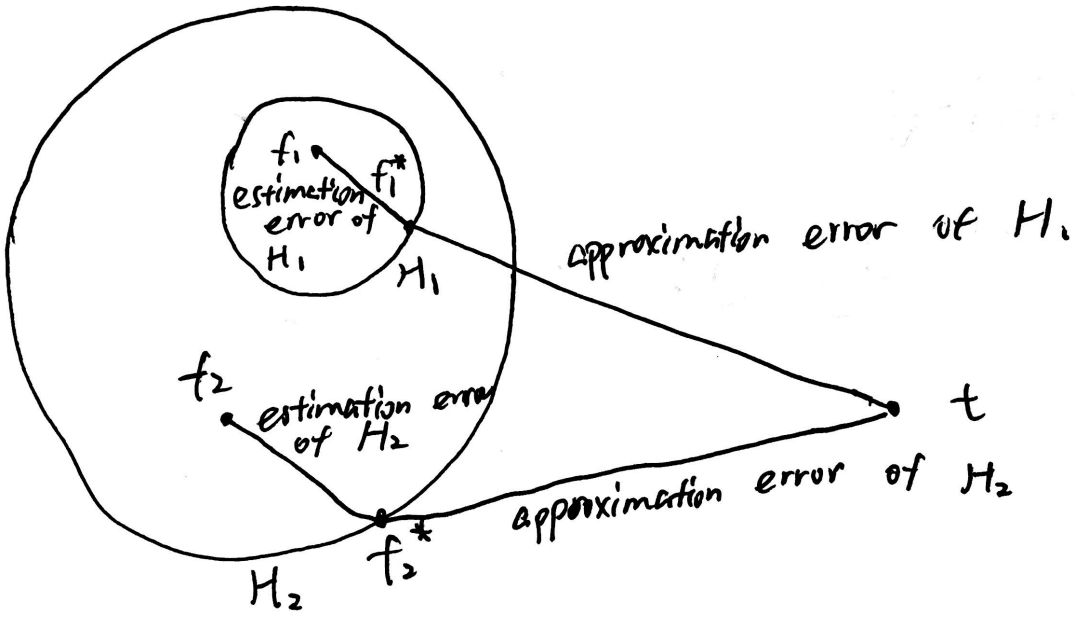



Figure 1:

## 2 VC Dimension

*a*

$\mathcal{H}_1$ is a half-space in p-dimension, so the VC dimension of $\mathcal{H}_1$ $p+1$.

$< u, v >_{\mathcal{H}_2} = (1 + < u, v >)^2 = (1 + u_1v_1 + u_2v_2 + \cdots + u_pv_p)^2$.

The VC-dimension of $\mathcal{H}_2$ is equivalent to choose 2 from $p+2$, which is $\frac{(p+1)(p+2)}{2}$.

*b*

Hypothesis space $\mathcal{H}_2$ should incluse $\mathcal{H}_1$.

Approximation error: $R^{true}f_1^* - R^* = inf_{\mathcal{H}_1} R^{true}f - inf_f R^{true}f$.

Estimation error: $R^{true}\hat{f}_1 - R^{true}f^*$.

And $R^{emp}f = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{[f(x_i)\neq y_i]}$, and $\hat{f}_1 \in \arg\min_{f\in\mathcal{H}_1} R^{emp}f$.

Similarly, we have:

Approximation error: $R^{true}f_2^* - R^* = inf_{\mathcal{H}_2} R^{true}f - inf_f R^{true}f$.

Estimation error: $R^{true}\hat{f}_1 - R^{true}f^*$.

And $R^{emp}f = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{[f(x_i)\neq y_i]}$, and $\hat{f}_2 \in \arg\min_{f\in\mathcal{H}_2} R^{emp}f$.

*c*

The cosine function class with single parameter, such as $\cos x$, has infinite VC-dimension.

## 3 Ridge Regression

*a* We have

$$\hat{\beta}^{ridge} = \arg\min_{\beta \in \mathbb{R}^p} \sum_{i=1}^{n} ||\mathbf{y} - \mathbf{X}\beta||_{\mathbf{2}}^{\mathbf{2}} + \lambda ||\beta||_{\mathbf{2}}^{\mathbf{2}}$$

Hence, we can write the loss function as $L(\beta) = (\mathbf{y} - \mathbf{X}\beta)^{\mathbf{T}}(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^{\mathbf{T}}\beta$. We need to minimize the loss function with respect to $\beta$. So we have

$$\begin{aligned} \frac{\partial L(\beta)}{\partial \beta} &= -2\mathbf{X}^{\mathbf{T}}(\mathbf{y} - \mathbf{X}\beta) + \mathbf{2}\lambda\beta = \mathbf{0} \\ &\rightarrow \mathbf{X}^{\mathbf{T}}\mathbf{y} + \mathbf{X}^{\mathbf{T}}\mathbf{X}\beta + \lambda\beta = \mathbf{0} \\ &\rightarrow (\mathbf{X}^{\mathbf{T}}\mathbf{X} + \lambda\mathbf{I})\beta = \mathbf{X}^{\mathbf{T}}\mathbf{y} \\ &\rightarrow \beta = (\mathbf{X}^{\mathbf{T}}\mathbf{X} + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{X}^{\mathbf{T}}\mathbf{y} \end{aligned}$$

So the closed form solution is $\hat{\beta}^{ridge} = (\mathbf{X}^{\mathbf{T}}\mathbf{X} + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{X}^{\mathbf{T}}\mathbf{y}$.

*b*

```
X = read_csv("X.dat", col_names = FALSE) %>% as.matrix()
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
beta1_star = c(0.1, 0.3, 0.2, 0.2, 0.9, 0.8, 0.9, 0.1, 0.4, 0.2, 0.7, 0.3, 0.1, 0.7, 0.8, 0.3, 0.2, 0.8
beta2_star = c(0.5, 0.6, 0.7, 0.9, 0.9, 0.8, 0.9, 0.8, 0.6, 0.5, 0.7, 0.6, 0.7, 0.7, 0.8, 0.8, 0.9, 0.8
```

```
ridge = function(beta, X){
  MSE = matrix(NA, 100,6)
  mu = X %*% beta
  for(i in 1:100){
    for(j in 1:6){
      error = rnorm(50,0,1)
      y = mu + error
      fit = glmnet(X, y, alpha = 0, lambda = j-1)
      y.pred = predict(fit, newx = X)
      MSE[i,j] = mean((y.pred - y)^2)
    }
  }
  return(MSE)
}
```

```
MSE1 = ridge(beta1_star, X)
MSE2 = ridge(beta2_star, X)

df = rbind(colMeans(MSE1),colMeans(MSE2))
colnames(df) = c("linear regression","lambda=1","lambda=2","lambda=3","lambda=4","lambda=5")
rownames(df) = c("beta1_star", "beta2_star")
kable(df)
```

| | linear regression | lambda=1 | lambda=2 | lambda=3 | lambda=4 | lambda=5 |
|---|---|---|---|---|---|---|
| beta1_star | 0.6015877 | 1.048089 | 1.603995 | 2.123422 | 2.533700 | 2.891672 |
| beta2_star | 0.6021323 | 1.194427 | 2.008598 | 2.733895 | 3.424388 | 4.051433 |

We can see the simple linear model has the best MSE among all models. This is likely because our data was generated by $\mathbf{y} = \mathbf{X}\beta^* + \epsilon$, which is a simple linear model. When we use ridge regression to fit this simulated data, the larger we penalize the number of parameters, the further we are from the true model, which contains all 50 parameters. So we can see as $\lambda$ increases, estimated MSE of ridge regression fit increases.

*c*

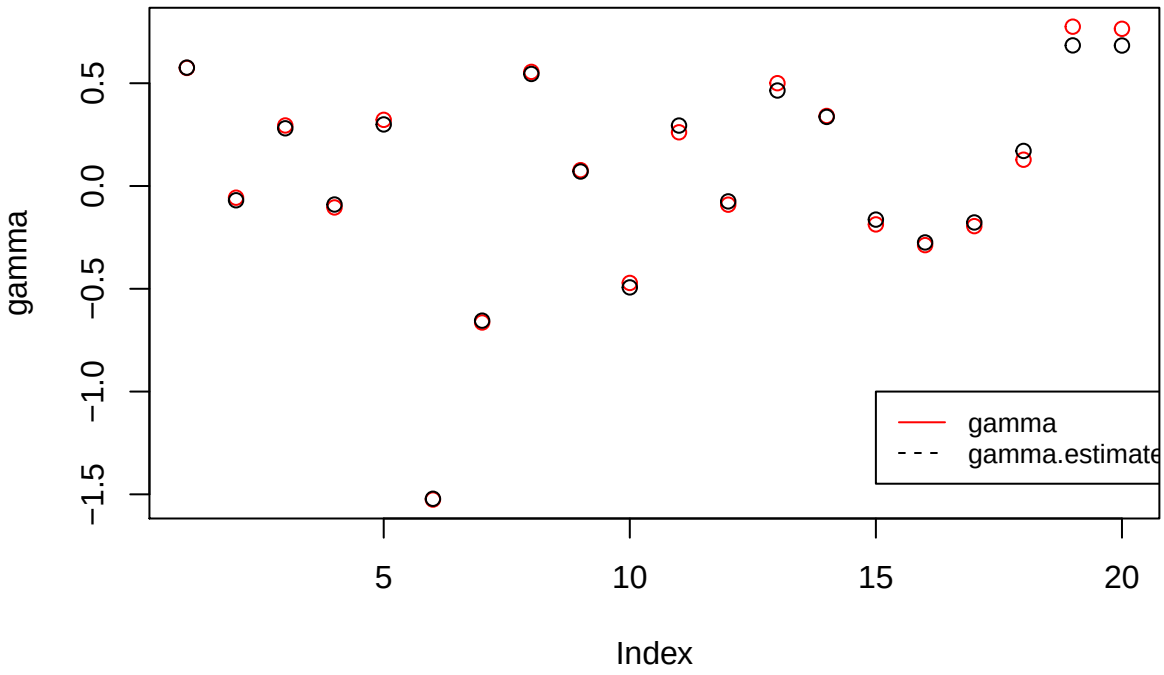
$$\begin{aligned}\hat{\beta} &= (\mathbf{X^T X} + \lambda \mathbf{1_p})^{-1}\mathbf{X^T Y} \\ &= (\mathbf{V L^2 V^T} + \lambda \mathbf{1_p})^{-1}\mathbf{V L U^T Y} \\ &= (\mathbf{V}(\mathbf{L^2} + \lambda \mathbf{1_p})\mathbf{V^T})^{-1}\mathbf{V L U^T Y} \\ &= \mathbf{V}(\mathbf{L^2} + \lambda \mathbf{1_p})^{-1}\mathbf{V^T V L U^T Y} \\ &= \mathbf{V}(\mathbf{L^2} + \lambda \mathbf{1_p})^{-1}\mathbf{L U^T Y}\end{aligned}$$

Since $\gamma = \mathbf{V^T}\beta$, we have $\hat{\gamma} = \mathbf{V^T V}(\mathbf{L^2} + \lambda \mathbf{1_p})^{-1}\mathbf{L U^T Y} = (\mathbf{L^2} + \lambda \mathbf{1_p})^{-1}\mathbf{L U^T Y}$

```
#center and scale X
X = scale(X, center = TRUE, scale = TRUE)
X.svd = svd(X)
L = diag(X.svd$d)
U = X.svd$u
V = X.svd$v
gamma = t(V) %*% beta1_star
```

```
mu = X %*% beta1_star
gamma.hat = matrix(NA, 100, 20)
for(i in 1:100){
  Y = mu + rnorm(50,0,1)
  gamma.hat[i,] = solve(L %*% L + diag(1, 20, 20)) %*% L %*% t(U) %*% Y
}
```

```
gamma.estimate = colMeans(gamma.hat)
plot(gamma, col="red")
points(gamma.estimate)
legend(15, -1,legend=c("gamma", "gamma.estimate"),
       col=c("red", "black"), lty=1:2, cex=0.8)
```

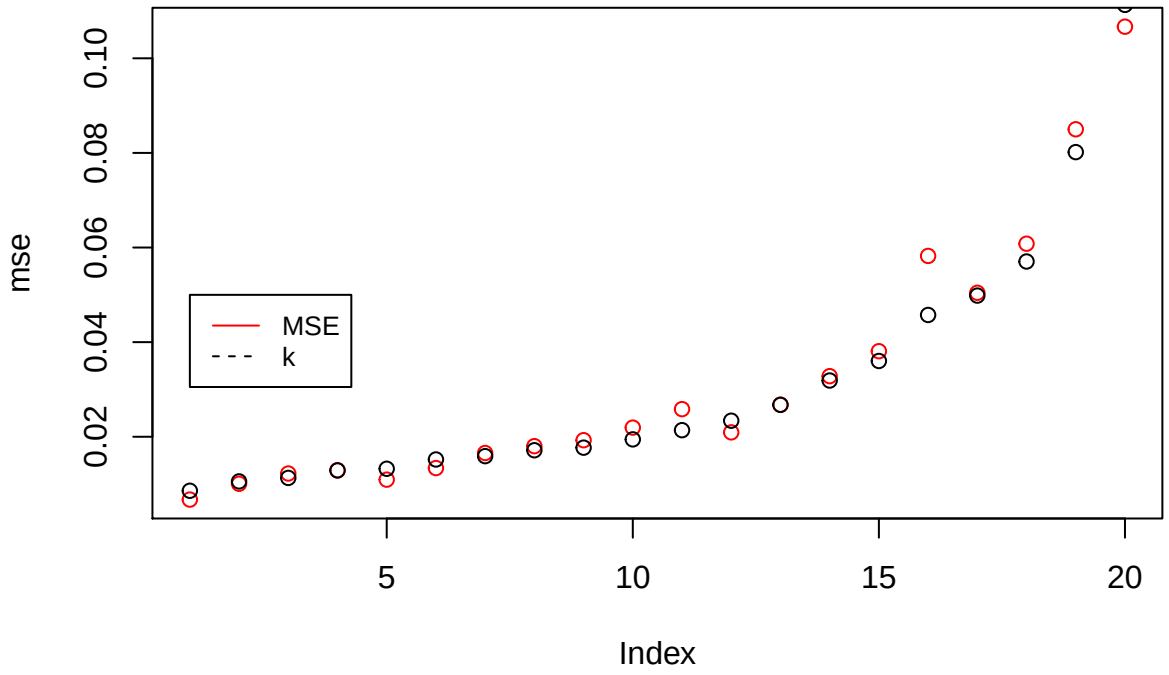



Let $k_i = \frac{l_i^2+\gamma_i^2}{(l_i^2+\lambda)^2}$.

```
mse = rep(NA, 20)
for(i in 1:20){
  mse[i] = mean((gamma.hat[,i] - gamma[i])^2)
}

k = rep(NA, 20)
for(i in 1:20){
  k[i] = (diag(L)[i]^2+gamma[i]^2)/(diag(L)[i]^2+1)^2
}

plot(mse, col="red")
points(k)
legend(1, 0.05,legend=c("MSE", "k"),
       col=c("red", "black"), lty=1:2, cex=0.8)
```


mse
0.02
0.04
0.06
0.08
0.10
MSE
k
5
10
15
20
Index