Homework 1

Guozhen Li
STA 208 - Statistical Machine Learning

April 15, 2018

1 Exercise 1

1.1 Predictor minimizing true risk

The true risk is

$$R(g) = \mathbb{E}[\ell(Y, g(X))]$$

By conditioning on X, we can write $R(g)$ as

$$R(g) = \mathbb{E}_X \ell(Y, g(X)) \cdot Pr(Y|X)$$

Minimize $R(g)$ pointwise:

$$\hat{g}(x) = \arg\min_{h \in \{0,1\}} \ell(Y, h) Pr(Y|X = x)$$

With $\ell()$ being the Hamming loss function this simplifies to:

$$\hat{g}(x) = \arg\max_{h \in \{0,1\}} Pr(Y = h|X = x)$$

* Reference: ESL pg. 20

1.2 True risk of Bayes classifier

???

1.3 Classify with a single x_j

To fit a model like $h(x) = 1\{x_j > 0\}$, we can try all possible j's and find the one that gives smallest emperical risk. Pseudo code for this algorithm as follows:

```
best_R = Inf
best_j = None
for j in [1, 2, ..., p]:
```

```
            pred = [1 if x[i,j]>0 else 0 for i in 1:N]
            loss = xor(pred, y)
            risk = sum(loss)/N
            if risk < best_R:
                    best_R = risk
                    best_j = j
    return best_j
```

This algorithm returns the best j.

When making a prediction based on a x_{new}, simply check the value of its jth component, and if $x_{new,j} > 0$, predict 1, otherwise predict 0.

1.4 Number of samples needed

???

2 Exercise 2

2.1 $\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$

In linear regression, we already know that

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\beta} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})_{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Here if we make $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})_{-1}\boldsymbol{X}^T$, then $\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$

In the case of kNN model, we can make a matrix $\boldsymbol{H}$ such that:

$$H_{i,j} = \begin{cases} \frac{1}{k}, & \text{if } x_j \in N_k(x_i) \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i, j \in 1, 2, ..., N$$

Here $N_k(x_i)$ is the k-nearest neighborhood of observation x_i (x_i itself included). $\boldsymbol{H}$ is a $N \times N$ matrix, and each row of it only has k elements with value $\frac{1}{k}$, and all other elements are 0. With such contruction, $\hat{\boldsymbol{y}} = \boldsymbol{H}\boldsymbol{y}$.

2.2 kNN leave-one-out

To leave-one-out, we need to reconstruct the $\boldsymbol{H}$ matrix, taking the sample left out into account. Assume the i_0th observation needs to be left out, make $\boldsymbol{H}$ such that

$$H_{i,j} = \begin{cases} \frac{1}{k}, & \text{if } x_j \in N_k(x_i) \text{ and } i \neq i_0 \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i, j \in 1, 2, ..., N$$

Here $N_k(x_i)$ should be the set of x_i's k nearest neighbors among all x's but excluding x_{i_0}.

Denote the $\boldsymbol{H}$ matrix for $\boldsymbol{X}$ with the ith observation left out as $\boldsymbol{H}_i$. The square error with the ith observation left out would be

$$\begin{aligned} e_i &= (\hat{\boldsymbol{y}} - \boldsymbol{y})^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) \\ &= (\boldsymbol{H}_i\boldsymbol{y} - \boldsymbol{y})^T(\boldsymbol{H}_i\boldsymbol{y} - \boldsymbol{y}) \\ &= \boldsymbol{y}^T(\boldsymbol{H}_i - \boldsymbol{I})^T(\boldsymbol{H}_i - \boldsymbol{I})\boldsymbol{y} \end{aligned}$$

Leave-one-out cross validated square error would be

$$\begin{aligned} SE &= \frac{1}{N}\sum_{i=1}^{N} e_i \\ &= \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}^T(\boldsymbol{H}_i - \boldsymbol{I})^T(\boldsymbol{H}_i - \boldsymbol{I})\boldsymbol{y} \end{aligned}$$

2.3 SVD in regression

Plug $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$ into linear regression model $\boldsymbol{Y} = \boldsymbol{X}\beta + \boldsymbol{e}$, we get

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T\beta + \boldsymbol{e}$$

Introduce a new vector $b = \boldsymbol{D}\boldsymbol{V}^T\beta$, the linear model becomes

$$\boldsymbol{Y} = \boldsymbol{U}b + \boldsymbol{e}$$

This takes the form of our normal linear regression model, and b can be estimated by

$$\hat{b} = (\boldsymbol{U}^T\boldsymbol{U})^{-1}\boldsymbol{U}^T\boldsymbol{y}$$

Because $\boldsymbol{U}$ is an orthogonal matrix, $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$, $\hat{b}$ simplifies to

$$\hat{b} = \boldsymbol{U}^T\boldsymbol{y}$$

And $\hat{\beta}$ can be estimated by

$$\hat{\beta} = (\boldsymbol{V}^T)^{-1}\boldsymbol{D}^{-1}\hat{b}$$

Again, because $\boldsymbol{V}$ is also an orthogonal matrix, $(\boldsymbol{V}^T)^{-1} = \boldsymbol{V}$, thus

$$\hat{\beta} = \boldsymbol{V}\boldsymbol{D}^{-1}\hat{b}$$

Define $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}^{-1}$, we get an estimator for β and

$$\hat{\beta} = \boldsymbol{A}\hat{b}$$

To fit the model on a dataset, first performe SVD on the design matrix $\boldsymbol{X}$, then calculate $\boldsymbol{D}^{-1}$ by replacing every diagonal element with its inverse (because $\boldsymbol{D}$ is diagonal, all other places shoudl be 0). Then multiply $\boldsymbol{U}$ and $\boldsymbol{y}$ to get $\hat{b}$. Finally calculate $\hat{\beta}$ via $\hat{\beta} = \boldsymbol{A}\hat{b}$.

Reference: Mandel, John. "Use of the singular value decomposition in regression analysis." The American Statistician 36.1 (1982): 15-24.

2.4 Change rank

Take the first r elements of $\hat{b}$ to be $\hat{b}_r$, take the first r columns of $\boldsymbol{A}$ to be $\boldsymbol{A}_r$, then $\hat{\beta}_r = \boldsymbol{A}_r\hat{b}_r$.

If $\boldsymbol{A}$ and $\hat{b}$ already exist, this computation should take $O(1)$ time, because it is just a slicing of existing vector/matrix.