STAT 425

# Variable Selection

# Variable Selection

- Consider a MLR model $Y \sim 1 + X_1 + X_2 + \cdots + X_p$ where we have $p$ non-intercept predictors. Later we drop the intercept for notational simplicity.
- In many applications, we have a lot of potential explanatory variables, i.e., $p$ is large and we could even have $p >> n$, but only a small portion of the $p$ variables are believed to be relevant to $Y$.
- Of interest is to find the following subset of the p predictors:

$$S = \{j : \beta_j \neq 0\}$$

- In some applications as sales prediction, the key question we need to answer is to identify this set $S$, e.g., which variables among the $p$ variables are really effective for boosting the sales ($Y$).
- If our goal is simply to do well on prediction, then should we care about variable selection?

  Recall that the LS estimate $\hat{\boldsymbol{\beta}}$ is unbiased, i.e., estimates for irrelevant $\hat{\beta}_j$ (with $j \in S^c$) will eventually go to zero anyway. To understand this, let's examine the training and the test errors.

# Test Error vs. Training Error

- Training data: $(\mathbf{x}_i, y_i)_{i=1}^n$
- Test data: $(\mathbf{x}_i, y_i^*)_{i=1}^n$ is an independent (imaginary) data set collected at the same location $\mathbf{x}_i$'s (also known as, in-sample prediction)
- Assume the data comes from a linear model:
  $\mathbf{y}_{n\times 1}$, $\mathbf{y}^*_{n\times 1}$ are i.i.d $\sim N_n(\boldsymbol{\mu}, \sigma^2\mathbf{I}_n)$ and $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$
- We can also write:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$
$$\mathbf{y}^* = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}^*$$

with $\mathbf{e}_{n\times 1}$, $\mathbf{e}^*_{n\times 1}$ $\quad i.i.d \sim N_n(\mathbf{0}, \sigma^2\mathbf{I}_n)$ are independent.

# Mean square testing error and Mean square training error

$$
\begin{aligned}
E[\text{Test Error}] &= E||\mathbf{y}^* - \mathbf{X}\hat{\boldsymbol{\beta}}||^2 \\
&= E||(\mathbf{y}^* - \mathbf{X}\boldsymbol{\beta}) + (\mathbf{X}\boldsymbol{\beta} - \mathbf{X}\hat{\boldsymbol{\beta}})||^2 \\
&= E||\mathbf{y}^* - \boldsymbol{\mu}||^2 + E||\mathbf{X}\boldsymbol{\beta} - \mathbf{X}\hat{\boldsymbol{\beta}}||^2 \\
&= E||\mathbf{e}^*||^2 + Tr(\mathbf{X}Cov(\hat{\boldsymbol{\beta}})\mathbf{X}^\top) \\
&= n \cdot \sigma^2 + \sigma^2 Tr\mathbf{H} = n \cdot \sigma^2 + p \cdot \sigma^2
\end{aligned}
$$

$$
\begin{aligned}
E[\text{Train Error}] &= E||\mathbf{y} - \hat{\mathbf{y}}||^2 = E||(\mathbf{I} - \mathbf{H})\mathbf{y}||^2 \\
&= Tr((\mathbf{I} - \mathbf{H})Cov(\mathbf{y})(\mathbf{I} - \mathbf{H})^\top) \\
&= \sigma^2 Tr((\mathbf{I} - \mathbf{H})) = (n - p) \cdot \sigma^2
\end{aligned}
$$

From the previous equations we can conclude:

- Testing error increases with $p$
- Training error decreases with $p$
- When adding more variables ($p$ large) the testing error increases. If our goal is pure prediction, adding more variables to matrix $\mathbf{X}$ is not the best option. We should remove some irrelevant variables.
- The analysis on the previous slide might indicate that the best model (i.e., the one with the smallest expected test error), is the intercept-only model with $p = 0$.
- This of course is not true. The previous analysis is based on the assumption that the mean of $\mathbf{y}$ is in the column space of $\mathbf{X}$, i.e., there exists some coefficient vector $\boldsymbol{\beta}$ such that $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$. In general, we run a linear regression model using only a subset of the columns of $\mathbf{X}$. This means there will be an additional Bias term.

# Model Index $\boldsymbol{\gamma}$

- Index each model (i.e., each subset of the p variables) by a $p$-dimensional binary vector $\boldsymbol{\gamma}$:

$$\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_p), \quad \gamma_j = 0/1$$

  where $\gamma_j = 1$ indicates that $X_j$ is included in the model, and $\gamma_j = 0$ otherwise.
- So there are a total of $2^p$ possible subsets or sub-models. In particular

$$\boldsymbol{\gamma} = (1, 1, \ldots, 1)$$

  refers to the full model including all $p$ variables (largest dim), and

$$\boldsymbol{\gamma} = (0, 0, \ldots, 0)$$

  refers to the intercept-only model (smallest dim).

Suppose that $\boldsymbol{\mu} = \mathbf{X}\boldsymbol{\beta}$ where $\boldsymbol{\mu}$ is the mean of $\mathbf{y}$. If we fit the data $\mathbf{y}$ with respect to model $\boldsymbol{\gamma}$, i.e., we fit a linear model with a sub-design matrix $X_{\boldsymbol{\gamma}}$ where $X_{\boldsymbol{\gamma}}$ contains only columns from $\mathbf{X}$ such that $\gamma_j = 1$. We can show that the Testing Error and the Training error for model $\boldsymbol{\gamma}$ are:

$$E[\text{Test Error}] = n\sigma^2 + p\sigma^2 + Bias_{\boldsymbol{\gamma}}$$
$$E[\text{Training Error}] = n\sigma^2 - p\sigma^2 + Bias_{\boldsymbol{\gamma}}$$

Bigger model (i.e., $p$ large) $\rightarrow$ small Bias, but large variance ($p\sigma^2$); Smaller model (i.e., $p$ small) $\rightarrow$ large Bias, but small variance ($p\sigma^2$). So to reduce the test error (i.e., prediction error), the key is to find the best trade-off between Bias and Variance.

# Model selection procedures

- Testing-based procedures: Select best model based on statistical tests for model comparison.
- Criterion-based procedures: Select best model based on an information criteria (combining model fit and model complexity) for model comparison.

# Testing-based procedures

Backward elimination

1. Start with all the predictors in the model.
2. Remove the predictor with highest $p-$value $> \alpha_0$ (most insignificant).
3. Refit the model, and repeat the above process.
4. Stop when all $p-$values $\leq \alpha_0$.
   ($\alpha_0$ is often set to 15% or 20% which is higher than usual)

# Testing-based procedures

Forward selection

1. Start with the intercept-only model.
2. For all predictors not in the model, check their $p$-value if being added to the model. Add the one with the lowest $p-$value $\leq \alpha_0$ (most significant).
3. Refit the model, and repeat the above process.
4. Stop when no more predictors can be added.

## Pros and Cons of Testing-based procedures

- Main advantage: Computation cost is low.
- Due to the "one-at-a-time" nature of adding/dropping variables, this type of procedures does not compare all possible models. So it's possible to miss the "optimal" model.
- It's not clear how to choose $\alpha_0$, the cut-off for $p$-values.

# Criterion-based procedures

1. Score each model according to an information criteria
2. Use a searching algorithm to find the optimal model

Model selection criteria/scores often takes the following form:

$$\text{Training error} + \text{Complexity-penalty}$$

- In the context of linear regression models, complexity of a model increases with the number of predictor variables (i.e., $p_{\boldsymbol{\gamma}}$).
- Why we do not use $R^2$ or $RSS$?

# Model Selection Criteria

- AIC/BIC - lower is better

$$AIC : -2 \times loglik_{\boldsymbol{\gamma}} + 2p_{\boldsymbol{\gamma}}$$
$$BIC : -2 \times loglik_{\boldsymbol{\gamma}} + \log(n)p_{\boldsymbol{\gamma}}$$

  where $p_{\boldsymbol{\gamma}}$ is the number of predictors included in model $\boldsymbol{\gamma}$.
- **R** functions AIC and BIC use $p_{\boldsymbol{\gamma}} + 1$ instead of $p_{\boldsymbol{\gamma}}$, due to estimating $\sigma^2$, but the extra constant does not affect the comparison between models.
- For the Gaussian linear regression model:

$$-2 \times loglik_{\boldsymbol{\gamma}} = n \log \frac{RSS_{\boldsymbol{\gamma}}}{n} + \text{constant}$$

- When $n$ is large, adding predictors costs more in BIC than AIC. So AIC tends to pick a bigger model than BIC.

- Adjusted-$R^2$ for model $\boldsymbol{\gamma}$

$$\begin{aligned} R_a^2 &= 1 - \frac{RSS/(n - p_{\boldsymbol{\gamma}} - 1)}{TSS/(n-1)} \\ &= 1 - (1 - R^2)(\frac{n-1}{n - p_{\boldsymbol{\gamma}} - 1}) \\ &= 1 - \frac{\hat{\sigma}^2_{\boldsymbol{\gamma}}}{\hat{\sigma}^2_0} \end{aligned}$$

  The higher the $R_a^2$ the better.

- Mallow's $C_p$

$$C_p = \frac{RSS_{\boldsymbol{\gamma}}}{\hat{\sigma}^2} + 2p_{\boldsymbol{\gamma}} - n$$

  where $\hat{\sigma}^2$ is the estimate of the error variance from the full model. Mallow's $C_p$ behaves very similar to AIC.

# Searching Algorithms

- Leaps and Bounds: return the global optimal solution among all possible models, but only feasible for less than 50 variables.
  - Find the $p$ models with the smallest $RSS$ amongst all models of the same size[1].
  - Then evaluate the score on the $p$ models and report the optimal one.

[1] Note that step 1, we do not need to visit every model. For example, suppose we know that $RSS(X1, X2) < RSS(X3, X4, X5, X6)$; then we do not need to visit any size 2 or 3 sub-models of set $(X3, X4, X5, X6)$, which can be leaped over

- Greedy algorithms: fast, but only return a local optimal solution (which might be good enough in practice).
  - Backward: start with the full model and sequentially delete predictors until the score does not improve.
  - Forward: start with the null model and sequentially add predictors until the score does not improve.
  - Stepwise: consider both deleting and adding one predictor at each stage.

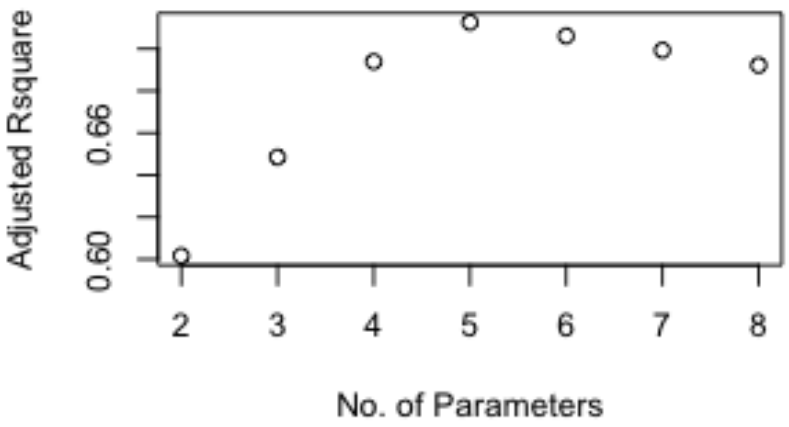
# Example: Life Expectancy data set

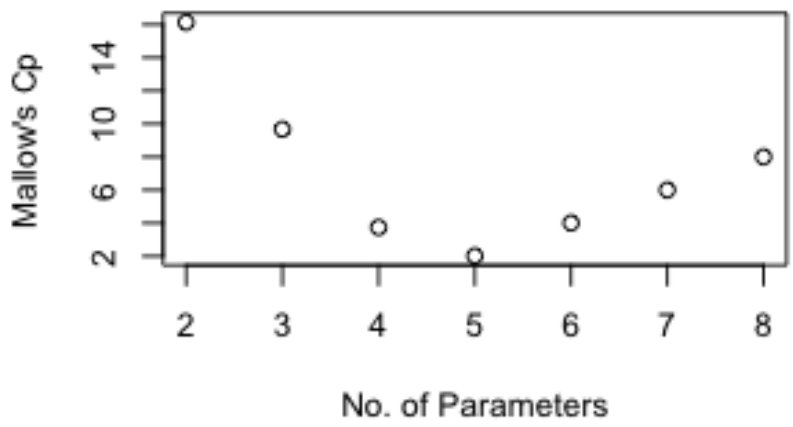
*state.77* data set from library **datasets**

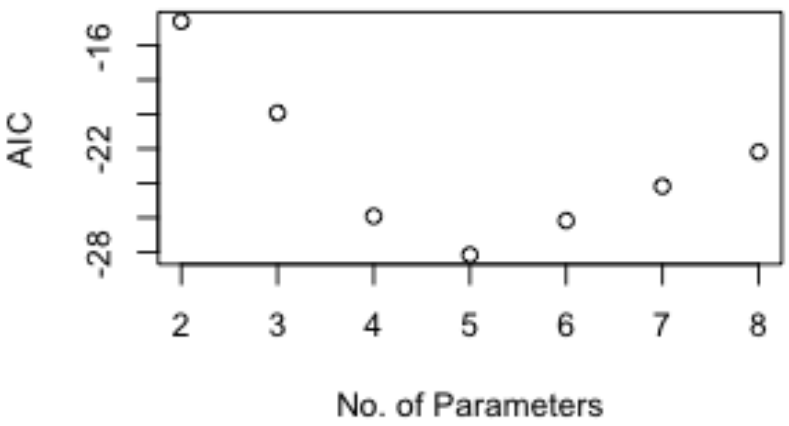
```
# Read state.x77 data set
statedata=data.frame(state.x77,row.names=state.abb,
                     check.names=T)
g = lm(Life.Exp ~., data=statedata)
```

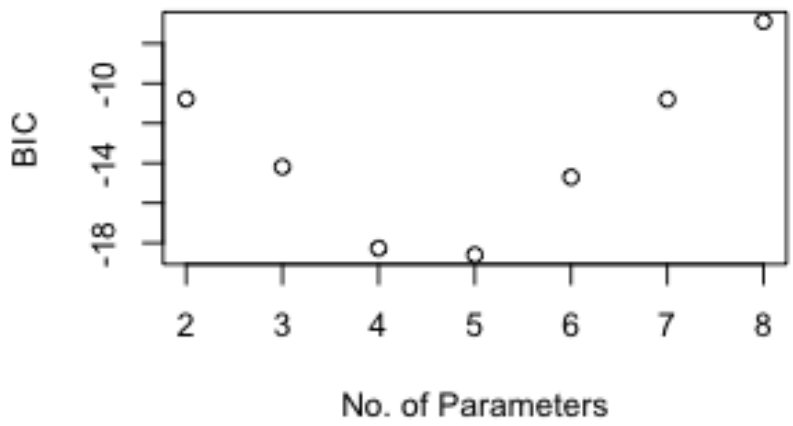
Leaps and Bounds method

Use function *regsubsets* from library **leaps** to evaluate different scores for sub-sets of models up to size $p$ (including the intercept). In this example a model of size $p = 5$ is selected by all criteria. This is not always the case.


Adjusted Rsquare
0.60
0.66
2
3
4
5
6
7
8
No. of Parameters


Mallow's Cp
2
6
10
14
2
3
4
5
6
7
8
No. of Parameters


AIC
-28
-22
-16
2
3
4
5
6
7
8
No. of Parameters


BIC
-18
-14
-10
2
3
4
5
6
7
8
No. of Parameters

## Searching methods

Use function *step* from the **stats** library to apply searching algorithms based on the AIC (default) or BIC criteria ($k = \log(n)$). The option direction=both uses the Stepwise searching algorithm. You can also use the options direction=forward and direction=backward.

```{r}
step(g, direction="both")
step(g, direction="both", k=log(n))
```