Ken Ye

STA325 HW2

#1

a)

Test-error criterion can be used to indirectly estimate traing error by making an adjustment to the training error to account for the bias due to overfitting.

Advs: works well for simple models such as linear regression; more interpretability w/ the adjustment on training error; closed form criterion, can be computed efficiently and easy to apply.

Disadvs: sometimes perform poorly if the adjustment made for training error is far away from the true data set or data distribution.

CV can be used to directly estimate the test error.

Advs: works for a wider range of models such as complicated non-linear models; provides a direct estimate; no need to estimate irreducible noise variance σ^2

Disadvs: results may be highly variable because it depends on which part is used for training; computationaly expensive

Therefore, when we are working w/ a simple model (e.g. linear regression), when computational efficiency is a priority, or when dataset is small, we could use test-error criterion. When we have a more complex / non-parametric model, when dataset is large, or when a more accurate out-of-sample error is required, CV might be a better option.

b)

$$AIC = \frac{1}{n}(RSS + 2d\hat{\sigma}^2)$$

$$BIC = \frac{1}{n}(RSS + \log(n) d\hat{\sigma}^2)$$

where n = # observations
d = # predictors
$\hat{\sigma}^2$ = variance of the ϵ associated w/ each response.

Formula difference: BIC replaces 2 w/ $\log(n)$.

$\log(n) > 2$ when $n > 7$, thus BIC places a heavier penalty on model when there are more than 7 observations, resulting in a model w/ less predictors.

AIC would be preferable when prediction accuracy is the goal, and BIC should be chosen if we want a simpler and more interpretable model.

c)

penalizing more parameters (d): adding more parameters can make a model fit the training data more closely but it may also capture noise (overfitting). The penalty term discourages overly complex models by adding a cost for each additional parameter.

penalizing larger estimated irreducible noise variance ($\hat{\sigma}^2$): a larger $\hat{\sigma}^2$ indicates that the model is not capturing the underlying patterns in the data well, suggesting the noise is high. Penalizing based on $\hat{\sigma}^2$ discourages models that do not significantly reduce the unexplained variability in the data.

d)

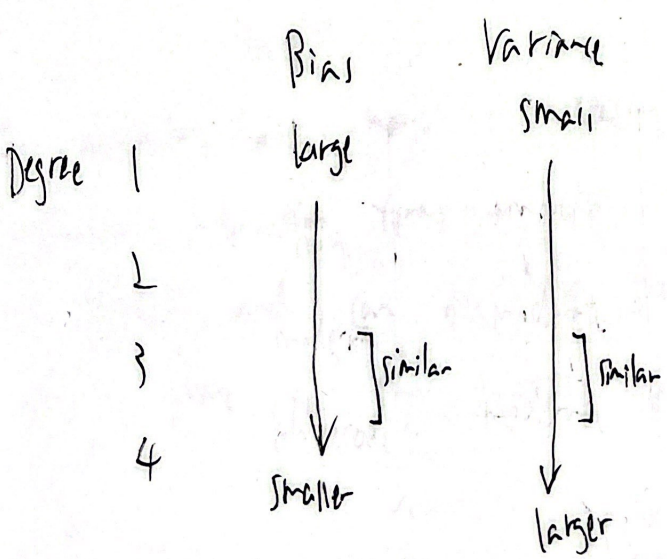
$\hat{f}_1(AIC)$ is likely to have higher variance and lower bias than $\hat{f}_2(BIC)$ because BIC tend to favor simpler models than AIC due to its heavier penalty for addition parameters ($BIC > AIC$ when $n > 7$), and we know that a model with too many parameters may have low bias, but high variance (overfitting), while a model with too few parameters may have high bias, but low variance (underfitting).

#2

a)

They each need 3, 5, 7, and 9 parameters, respectively. For a polynomial of degree M, we need $2M+1$ parameters.

b)



The bias is biggest with model of degree 1 and decrease, with degree 2 and degree 3 (degree 4 looks very similar to degree 3, but bias should be getting smaller) as the fitted line moves closer to the Bayes optimal. The variance is smallest with degree 1 model and increases as more parameters are added (degree 2 → degree 3 → degree 4) because model complexity brings more variance to the fit.

c)

True optimal model : order = 4

Est. optimal model : order = 5

The black curve is generally lower than the orange curve because the 10-fold CV overestimated the test-error because splitting data into k=10 folds and using 90% of the data for training introduces bias that lead to an overestimation of the test error. ↑ instead of 100%

d)

Way #1 : # of parameters. If we increase the number of predictors, the model complexity will increase.

way #2 : Interaction terms. If we add terms of interactions b/w 2 variables that' account for situations where the effect of one variable on the outcome is affected by the level of another variable, thus adding more complexity to the model.

In conclusion, both ways — increasing the # of parameters and including interaction terms — allows the model to have more degrees of freedom and can fit the training data more closely, though there is the caveat of overfitting.

#4

a)

The model chosen by the best subset method will have the smaller training RSS because it considers all possible models (2^p of them) and chooses the one w/ smallest RSS.

b)

It's not possible to definitively say which method (best subset, forward stepwise, or backward stepwise) will provide the model w/ the smallest test RSS for k predictors w/o additional info or testing. This is because the model w/ the smallest test RSS is contingent upon the true underlying data-generating process, which might favor one method over the others in certain contexts. CV or other model validation technique, would typically be used to estimate test error rates and select the best model.

c)

i. [True] In forward stepwise selection, we start w/ a model w/ no predictors and sequentially add predictors that provide the best improvement to the fit.

ii. [True] In backward stepwise selection, we start w/ a model containing all predictors and sequentially remove the predictor that has the least impact on the fit.

iii. [False] Forward and backward stepwise selection do not necessarily yield the same predictors because they add/remove predictors based on different criteria at each step.

iv. [False] Explanation same as iii.

v. [False] Best subset selection chooses the best model at each level k based on some criterion (like RSS/R^2) w/o considering the models at other levels.

hw2

Ken Ye

2023-10-03

Question 3

We will now perform cross-validation on a simulated data set.

(a) Generate a simulated data set as follows:

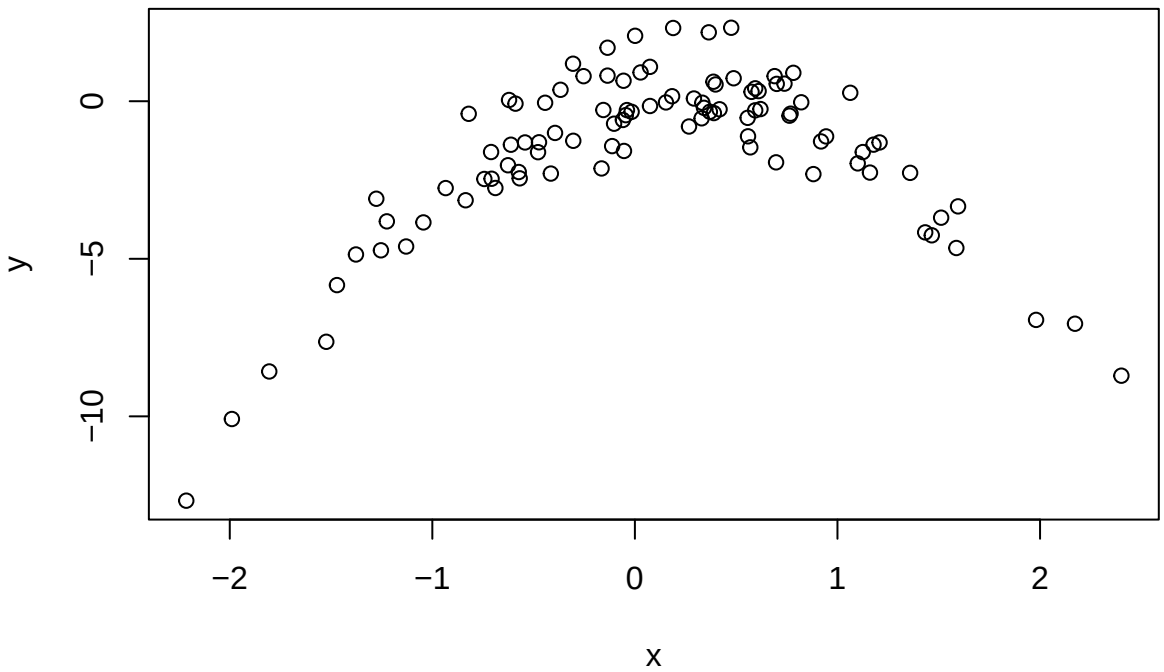
```
set.seed(1)
x = rnorm(100)
y = x - 2 * x^2 + rnorm(100)
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form.

The n (number of observations) is 100, and the p (number of predictors) is 2. The generative model is $Y = X - 2X^2 + \epsilon$, where ϵ is the noise term following distribution N(0,1).

(b) Create a scatterplot of X against Y. Comment on what you find.

```
plot(x, y)
```



The relationship between X and Y is clearly not linear, instead, it seem to follow a parabolic (concave down) trend. The peak of Y occurs at around X=0.

(c) Set a random seed, and then compute the LOOCV errors that result from fitting the following four models using least squares. Note you may find it helpful to use the data.frame() function to create a single data set containing both X and Y.

```
library(boot)
set.seed(2)
df <- data.frame(x, y)
```

```
# Model 1: Linear
model1 = glm(y ~ x)
cv1 = cv.glm(df, model1)
loocv_error1 = cv1$delta[1]

# Model 2: Quadratic
model2 = glm(y ~ x + I(x^2))
cv2 = cv.glm(df, model2)
loocv_error2 = cv2$delta[1]

# Model 3: Cubic
model3 = glm(y ~ x + I(x^2) + I(x^3))
cv3 = cv.glm(df, model3)
loocv_error3 = cv3$delta[1]

# Model 4: Quartic
model4 = glm(y ~ x + I(x^2) + I(x^3) + I(x^4))
cv4 = cv.glm(df, model4)
loocv_error4 = cv4$delta[1]
```

```
# Compare LOOCV Errors
loocv_errors = data.frame(
  Model = c("Linear", "Quadratic", "Cubic", "Quartic"),
  LOOCV_Error = c(loocv_error1, loocv_error2, loocv_error3, loocv_error4)
)
print(loocv_errors)
```

```
##       Model LOOCV_Error
## 1    Linear   7.2881616
## 2 Quadratic   0.9374236
## 3     Cubic   0.9566218
## 4   Quartic   0.9539049
```

(d) Repeat (c) using another random seed, and report your results. Are your results the same as what you got in (c)? Why?

```
set.seed(3)
```

```
# Model 1: Linear
model1 = glm(y ~ x)
cv1 = cv.glm(df, model1)
loocv_error1 = cv1$delta[1]

# Model 2: Quadratic
model2 = glm(y ~ x + I(x^2))
```

```r
cv2 = cv.glm(df, model2)
loocv_error2 = cv2$delta[1]

# Model 3: Cubic
model3 = glm(y ~ x + I(x^2) + I(x^3))
cv3 = cv.glm(df, model3)
loocv_error3 = cv3$delta[1]

# Model 4: Quartic
model4 = glm(y ~ x + I(x^2) + I(x^3) + I(x^4))
cv4 = cv.glm(df, model4)
loocv_error4 = cv4$delta[1]
```

```r
# Compare LOOCV Errors
loocv_errors = data.frame(
  Model = c("Linear", "Quadratic", "Cubic", "Quartic"),
  LOOCV_Error = c(loocv_error1, loocv_error2, loocv_error3, loocv_error4)
)
print(loocv_errors)
```

```
##       Model LOOCV_Error
## 1    Linear   7.2881616
## 2 Quadratic   0.9374236
## 3     Cubic   0.9566218
## 4   Quartic   0.9539049
```

The results (LOOCV errors) are exactly the same for (c) and (d), though we used different seeds. This is because LOOCV uses each data point as a test set exactly once and averages the error over all iterations, and in this case, the random seed (which likely changes the order which each data point is used as a test set) doesn't make a difference.

(e) Which of the models in (c) had the smallest LOOCV error? Is this what you expected? Explain your answer.

The second model (quadratic) in (c) had the smallest LOOCV error, and it fits my expectation because we know the true generative model $Y = X - 2X^2 + \epsilon$ indicates a quadratic relationship between X and Y, which is illustrate in the scatterplot in (b) as well.