Introduction to Machine Learning
Unit 4 Solutions: Model Order Selection

Prof. Sundeep Rangan

1. For each of the following pairs of true functions $f_0(\mathbf{x})$ and model classes $f(\mathbf{x}, \boldsymbol{\beta})$ determine: (i) if the model class is linear; (ii) if there is no under-modeling; and (iii) if there is no under-modeling, what is the true parameter?

   (a) $f_0(x) = 1 + 2x$, $f(x, \boldsymbol{\beta}) = \beta_0 + \beta_1 x + \beta_2 x^2$

   (b) $f_0(x) = 1 + 1/(2 + 3x)$, $f(x, a_0, a_1, b_0, b_1) = (a_0 + a_1 x)/(b_0 + b_1 x)$.

   (c) $f_0(x) = (x_1 - x_2)^2$ and

   $$f(\mathbf{x}, a, b_1, b_2, c_1, c_2) = a + b_1 x_1 + b_2 x_2 + c_1 x_1^2 + +c_2 x_2^2.$$

   Solution

   (a) Linear model; no undermodeling; true parameter is $\boldsymbol{\beta} = (1, 2, 0)$.

   (b) The model is nonlinear. We can write

   $$f_0(x) = 1 + \frac{1}{2 + 3x} = \frac{3 + 3x}{2 + 3x},$$

   So, there is no under-modeling and the true parameters are $(a_0, a_1, b_0, b_1) = (3, 3, 2, 3)$.

   (c) The model is linear. The true function is

   $$f_0(x) = (x_1 - x_2)^2 = x_1^2 - 2x_1 x_2 + x_2^2.$$

   There is undermodeling since the model class doesn't have an $x_1 x_2$ term.

2. You want to fit an exponential model of the form,

   $$y \approx \widehat{y} = \sum_{j=0}^{d} \beta_j e^{-ju/d},$$

   where the input u and output y are scalars. You are given python functions:

```
model = LinearRegression()
model.fit(X,y)            # Fits a linear model for a data matrix X
yhat = model.predict(X)   # Predicts values
```

   Using these functions, write python code that, given vectors `u` and `y`:

- Splits the data into training and test using half the samples for each.
- Fits models of order `dtest = [1,2,...,10]` on the training data.
- Selects the model with the lowest mean squared error.

Solution One solution is as follows:

```
# Split data into training and test
# You can also use train_test_split from sklearn
n = len(u)
ntr = n // 2
utr = u[:ntr]
ytr = y[:ntr]
uts = u[ntr:]
yts = y[ntr:]

# Loop over model orders
dtest = np.arange(1,11)  # Note this loop goes d=1,...,10
nd = len(dtest)
mse = np.zeros(nd)
for i, d in dtest:

    # Transform the data
    # Create the feature matrices using python broadcasting
    #   Xtr[i,j] = np.exp(—utr[i]*j/d)
    #   Xts[i,j] = np.exp(—uts[i]*j/d)
    powers = np.arange(0,d+1)/d
    Xtr = np.exp(—utr[:,None]*powers[None,:])
    Xts = np.exp(—uts[:,None]*powers[None,:])

    # Create model
    model = LinearRegression()

    # Fit model on training data
    model.fit(Xtr, ytr)

    # Measure MSE on test data
    yhat = model.predict(Xts)
    mse[i] = np.mean((yhat — yts)**2)

# Select model with lower test error
im = np.argmin(mse)
dopt = dtest[im]
```

3. Suppose we want to fit a model,

$$y \approx \widehat{y} = f(x, \beta) = \beta x^2.$$

We get data (x_i, y_i), $i = 1, \ldots, N$ and compute the estimate,

$$\widehat{\beta} = \frac{\sum_{i=1}^N y_i}{\sum_{i=1}^N x_i^2}.$$

Note: This is not optimal least-squares estimator. But, it is easier to analyze. For each case below compute the bias,

$$\text{Bias}(x) := \mathbb{E}(f(x, \widehat{\beta})) - f(x, \beta_0),$$

as a function of the test point x, true parameter β_0 and test data x_i.

(a) The training data has no noise: $y_i = f(x_i, \beta_0)$.

(b) The training data is $y_i = f(x_i, \beta_0) + \epsilon_i$ where the noise is i.i.d. $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

(c) The training data is $y_i = f(x_i + \epsilon_i, \beta_0)$ where the noise is i.i.d. $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$.

Solution To compute the bias, first observe that

$$\begin{aligned} \text{Bias}(x) &:= \mathbb{E}(f(x, \widehat{\beta})) - f(x, \beta_0) = \mathbb{E}(\widehat{\beta} x^2) - \beta_0 x^2 \\ &= \left[\mathbb{E}(\widehat{\beta}) - \beta_0\right] x^2, \end{aligned} \tag{1}$$

where we have used the linearity of expectation. Remember that the test point x is not random, so you can "pull it out" of the expectation. Now, we look at the expectation of $\widehat{\beta}$:

$$\mathbb{E}\left(\widehat{\beta}\right) = \mathbb{E}\left[\frac{\sum_{i=1}^N y_i}{\sum_{i=1}^N x_i^2}\right] = \frac{\sum_{i=1}^N \mathbb{E}(y_i)}{\sum_{i=1}^N x_i^2}, \tag{2}$$

where we have used that training data x_i is not random. So, again we can pull the denominator out of the expectation. We can now evaluate the bias for each of the three cases.

(a) In this case,

$$\mathbb{E}(y_i) = \mathbb{E}(f(x_i, \beta_0)) = \mathbb{E}(x_i^2 \beta_0) = x_i^2 \beta_0.$$

In the last step, since there is no randomness (remember the training data x_i and true parameter β_0 are not random). Substituting this expectation into (2) we obtain,

$$\mathbb{E}\left(\widehat{\beta}\right) = \frac{\sum_{i=1}^N \mathbb{E}(y_i)}{\sum_{i=1}^N x_i^2} = \frac{\sum_{i=1}^N x_i^2 \beta_0}{\sum_{i=1}^N x_i^2} = \beta_0.$$

Therefore, from (1),

$$\text{Bias}(x) = \left[\mathbb{E}(\widehat{\beta}) - \beta_0\right] x^2 = [\beta_0 - \beta_0]\, x^2 = 0.$$

So, the bias is zero. We say the estimator is *unbiased.*

(b) In this case,

$$\begin{aligned} \mathbb{E}(y_i) &= \mathbb{E}(f(x_i, \beta_0) + \epsilon_i) = \mathbb{E}(x_i^2 \beta_0 + \epsilon_i) \\ &= x_i^2 \beta_0 + \mathbb{E}(\epsilon_i) = x_i^2 \beta_0, \end{aligned}$$

where in the last step, we used that $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ so $\mathbb{E}(\epsilon_i) = 0$. The final expression, $\mathbb{E}(y_i) = x_i^2 \beta_0$ is identical to part (a). So, again we get $\text{Bias}(x) = 0$ for all test points x.

(c) For this case,

$$\begin{aligned}\mathbb{E}(y_i) &= \mathbb{E}(f(x_i+\epsilon_i,\beta_0)) = \mathbb{E}((x_i+\epsilon_i)^2\beta_0) \\ &= \left[x_i^2\beta_0 + 2\mathbb{E}(\epsilon_i)x_i\beta_0 + \mathbb{E}(\epsilon_i^2)\right]\beta_0 \\ &= \left[x_i^2+\sigma^2\right]\beta_0,\end{aligned}$$

where, in the last step, we used that $\mathbb{E}(\epsilon_i)=0$ and $\mathbb{E}(\epsilon_i^2)=\sigma^2$. Substituting this into (2),

$$\mathbb{E}\left(\widehat{\beta}\right) = \frac{\sum_{i=1}^N \mathbb{E}(y_i)}{\sum_{i=1}^N x_i^2} = \frac{\sum_{i=1}^N (x_i^2+\sigma^2)\beta_0}{\sum_{i=1}^N x_i^2} = \beta_0 + \frac{\beta_0 N\sigma^2}{\sum_{i=1}^N x_i^2}.$$

Substituting into (1),

$$\text{Bias}(x) = \left[\mathbb{E}(\widehat{\beta}) - \beta_0\right]x^2 = \frac{\beta_0 N\sigma^2 x^2}{\sum_{i=1}^N x_i^2}.$$

So, in this case, there is a bias in the estimator.

4. In this problem, we will see how to calculate the bias when there is undermodeling. Suppose that training data (x_i, y_i), $i=1,\ldots,n$ is fit using a simple linear model of the form,

$$\hat{y} = f(x,\boldsymbol{\beta}) = \beta_0 + \beta_1 x.$$

However, the true relation between x and y is given

$$y = f_0(x), \quad f_0(x) = \beta_{00} + \beta_{01}x + \beta_{02}x^2,$$

where the "true" function $f_0(x)$ is quadratic and $\boldsymbol{\beta}_0 = (\beta_{00}, \beta_{01}, \beta_{02})$ is the vector of the true parameters. There is no noise.

(a) Write an expression for the least-squares estimate $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_0, \widehat{\beta}_1)$ in terms of the training data (x_i, y_i), $i=1,\ldots,n$. These expressions will involve multiple steps. You do not need to simplify the equations. Just make sure you state clearly how one would compute $\widehat{\boldsymbol{\beta}}$ from the training values.

(b) Using the fact that $y_i = f_0(x_i)$ in the training data, write the expression for $\boldsymbol{\beta} = (\widehat{\beta}_0, \widehat{\beta}_1)$ in terms of the values x_i and the true parameter values $\boldsymbol{\beta}_0$. Again, you do not need to simplify the equations. Just make sure you state clearly how one would compute $\widehat{\boldsymbol{\beta}}$ from the true parameter vector $\boldsymbol{\beta}_0$ and $\mathbf{x}$.

(c) Suppose that the true parameters are $\boldsymbol{\beta}_0 = (1, 2, -1)$ and the model is trained using 10 values x_i uniformly spaced in $[0,1]$. Write a short python program to compute the estimate parameters $\widehat{\boldsymbol{\beta}}$. Plot the estimated function $f(x,\widehat{\boldsymbol{\beta}})$ and true function $f_0(x)$ for $x \in [0,3]$.

(d) For what value x in this range $x \in [0,3]$ is the bias $\text{Bias}^2(x) = (f(x,\widehat{\boldsymbol{\beta}}) - f_0(x))^2$ largest?

Solution

(a) This is simple linear regression so, from the class notes, the parameter estimates are

$$\widehat{\beta}_1 = \frac{s_{xy}}{s_{xx}}, \quad \beta_0 = \bar{y} - \widehat{\beta}_1 \bar{x},$$

where $\bar{x}$ and $\bar{y}$ are the sample means and s_{xy} and s_{xx} are the sample co-variance and variance:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i,$$
$$s_{xy} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}), \quad s_{xx} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2.$$

(b) Use the above expressions but substitute $y_i = \beta_{00} + \beta_{01}x_i + \beta_{02}x_i^2$.

(c) Suppose that the true parameters are $\boldsymbol{\beta}_0 = (1, 2, 0.5)$ and the model is trained using 10 values x_i uniformly spaced in $[0, 1]$. Write a short python program to compute the estimate parameters $\widehat{\boldsymbol{\beta}}$. Plot the estimated function $f(x, \widehat{\boldsymbol{\beta}})$ and true function $f_0(x)$ for $x \in [0, 3]$.

You can compute the estimate and plot the estimate and true function with the following code:

```
import numpy.polynomial.polynomial as poly

beta0 = np.array([1,2,-1])  # True parameter value
x = np.linspace(0,1,10)      # Training values for x
y = poly.polyval(x,beta0)    # Training values for y

# Get parameter estimate based on simple linear regression formula
xm = np.mean(x)
ym = np.mean(y)
sxy = np.mean((x-xm)*(y-ym))
sxx = np.mean((x-xm)**2)
betahat1 = sxy/sxx
betahat0 = ym - betahat1*xm

# Plot true function and estimate
xp = np.linspace(0,3,100)
yp0 = poly.polyval(xp,beta0)
yphat = betahat0 + betahat1*xp

plt.plot(xp,np.column_stack((yp0, yphat)), '-')
plt.scatter(x,y)
plt.legend(['True', 'Est', 'Training'], loc='upper left')
plt.grid()
plt.xlim([0,3])
plt.xlabel('x')
```

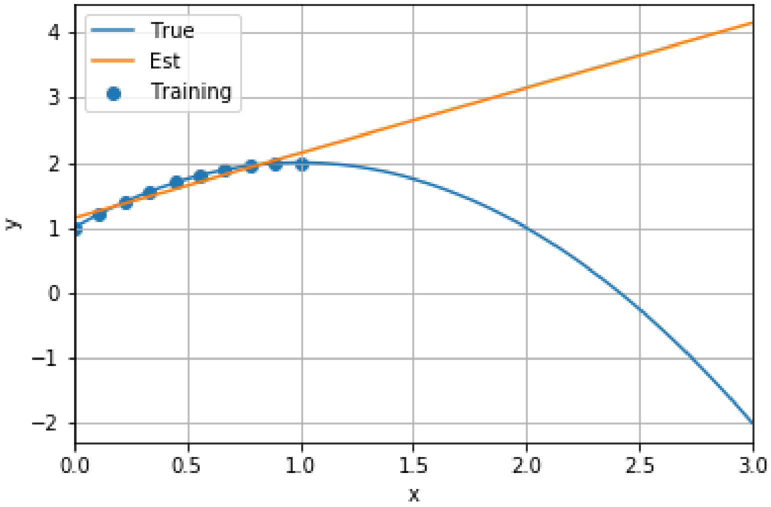

Figure 1: True function $f_0(x)$, estimate $f(x, \widehat{\boldsymbol{\beta}})$ and training points (x_i, y_i)

```
plt.ylabel('y')
plt.savefig('bias.png')
```

The resulting figure is shown in Fig. 1.

(d) We see that the linear fit attempts to fit the quadratic in the region $x \in [0, 1]$ where the training data was. But, the fit is poor outside this region. In particular, in the interval $[0, 3]$, the bias error (difference between the true and estimated function) is largest at $x = 3$.

5. A medical researcher wishes to evaluate a new diagnostic test for cancer. A clinical trial is conducted where the diagnostic measurement y of each patient is recorded along with attributes of a sample of cancerous tissue from the patient. Three possible models are considered for the diagnostic measurement:

- Model 1: The diagnostic measurement y depends linearly only on the cancer volume.
- Model 2: The diagnostic measurement y depends linearly on the cancer volume and the patient's age.
- Model 3: The diagnostic measurement y depends linearly on the cancer volume and the patient's age, but the dependence (slope) on the cancer volume is different for two types of cancer – Type I and II.

(a) Define variables for the cancer volume, age and cancer type and write a linear model for the predicted value $\hat{y}$ in terms of these variables for each of the three models above. For Model 3, you will want to use one-hot coding.

(b) What are the numbers of parameters in each model? Which model is the most complex?

(c) Since the models in part (a) are linear, given training data, we should have $\hat{\mathbf{y}} = \mathbf{A}\boldsymbol{\beta}$ where $\hat{\mathbf{y}}$ is the vector of predicted values on the training data, $\mathbf{A}$ is a feature matrix and $\boldsymbol{\beta}$ is the vector of parameters. To test the different models, data is collected from 100 patients. The records of the first three patients are shown below:

Patient ID	Measurement y	Cancer type	Cancer volume	Patient age
12	5	I	0.7	55
34	10	II	1.3	65
23	15	II	1.6	70
⋮	⋮	⋮	⋮	⋮

Based on this data, what would be the values of first three rows of the three $\mathbf{A}$ matrices be for the three models in part (a)?

(d) To evaluate the models, 10-fold cross validation is used with the following results.

Model	Mean training RSS	Mean test RSS	Test RSS std deviation
1	2.0	2.01	0.03
2	0.7	0.72	0.04
3	0.65	0.70	0.05

All RSS values are per sample, and the last column is the (biased) standard deviation – not the standard error. Which model should be selected based on the "one standard error rule"?

Solution

(a) Let x_1 be the cancer volume, x_2 be the patient's age and x_3 be the cancer type:

$$x_3 = \begin{cases} 0 & \text{cancer is Type I} \\ 1 & \text{cancer is Type II,} \end{cases}$$

Then, the models can be written as:

$$\begin{aligned} &\text{Model 1:} \quad \hat{y} = \beta_0 + \beta_1 x_1, \\ &\text{Model 2:} \quad \hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2, \\ &\text{Model 3:} \quad \hat{y} = \beta_0 + \beta_1 x_1(1 - x_3) + \beta_2 x_1 x_3 + \beta_3 x_2. \end{aligned}$$

In Model 3, we have used one-hot coding on the slope of x_1. Specifically, when $x_3 = 0$ (Type I cancer), the slope for x_1 is β_1; when $x_3 = 1$ (Type II cancer), the slope for x_1 is β_2.

(b) Models 1, 2 and 3 have 2, 3 and 4 parameters respectively. Model 3 is most complex.

(c) For Model 1, the first three rows of the feature matrix are:

$$\mathbf{A} = \begin{bmatrix} 1 & x_{11} \\ 1 & x_{21} \\ 1 & x_{31} \\ \vdots & \vdots \end{bmatrix} = \begin{bmatrix} 1 & 0.7 \\ 1 & 1.3 \\ 1 & 1.6 \\ \vdots & \vdots \end{bmatrix}.$$

For Model 2, the first three rows of the feature matrix are:

$$\mathbf{A} = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \\ 1 & x_{31} & x_{32} \\ \vdots & & \vdots \end{bmatrix} = \begin{bmatrix} 1 & 0.7 & 55 \\ 1 & 1.3 & 65 \\ 1 & 1.6 & 70 \\ \vdots & & \vdots \end{bmatrix}.$$

For Model 3, the first three rows of the feature matrix are:

$$\mathbf{A} = \begin{bmatrix} 1 & x_{11}(1-x_{13}) & x_{11}x_{13} & x_{12} \\ 1 & x_{21}(1-x_{23}) & x_{21}x_{23} & x_{22} \\ 1 & x_{31}(1-x_{33}) & x_{31}x_{33} & x_{32} \\ \vdots & & & \vdots \end{bmatrix} = \begin{bmatrix} 1 & 0.7 & 0 & 55 \\ 1 & 0 & 1.3 & 65 \\ 1 & 0 & 1.6 & 70 \\ \vdots & & & \vdots \end{bmatrix}.$$

(d) The lowest test error is for Model 3 with a mean RSS $= 0.70$. The standard deviation is 0.05, so the standard error is

$$SE = 0.05/\sqrt{K-1} = 0.05/\sqrt{9} = 0.0167.$$

Note the use of $\sqrt{K-1}$ since the standard deviation was biased. Hence, the RSS target is $0.70 + 0.0167 = 0.7167$. The least complex model below this target is Model 3.