Introduction to Machine Learning
Homework 3 Solutions: Model Order Selection

Prof. Sundeep Rangan

1. (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

   $$f_0(x) = (x_1 - x_2)^2 = x_1^2 - 2x_1x_2 + x_2^2.$$

   There is undermodeling since the model class doesn't have an x_1x_2 term.

2. (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
```

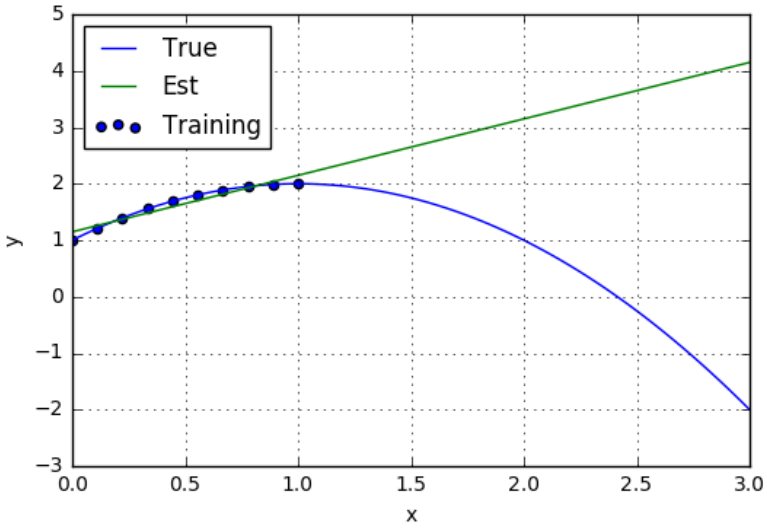

Figure 1: True function $f_0(x)$, estimate $f(x, \widehat{\boldsymbol{\beta}})$ and training points (x_i, y_i)

```
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')
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 $x = 3$.

3. (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:

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

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:

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

(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.