Introduction to Machine Learning
Problems Unit 4: Model Order Selection

Prof. Sundeep Rangan
Solutions by Willem Neuefeind Lessig 9/24/25

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) $f_0(x) = 1 + 2x$, $f(x, \boldsymbol{\beta}) = \beta_0 + \beta_1 x + \beta_2 x^2$
   Solution:
   (i) Linear: Yes, the model is linear in parameters $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$.
   (ii) Under-modeling: No, since $f_0(x) = 1 + 2x$ can be represented as $\beta_0 + \beta_1 x + \beta_2 x^2$ with $\beta_2 = 0$.
   (iii) True parameter: $\boldsymbol{\beta}_0 = (1, 2, 0)$.

   (b) $f_0(x) = 1 + 1/(2 + 3x)$, $f(x, a_0, a_1, b_0, b_1) = (a_0 + a_1 x)/(b_0 + b_1 x)$.
   Solution:
   (i) Linear: No, the model is nonlinear in parameters due to the ratio form.
   (ii) Under-modeling: No, we can rewrite $f_0(x) = 1 + \frac{1}{2+3x} = \frac{2+3x+1}{2+3x} = \frac{3+3x}{2+3x}$.
   (iii) True parameter: $a_0 = 3, a_1 = 3, b_0 = 2, b_1 = 3$.

   (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:
   (i) Linear: Yes, the model is linear in parameters.
   (ii) Under-modeling: Yes, there is under-modeling. $f_0(x) = (x_1 - x_2)^2 = x_1^2 - 2x_1 x_2 + x_2^2$ contains a cross-term $x_1 x_2$ that cannot be represented by the model class.
   (iii) Not applicable since there is under-modeling.

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:

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Split data into training and test (50—50 split)
u_train, u_test, y_train, y_test = train_test_split(u, y, test_size=0.5, random_state=42)

dtest = list(range(1, 11))  # [1, 2, ..., 10]
best_mse = float('inf')
best_d = None
best_model = None

for d in dtest:
    # Create feature matrix for exponential model
    # X has columns: [e^(—0*u/d), e^(—1*u/d), ..., e^(—d*u/d)]
    X_train = np.zeros((len(u_train), d+1))
    X_test = np.zeros((len(u_test), d+1))

    for j in range(d+1):
        X_train[:, j] = np.exp(—j * u_train / d)
        X_test[:, j] = np.exp(—j * u_test / d)

    # Fit linear regression model
    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict on test data
    yhat_test = model.predict(X_test)

    # Calculate MSE
    mse = mean_squared_error(y_test, yhat_test)

    # Keep track of best model
    if mse < best_mse:
        best_mse = mse
        best_d = d
        best_model = model
```

```
print(f"Best model order: {best_d}")
print(f"Best MSE: {best_mse}")
```

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: $y_i = f(x_i, \beta_0)$.
Solution: Since $y_i = \beta_0 x_i^2$, we have:

$$\widehat{\beta} = \frac{\sum_{i=1}^N \beta_0 x_i^2}{\sum_{i=1}^N x_i^2} = \beta_0$$

Therefore:

$$\text{Bias}(x) = \mathbb{E}(\beta_0 x^2) - \beta_0 x^2 = 0$$

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

$$\widehat{\beta} = \frac{\sum_{i=1}^N (\beta_0 x_i^2 + \epsilon_i)}{\sum_{i=1}^N x_i^2} = \beta_0 + \frac{\sum_{i=1}^N \epsilon_i}{\sum_{i=1}^N x_i^2}$$

Taking expectation:

$$\mathbb{E}(\widehat{\beta}) = \beta_0 + \frac{\sum_{i=1}^N \mathbb{E}(\epsilon_i)}{\sum_{i=1}^N x_i^2} = \beta_0$$

Therefore:

$$\text{Bias}(x) = \mathbb{E}(\widehat{\beta} x^2) - \beta_0 x^2 = \beta_0 x^2 - \beta_0 x^2 = 0$$

(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: Now $y_i = \beta_0 (x_i + \epsilon_i)^2 = \beta_0 (x_i^2 + 2 x_i \epsilon_i + \epsilon_i^2)$, so:

$$\widehat{\beta} = \frac{\sum_{i=1}^N \beta_0 (x_i^2 + 2 x_i \epsilon_i + \epsilon_i^2)}{\sum_{i=1}^N x_i^2}$$

Taking expectation:

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

$$= \beta_0 \left(1 + \frac{N\sigma^2}{\sum_{i=1}^N x_i^2}\right)$$

Therefore:

$$\text{Bias}(x) = \beta_0 x^2 \left(1 + \frac{N\sigma^2}{\sum_{i=1}^N x_i^2}\right) - \beta_0 x^2 = \beta_0 x^2 \frac{N\sigma^2}{\sum_{i=1}^N x_i^2}$$

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.

Solution: The least-squares estimate is given by:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$$

where:

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Step-by-step computation:

$$\mathbf{X}^\mathsf{T}\mathbf{X} = \begin{pmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{pmatrix} \tag{1}$$

$$\mathbf{X}^\mathsf{T}\mathbf{y} = \begin{pmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{pmatrix} \tag{2}$$

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y} \tag{3}$$

(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 $\mathbf{x}$.

Solution: Since $y_i = f_0(x_i) = \beta_{00} + \beta_{01}x_i + \beta_{02}x_i^2$, we have:

$$\mathbf{y} = \begin{pmatrix} \beta_{00} + \beta_{01}x_1 + \beta_{02}x_1^2 \\ \beta_{00} + \beta_{01}x_2 + \beta_{02}x_2^2 \\ \vdots \\ \beta_{00} + \beta_{01}x_n + \beta_{02}x_n^2 \end{pmatrix}$$

Then:

$$\mathbf{X}^\mathsf{T}\mathbf{y} = \begin{pmatrix} \sum_{i=1}^n (\beta_{00} + \beta_{01}x_i + \beta_{02}x_i^2) \\ \sum_{i=1}^n x_i(\beta_{00} + \beta_{01}x_i + \beta_{02}x_i^2) \end{pmatrix} \tag{4}$$

$$= \begin{pmatrix} n\beta_{00} + \beta_{01}\sum_{i=1}^n x_i + \beta_{02}\sum_{i=1}^n x_i^2 \\ \beta_{00}\sum_{i=1}^n x_i + \beta_{01}\sum_{i=1}^n x_i^2 + \beta_{02}\sum_{i=1}^n x_i^3 \end{pmatrix} \tag{5}$$

And $\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$ with the above expressions.

(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 $x \in [0, 3]$.

Solution:

```
import numpy as np
import matplotlib.pyplot as plt

# True parameters
beta_00, beta_01, beta_02 = 1, 2, -1

# Training data: 10 points uniformly spaced in [0,1]
x_train = np.linspace(0, 1, 10)
y_train = beta_00 + beta_01*x_train + beta_02*x_train**2

# Create design matrix for linear model
X = np.column_stack([np.ones(len(x_train)), x_train])

# Compute least squares estimate
beta_hat = np.linalg.solve(X.T @ X, X.T @ y_train)
print(f"Estimated parameters: beta_0 = {beta_hat[0]:.3f}, beta_1 = {beta_hat[1]:.3f}")

# Plot for x in [0,3]
x_plot = np.linspace(0, 3, 100)
y_true = beta_00 + beta_01*x_plot + beta_02*x_plot**2
y_estimated = beta_hat[0] + beta_hat[1]*x_plot

plt.figure(figsize=(10, 6))
plt.plot(x_plot, y_true, 'b-', label='True function $f_0(x)$', linewidth=2)
plt.plot(x_plot, y_estimated, 'r--', label='Estimated function $f(x,\\hat{\\beta})$', linewi
```

```
plt.scatter(x_train, y_train, color='black', s=50, label='Training data', zorder=5)
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.grid(True, alpha=0.3)
plt.title('True vs Estimated Function')
plt.show()
```

(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:

```
# Calculate squared bias
bias_squared = (y_estimated - y_true)**2

# Find maximum bias
max_bias_idx = np.argmax(bias_squared)
x_max_bias = x_plot[max_bias_idx]
max_bias_value = bias_squared[max_bias_idx]

print(f"Maximum squared bias occurs at x = {x_max_bias:.3f}")
print(f"Maximum squared bias value = {max_bias_value:.3f}")

# Plot squared bias
plt.figure(figsize=(10, 6))
plt.plot(x_plot, bias_squared, 'g-', linewidth=2)
plt.axvline(x_max_bias, color='red', linestyle='--', alpha=0.7)
plt.xlabel('x')
plt.ylabel('Squared Bias')
plt.title('Squared Bias vs x')
plt.grid(True, alpha=0.3)
plt.show()
```

The maximum squared bias typically occurs at $x = 3$ (the right endpoint) where the quadratic nature of the true function creates the largest deviation from the linear approximation.

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.

Solution: Let v = cancer volume, a = patient age, and define indicator variables: $I_1 = 1$ if cancer type I, 0 otherwise; $I_2 = 1$ if cancer type II, 0 otherwise.

Model 1: $\hat{y} = \beta_0 + \beta_1 v$

Model 2: $\hat{y} = \beta_0 + \beta_1 v + \beta_2 a$

Model 3: $\hat{y} = \beta_0 + \beta_1 v \cdot I_1 + \beta_2 v \cdot I_2 + \beta_3 a$

(Note: Model 3 allows different slopes for volume based on cancer type)

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

Solution:

- Model 1: 2 parameters (β_0, β_1)
- Model 2: 3 parameters $(\beta_0, \beta_1, \beta_2)$
- Model 3: 4 parameters $(\beta_0, \beta_1, \beta_2, \beta_3)$

Model 3 is the most complex with 4 parameters.

(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)?

Solution:

Model 1: $\mathbf{A}_1 = \begin{pmatrix} 1 & 0.7 \\ 1 & 1.3 \\ 1 & 1.6 \end{pmatrix}$

Model 2: $\mathbf{A}_2 = \begin{pmatrix} 1 & 0.7 & 55 \\ 1 & 1.3 & 65 \\ 1 & 1.6 & 70 \end{pmatrix}$

Model 3: $\mathbf{A}_3 = \begin{pmatrix} 1 & 0.7 & 0 & 55 \\ 1 & 0 & 1.3 & 65 \\ 1 & 0 & 1.6 & 70 \end{pmatrix}$

(Columns are: intercept, $v \cdot I_1$, $v \cdot I_2$, age)

(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: First, we need to convert the standard deviation to standard error: SE = $\frac{\text{std dev}}{\sqrt{10}}$ (since 10-fold CV)

- Model 1: Mean test RSS = 2.01, SE = $0.03/\sqrt{10} = 0.0095$
- Model 2: Mean test RSS = 0.72, SE = $0.04/\sqrt{10} = 0.0126$
- Model 3: Mean test RSS = 0.70, SE = $0.05/\sqrt{10} = 0.0158$

The best model (lowest test RSS) is Model 3 with RSS = 0.70.

One standard error rule: Select the simplest model within one SE of the best model.

Upper bound for Model 3: 0.70 + 0.0158 = 0.7158

Checking which models fall within this bound:

- Model 1: 2.01 ¿ 0.7158 (not within one SE)
- Model 2: 0.72 ¿ 0.7158 (not within one SE)
- Model 3: 0.70 ¡ 0.7158 (best model)

Only Model 3 falls within one standard error of the best model.

Therefore, **Model 3** should be selected using the one standard error rule.