HW01_selecting and fitting a model_EH

January 14, 2019

0.1 Selecting and fitting a model

Ellen Hsieh

```
In [1]: import matplotlib.pyplot as plt
        import numpy as np
```

0.1.1 1.

The solution for this part is included in the pdf file with the first part of the homework.

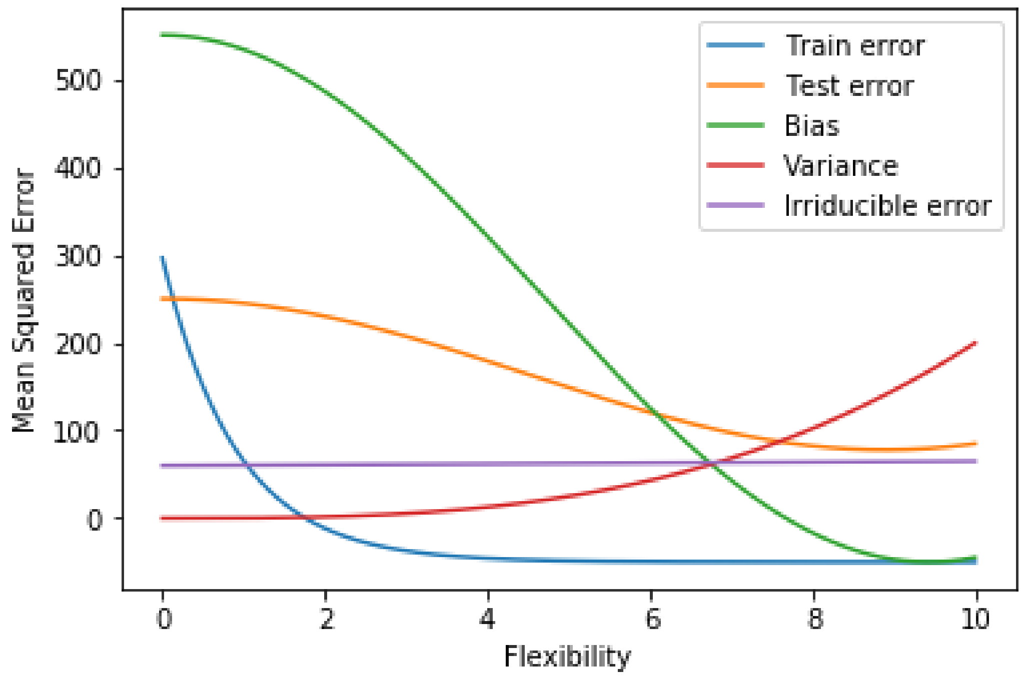
0.1.2 2.

```
In [2]: x = np.arange(0, 10, 0.01)

        y1 = 200 / 3 ** (x - 0.5) - 50
        y2 = 100 * np.cos(x / 3) + 150 + x**2 / 3
        y3 = 300 * np.cos(x / 3) + 250
        y4 = 0.2 * x ** 3
        y5 = 0.5 * x + 60

        plt.plot(x, y1, label = 'Train error')
        plt.plot(x, y2, label = 'Test error')
        plt.plot(x, y3, label = 'Bias')
        plt.plot(x, y4, label = 'Variance')
        plt.plot(x, y5, label = 'Irriducible error')

        plt.xlabel('Flexibility')
        plt.ylabel('Mean Squared Error')
        plt.legend()
        plt.show()
```



(1) The training error decreases as flexibility increases because a more fexible model will fit the observed data more closely.

(2) The testing error intially declines then start to increase at certain point as flexibility increases, which presents a U-shape. This particulat shape indicate the trade-off between bias and variance.

(3) The bias declines monotonically as flexibilty increases. Bias refers to the gap(error) between the real world and the model since the simple model such as linear regression cannot comprehensively represent the real-world problem.

(4) The variance refers to differences between the training data sets we use. Thus, if the felxibility increases which implies that the variance will also increase. If the curve fits the observational data very closely, changing any point may cause the curve to change significantly.

(5) The irreducible error is a parallel line. No matter how the flexibility changes, it remains the same.

0.1.3 3.

```
In [3]: np.random.seed(625)

        x1 = np.random.uniform(-1,1,200)
        x2 = np.random.uniform(-1,1,200)

        mean = 0
```

```
std = 0.5
error = np.random.normal(mean, std, 200)
y = x1 + x1*x1 + x2 + x2*x2 + error

prob_success = np.exp(y) / (1 + np.exp(y))
out_success = prob_success > 0.5
plt.scatter(x1[out_success], x2[out_success], label='success')
out_fail = prob_success <= 0.5
plt.scatter(x1[out_fail], x2[out_fail], label='failure')
plt.legend()

X, Y = np.meshgrid(np.linspace(-1, 1, 200), np.linspace(-1, 1, 200))
yy = X + X*X + Y + Y*Y
prob = np.exp(yy) / (1 + np.exp(yy))
plt.contour(X, Y, (prob > 0.5).astype(np.int), linestyles='dashed')
plt.xlabel('x1')
plt.ylabel('x2')
plt.title('Bayes classifier')
plt.show()
```