15.1) Bias-Variance Tradeoff

Vitor Kamada

February 2020

Reference

Tables, Graphics, and Figures from

Principles and Techniques of Data Science

Lau et al. (2019): Ch 15 Bias-Variance Tradeoff

https://www.textbook.ds100.org/ch/15/bias_intro.html

Bias-Variance Decomposition

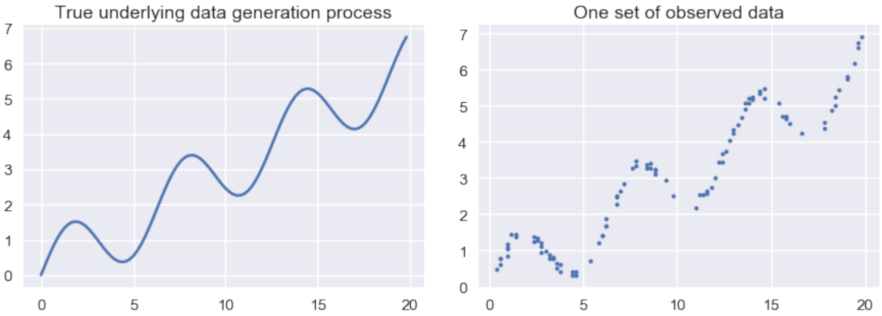
$$\gamma = f_{\theta}(z) + \epsilon$$

$$R(f_{\hat{\theta}}) = \mathbb{E}[(\gamma - f_{\hat{\theta}}(z))^2]$$

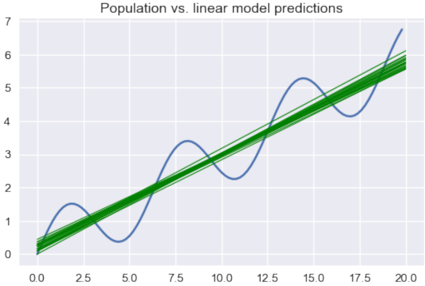
$$\{\mathbb{E}[f_{\hat{\theta}}(z)] - f_{\hat{\theta}}(z)\}^2 + Var(f_{\hat{\theta}}(z)) + Var(\epsilon)$$

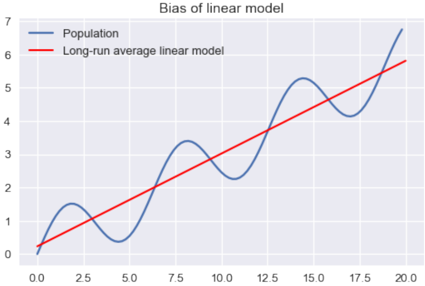
$$\text{bias}^2 + \text{model variance} + \text{irreducible error}$$

Linear Regression and Sine Waves

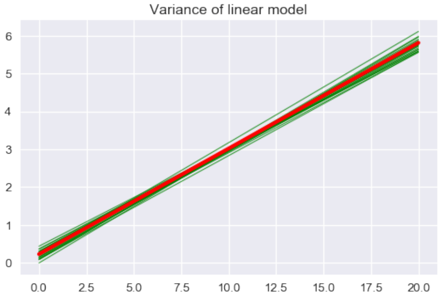


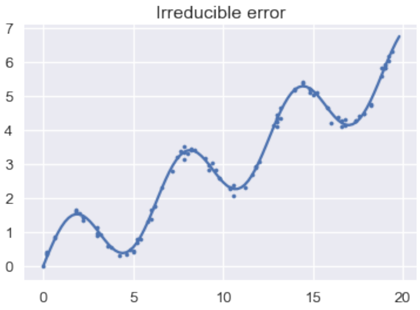
Many Sets of Data from the Population



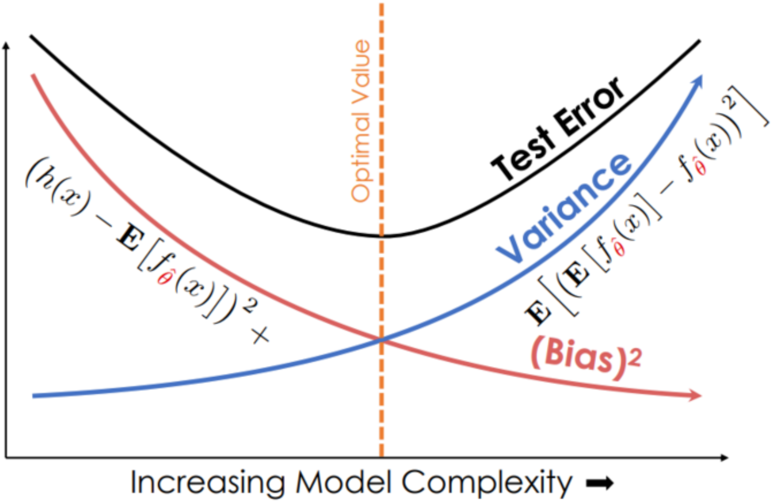


Model Variance and Noise

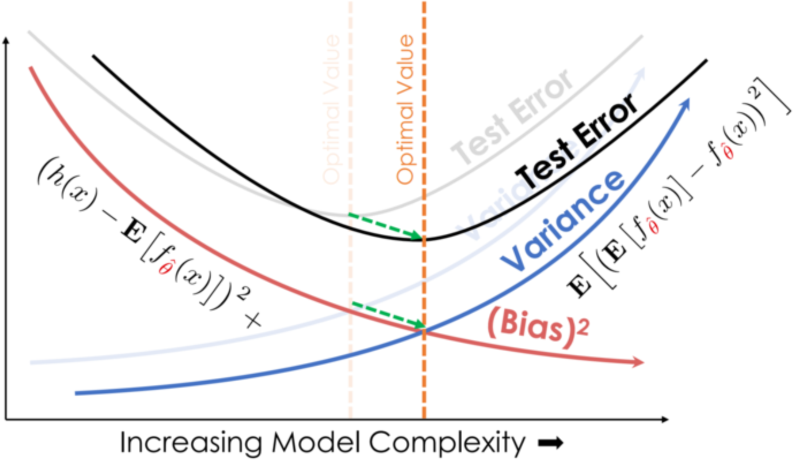




risk $=$ bias^2$+$ variance $+$ noise



More Data Reduces Bias and Variance



Derivation of Bias-Variance

$$\mathbb{E}[(\gamma - f_{\hat{\theta}}(z))^2]$$

$$\mathbb{E}[\gamma^2 - 2\gamma f_{\hat{\theta}} + f_{\hat{\theta}}(z)^2]$$

$$\mathbb{E}[\gamma^2] - \mathbb{E}[2\gamma f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

$$\mathbb{E}[(f_\theta(z) + \epsilon)^2] - \mathbb{E}[2(f_\theta(z) + \epsilon)]\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

$$\mathbb{E}[f_\theta(z)^2 + 2f_\theta(z)\epsilon + \epsilon^2] - (2f_\theta(z) + \mathbb{E}[2\epsilon])\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

$$f_\theta(z)^2 + 2f_\theta(z)\mathbb{E}[\epsilon] + \mathbb{E}[\epsilon^2] - (2f_\theta(z) + 2\mathbb{E}[\epsilon])\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

bias^2+ model variance + noise

$$f_\theta(z)^2 + 2f_\theta(z)\mathbb{E}[\epsilon] + \mathbb{E}[\epsilon^2] - (2f_\theta(z) + 2\mathbb{E}[\epsilon])\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

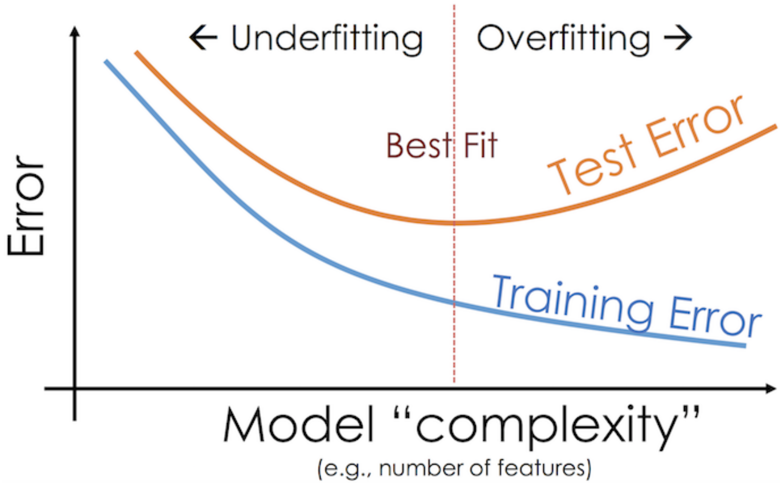
$$f_\theta(z)^2 + \mathrm{Var}(\epsilon) - 2f_\theta(z)\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2]$$

$$f_\theta(z)^2 + \mathrm{Var}(\epsilon) - 2f_\theta(z)\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)^2] - \mathbb{E}[f_{\hat{\theta}}(z)]^2 + \mathbb{E}[f_{\hat{\theta}}(z)]^2$$

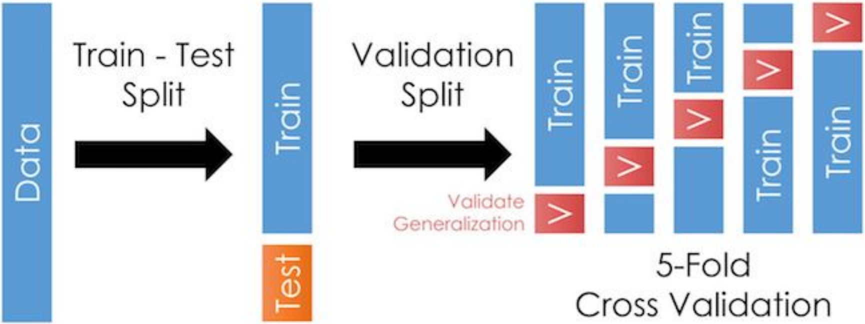
$$f_\theta(z)^2 - 2f_\theta(z)\mathbb{E}[f_{\hat{\theta}}(z)] + \mathbb{E}[f_{\hat{\theta}}(z)]^2 + Var(f_{\hat{\theta}}(z)) + \mathrm{Var}(\epsilon)$$

$$(f_\theta(z) - \mathbb{E}[f_{\hat{\theta}}(z)])^2 + Var(f_{\hat{\theta}}(z)) + \mathrm{Var}(\epsilon)$$

Training Error and Test Error



K-fold Cross-Validation

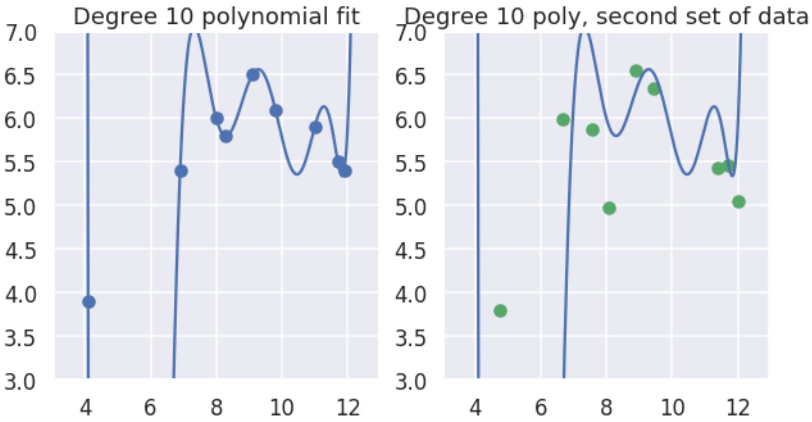


```python
path ='https://github.com/DS-100/textbook/raw/master/content/'
ice = pd.read_csv(path + 'ch/15/icecream.csv')
```

	sweetness	overall
0	4.1	3.9
1	6.9	5.4
2	8.3	5.8
...	...	...
6	11.0	5.9
7	11.7	5.5
8	11.9	5.4

9 rows × 2 columns

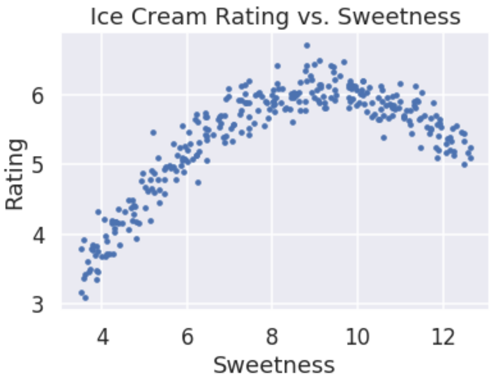
```python
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
trans_ten = PolynomialFeatures(degree=10)
X_ten = trans_ten.fit_transform(ice[['sweetness']])
y = ice['overall']
clf_ten = LinearRegression(fit_intercept=False).fit(X_ten, y)
```



Overall Rating vs Ice Cream Sweetness

	sweetness	overall
0	3.60	3.09
1	3.50	3.17
2	3.69	3.46
...	...	...
6	11.00	5.90
7	11.70	5.50
8	11.90	5.40

309 rows × 2 columns



70/30% Train-Test Split

```python
from sklearn.model_selection import train_test_split
test_size = 92
X_train, X_test, y_train, y_test = train_test_split(ice[['sweetness']],
              ice['overall'], test_size=test_size, random_state=0)
print(f'  Training set size: {len(X_train)}')
print(f'      Test set size: {len(X_test)}')
```

```
Training set size: 217
    Test set size: 92
```

Polynomial Degree from 1 to 10

```
transformers = [PolynomialFeatures(degree=deg)
                for deg in range(1, 11)]
X_train_polys = [transformer.fit_transform(X_train)
                 for transformer in transformers]
# Display the X_train with degree 5 polynomial features
X_train_polys[4]
```

```
([[      1.  ,       8.8 ,      77.44,     681.47,    5996.95,   52773.19],
  [      1.  ,      10.74,     115.35,    1238.83,   13305.07,  142896.44],
  [      1.  ,       9.98,      99.6 ,     994.01,    9920.24,   99003.99],
  ...,
  [      1.  ,       6.79,      46.1 ,     313.05,    2125.59,   14432.74],
  [      1.  ,       5.13,      26.32,     135.01,     692.58,    3552.93],
  [      1.  ,       8.66,      75.  ,     649.46,    5624.34,   48706.78]])
```

5-fold Cross-Validation

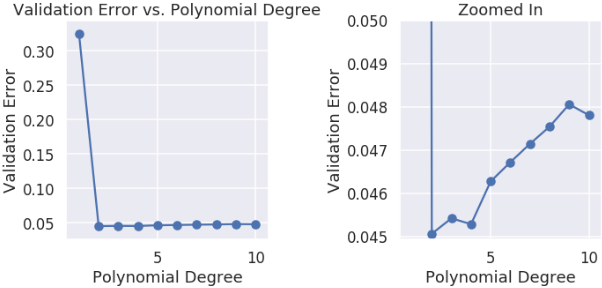
```python
from sklearn.model_selection import KFold
def mse_cost(y_pred, y_actual):
    return np.mean((y_pred - y_actual) ** 2)

def compute_CV_error(model, X_train, Y_train):
    kf = KFold(n_splits=5)
    validation_errors = []
    for train_idx, valid_idx in kf.split(X_train):
        # split the data
        split_X_train, split_X_valid = X_train[train_idx], X_train[valid_idx]
        split_Y_train, split_Y_valid = Y_train.iloc[train_idx], Y_train.iloc[valid_idx]
        # Fit the model on the training split
        model.fit(split_X_train,split_Y_train)
        # Compute the RMSE on the validation split
        error = mse_cost(split_Y_valid,model.predict(split_X_valid))
        validation_errors.append(error)
    #average validation errors
    return np.mean(validation_errors)
```

```python
cv_df = pd.DataFrame({'Validation Error':
 cross_validation_errors}, index=range(1, 11))
cv_df.index.name = 'Degree'
display(cv_df)
```

	Validation Error
Degree	
1	0.32
2	0.05
3	0.05
...	...
8	0.05
9	0.05
10	0.05

10 rows × 1 columns



Final Model

```
best_trans = transformers[1]
best_model = LinearRegression(fit_intercept=False).fit(X_train_polys[1],
                                                        y_train)
training_error = mse_cost(best_model.predict(X_train_polys[1]), y_train)
validation_error = cross_validation_errors[1]
test_error = mse_cost(best_model.predict(best_trans.transform(X_test)),
                                                          y_test)
print('Degree 2 polynomial')
print(f'  Training error: {training_error:0.5f}')
print(f'Validation error: {validation_error:0.5f}')
print(f'      Test error: {test_error:0.5f}')
```

```
Degree 2 polynomial
  Training error: 0.04409
Validation error: 0.04506
      Test error: 0.04698
```