

Deep Learning

Lab 5: Regularization

Chia-Hung Yuan & DataLab

Department of Computer Science,
National Tsing Hua University, Taiwan

Regularization

- Regularization refers to techniques that improve the generalizability of a trained model

Outline

- Scikit-learn
- Learning Theory
 - Error Curves and Model Complexity
 - Learning Curves and Sample Complexity
- Weight Decay
 - Ridge Regression
 - LASSO
- Validation
- Assignment

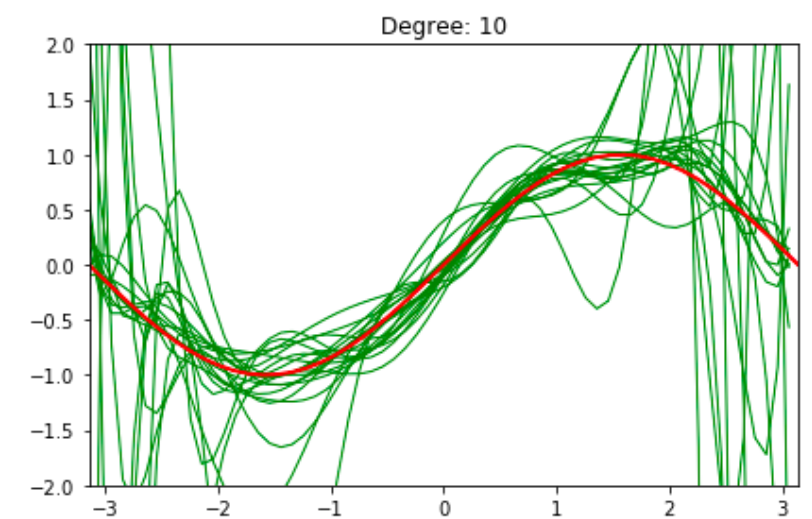
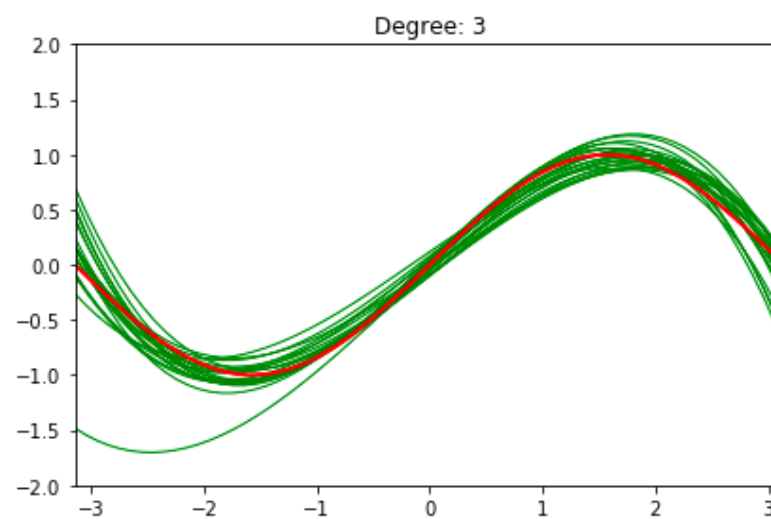
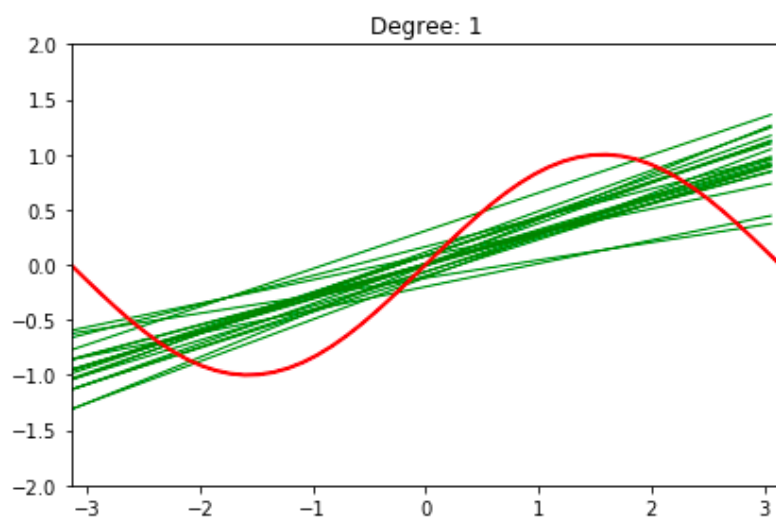
Scikit-learn

- Scikit-learn is a free software machine learning library for the Python programming language
- It features various classification, regression and clustering algorithms
 - including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy
- `pip install scikit-learn` / `conda install scikit-learn`



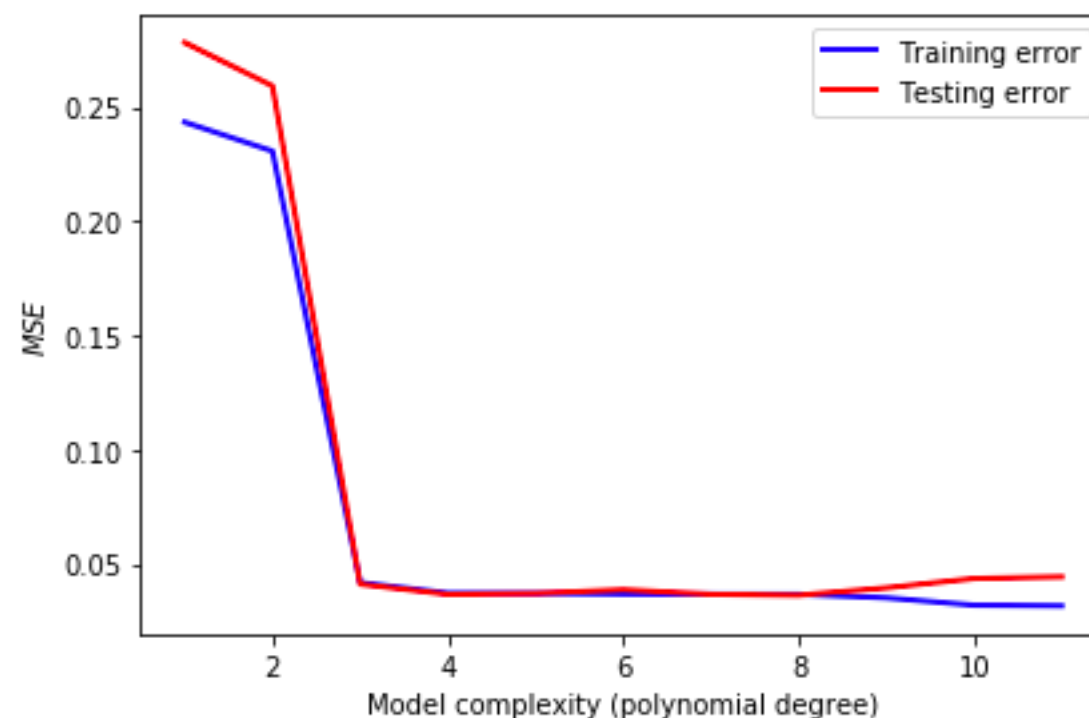
Learning Theory

- Learning theory provides a means to understand the generalizability of the model
- **Model complexity** plays a crucial role
 - Too simple: high bias and underfitting
 - Too complex: high variance and overfitting



Error Curves and Model Complexity

- It is relatively hard to observe the figures showed in the last slide, since normally we will never know the data distribution of ground truth
- Instead, we can get those information by observing the training and testing error

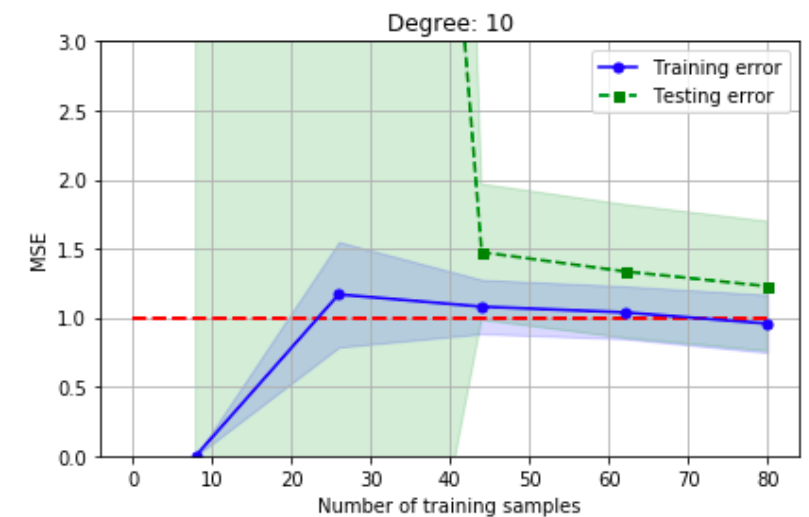
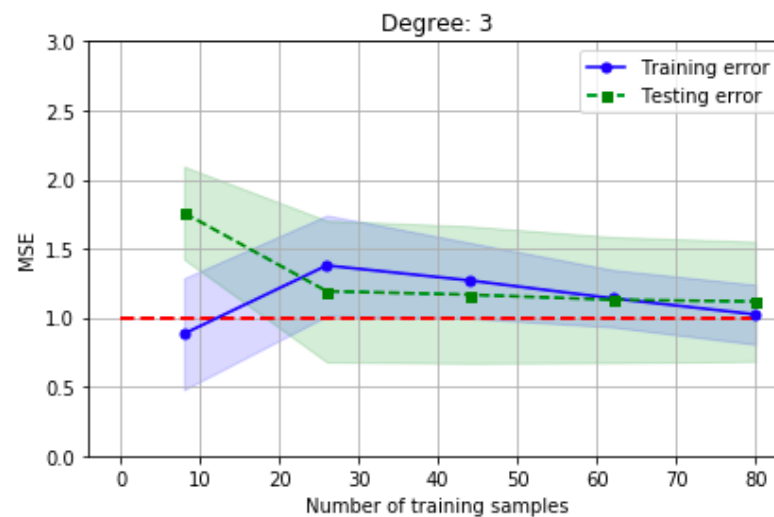
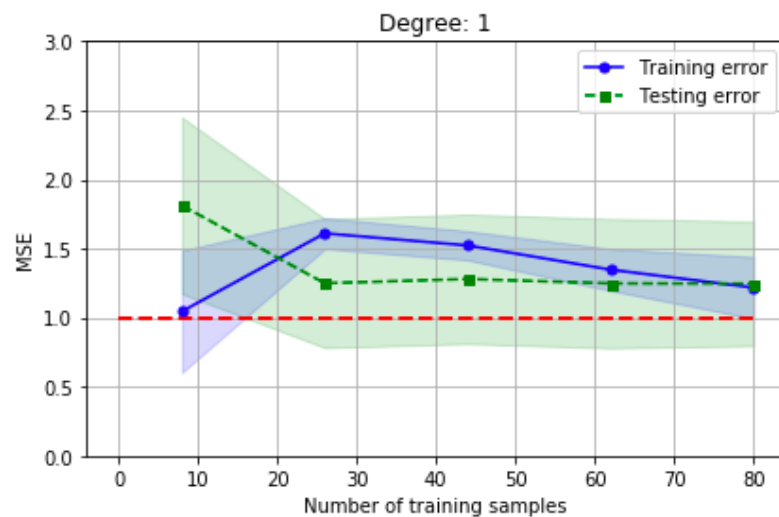


Error Curves and Model Complexity

- Although the error curve visualizes the impact of model complexity, the bias-variance tradeoff holds only when you have sufficient training examples

Learning Curves and Sample Complexity

- The bounding methods of learning theory tell us that a model is likely to overfit regardless of its complexity **when the size of training set is small**. The **learning curves** are a useful tool for understanding how much training examples are sufficient



Weight Decay

- A common regularization approach. The idea is to add a term in the cost function against complexity

- Ridge Regression (L_2)

$$\arg \min_{\mathbf{w}, b} \|\mathbf{y} - (X\mathbf{w} - b\mathbf{1})\|^2 + \alpha \|\mathbf{w}\|^2$$

- LASSO (L_1)

$$\arg \min_{\mathbf{w}, b} \|\mathbf{y} - (X\mathbf{w} - b\mathbf{1})\|^2 + \alpha \|\mathbf{w}\|_1$$

Ridge Regression

- A small value α drastically reduces the testing error. Nevertheless, it's not a good idea to increase α forever, since it will over-shrink the coefficients of w and result in underfitting

$$\arg \min_{w,b} \|y - (Xw - b\mathbf{1})\|^2 + \alpha \|w\|^2$$

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error

poly = PolynomialFeatures(degree=3)
X_poly = poly.fit_transform(X_std)
X_train, X_test, y_train, y_test = train_test_split(
    X_poly, y, test_size=0.3, random_state=0)

for a in [0, 1, 10, 100, 1000]:
    lr_rg = Ridge(alpha=a)
    lr_rg.fit(X_train, y_train)

    y_train_pred = lr_rg.predict(X_train)
    y_test_pred = lr_rg.predict(X_test)

    print('\n[Alpha = %d]' % a )
    print('MSE train: %.2f, test: %.2f' % (
        mean_squared_error(y_train, y_train_pred),
        mean_squared_error(y_test, y_test_pred)))
```

```
[Alpha = 0]
MSE train: 0.00, test: 19958.68

[Alpha = 1]
MSE train: 0.73, test: 23.05

[Alpha = 10]
MSE train: 1.66, test: 16.83

[Alpha = 100]
MSE train: 3.60, test: 15.16

[Alpha = 1000]
MSE train: 8.81, test: 19.22
```

LASSO

- An alternative weight decay approach that can lead to sparse w is the LASSO. Depending on the value of α , certain weights can become zero much faster than others

$$\arg \min_{w,b} \|y - (Xw - b\mathbf{1})\|^2 + \alpha \|w\|_1$$

```
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error

for a in [0.001, 0.01, 0.1, 1, 10]:
    lr_rg = Lasso(alpha=a)
    lr_rg.fit(X_train, y_train)

    y_train_pred = lr_rg.predict(X_train)
    y_test_pred = lr_rg.predict(X_test)

    print('\n[Alpha = %.2f]' % a )
    print('MSE train: %.2f, test: %.2f' % (
        mean_squared_error(y_train, y_train_pred),
        mean_squared_error(y_test, y_test_pred)))
```

```
[Alpha = 0.0000]
MSE train: 0.55, test: 61.02
```

```
[Alpha = 0.0010]
MSE train: 0.64, test: 29.11
```

```
[Alpha = 0.0100]
MSE train: 1.52, test: 19.51
```

```
[Alpha = 0.1000]
MSE train: 4.34, test: 15.52
```

```
[Alpha = 1.0000]
MSE train: 14.33, test: 22.42
```

```
[Alpha = 10.0000]
MSE train: 55.79, test: 53.42
```

Ridge vs LASSO

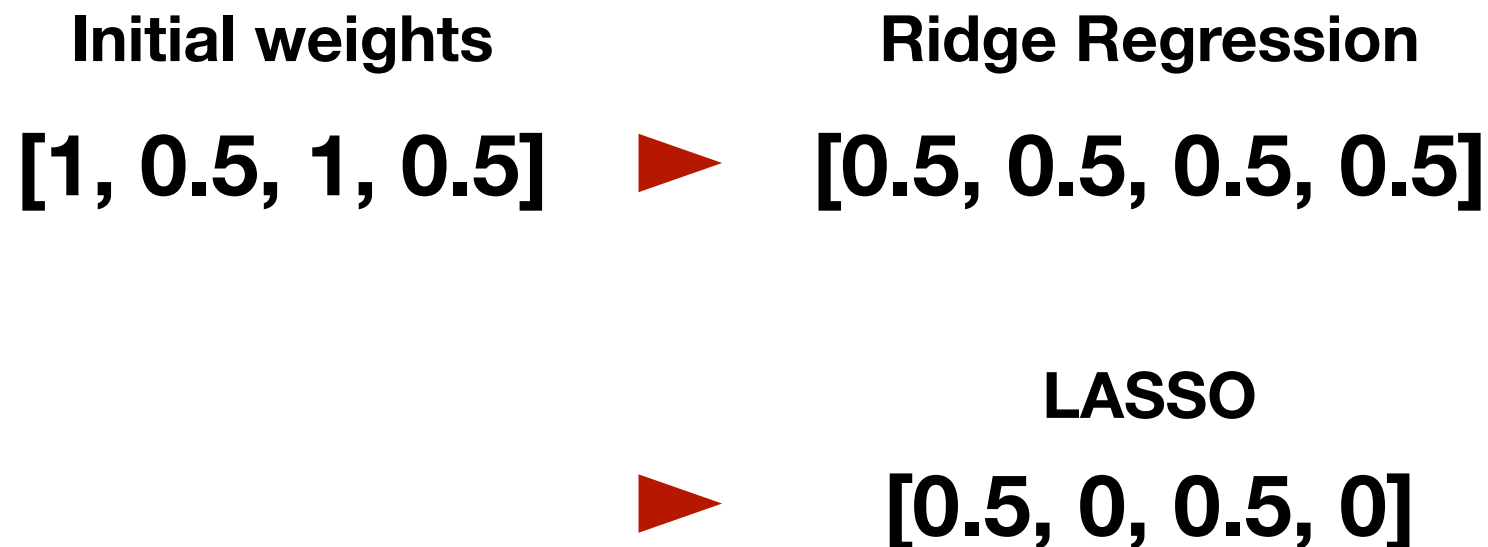
- Why is LASSO sparse?

- Ridge Regression (L_2)

$$\arg \min_{\mathbf{w}, b} \|\mathbf{y} - (X\mathbf{w} - b\mathbf{1})\|^2 + \alpha \|\mathbf{w}\|^2$$

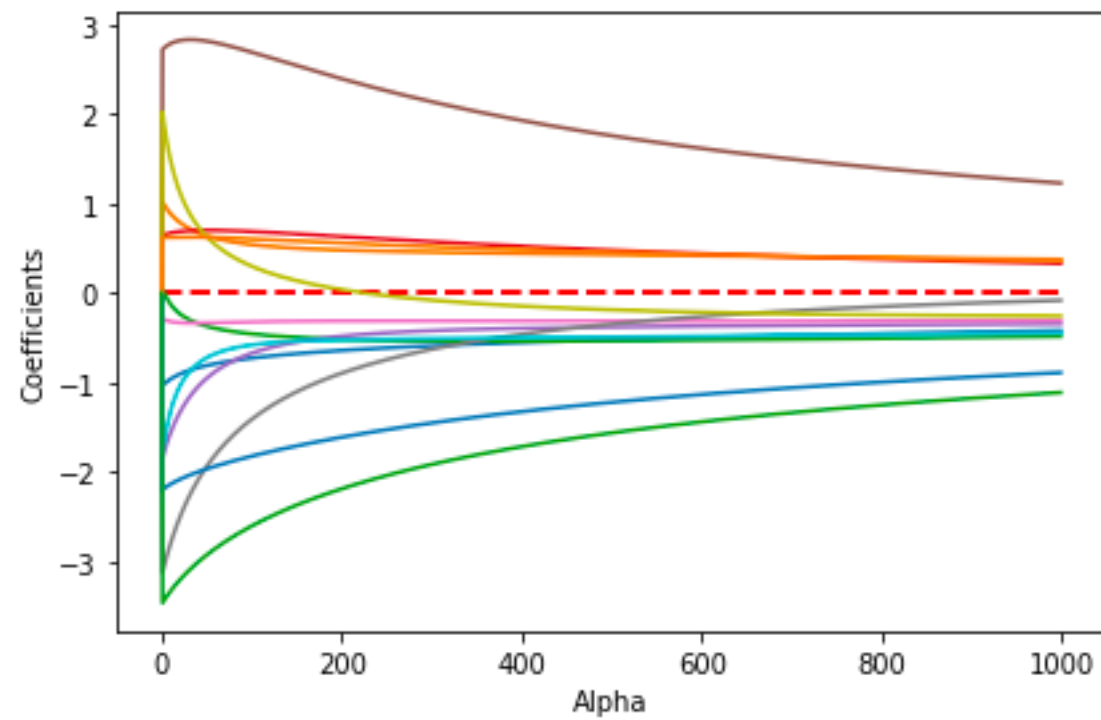
- LASSO (L_1)

$$\arg \min_{\mathbf{w}, b} \|\mathbf{y} - (X\mathbf{w} - b\mathbf{1})\|^2 + \alpha \|\mathbf{w}\|_1$$

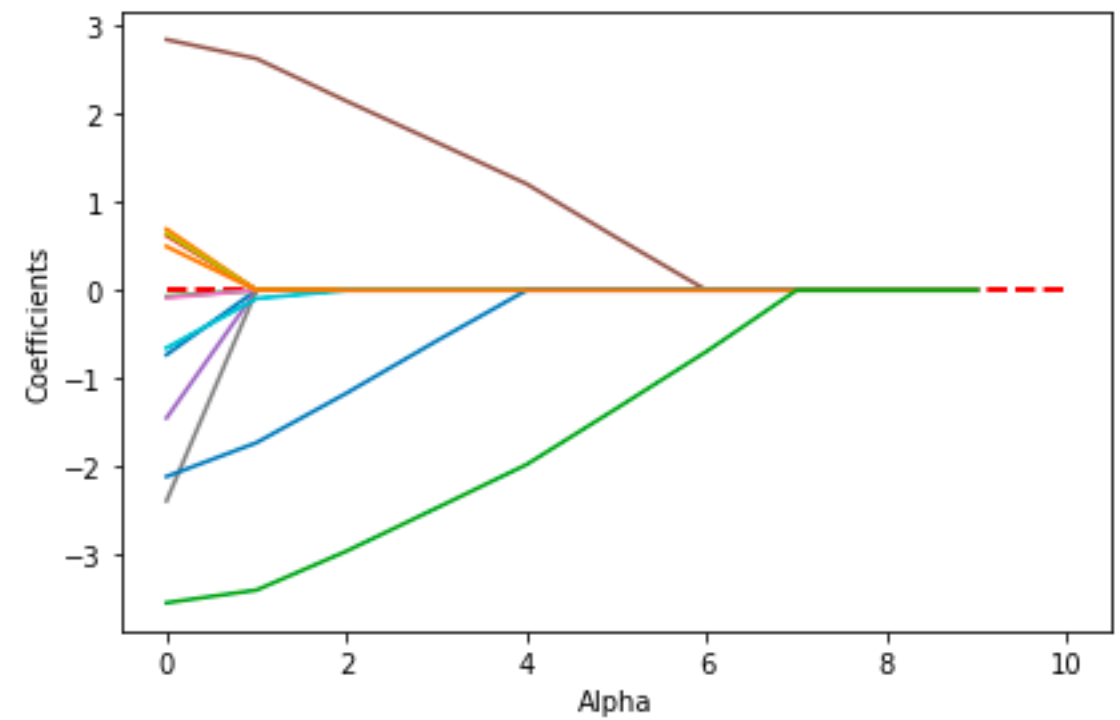


Ridge vs LASSO

Ridge

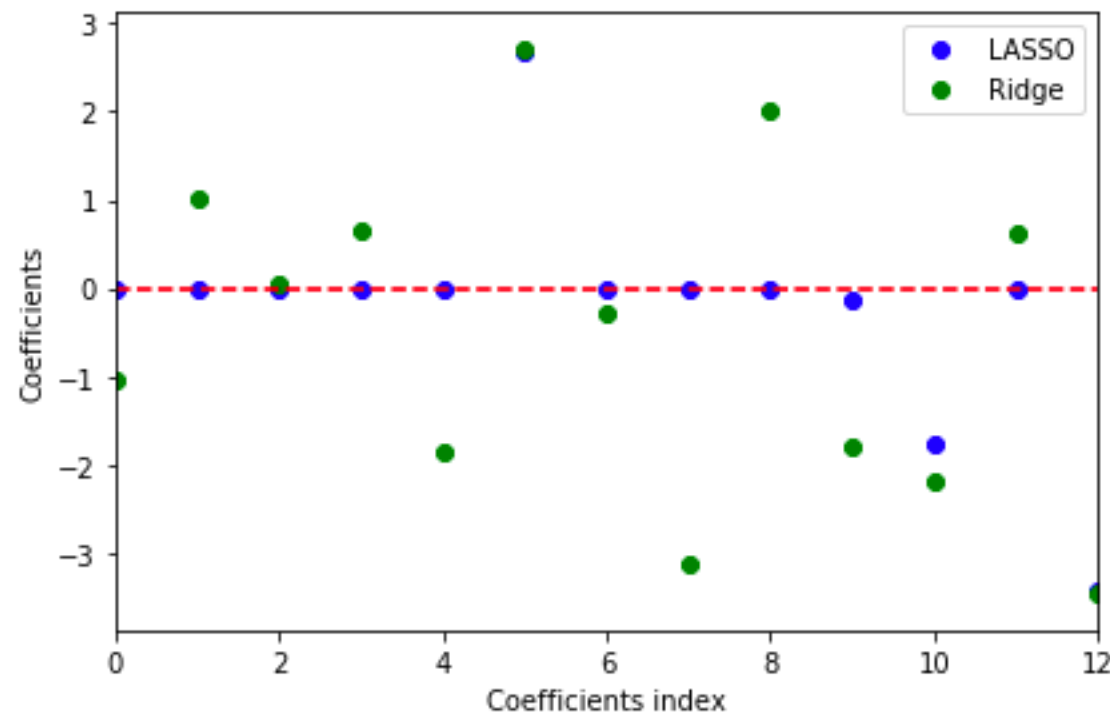


LASSO



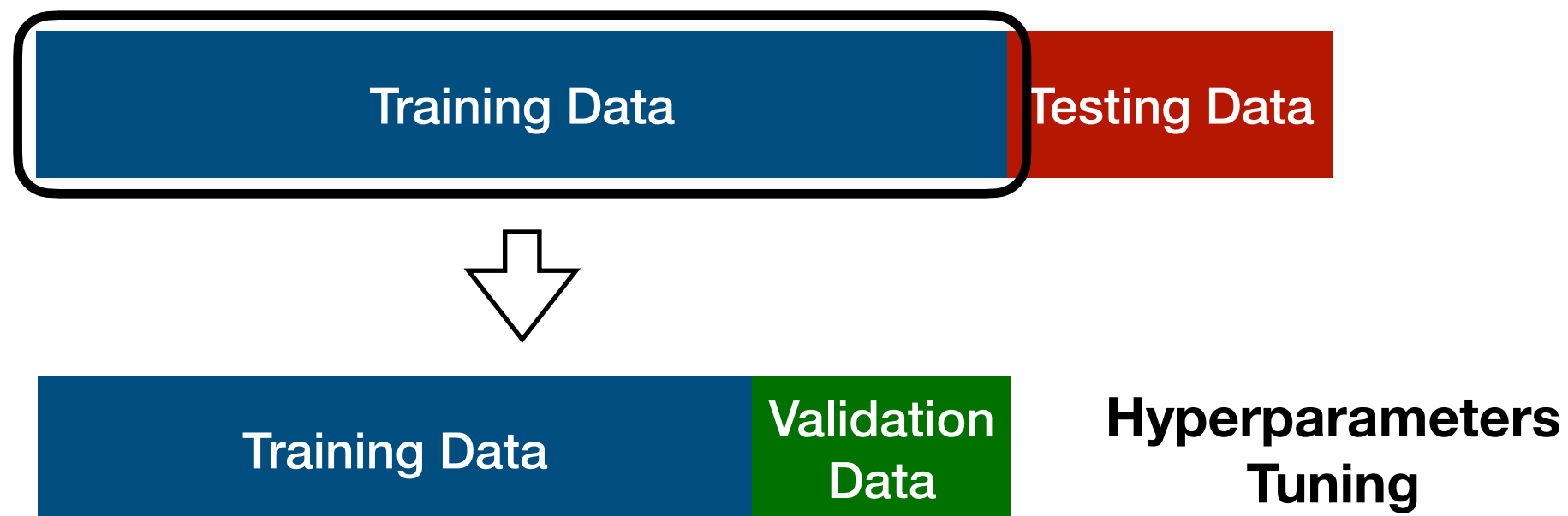
Ridge vs LASSO

- LASSO can also be treated as a supervised feature selection technique when choosing a suitable regularization strength α to make only part of coefficients become exactly zeros



Validation

- Another useful regularization technique that helps us decide the proper value of hyperparameters
- The idea is to split your data into the training, validation, and testing sets and then select the best value based on validation performance
- NOTE: It is important that we should never peek testing data during training



Validation

[Degree = 1]
MSE train: 25.00, valid: 21.43, test: 32.09

[Degree = 2]
MSE train: 9.68, valid: 14.24, test: 20.24

[Degree = 3]
MSE train: 3.38, valid: 17.74, test: 18.63

[Degree = 4]
MSE train: 1.72, valid: 16.67, test: 30.98

[Degree = 5]
MSE train: 0.97, valid: 59.73, test: 57.02

[Degree = 6]
MSE train: 0.60, valid: 1444.08, test: 33189.41