07_exercise_ridge_regression_poly_expansion_robertson

April 2, 2025

Exercise 7: Ridge Regression and Polynomial Feature Expansion

CPSC 381/581: Machine Learning

Yale University

Instructor: Alex Wong

Student: Hailey Robertson

Prerequisites:

1. Enable Google Colaboratory as an app on your Google Drive account
2. Create a new Google Colab notebook, this will also create a "Colab Notebooks" directory under "MyDrive" i.e.

```
/content/drive/MyDrive/Colab Notebooks
```

3. Create the following directory structure in your Google Drive

```
/content/drive/MyDrive/Colab Notebooks/CPSC 381-581: Machine Learning/Exercises
```

4. Move the 04_exercise_ridge_regression_poly_expansion.ipynb into

```
/content/drive/MyDrive/Colab Notebooks/CPSC 381-581: Machine Learning/Exercises
```

so that its absolute path is

```
/content/drive/MyDrive/Colab Notebooks/CPSC 381-581: Machine Learning/Exercises/07_exercise_ri
```

In this exercise, we will optimize a linear and ridge regression with polynomial feature expansion to experiment with over and underfitting.

Submission:

1. Implement all TODOs in the code blocks below.
2. Report your training and testing scores.

```
Experiment 1: Overfitting Linear Regression with Polynomial Expansion
Results for linear regression model with degree-1 polynomial expansion
Training set mean squared error: 44.1064
Testing set mean squared error: 44.3818
Results for linear regression model with degree-2 polynomial expansion
Training set mean squared error: 39.3357
Testing set mean squared error: 40.2318
Results for linear regression model with degree-3 polynomial expansion
```

```
Training set mean squared error: 34.9943
Testing set mean squared error: 40.3880
Results for linear regression model with degree-4 polynomial expansion
Training set mean squared error: 26.4899
Testing set mean squared error: 56.9884
Results for linear regression model with degree-5 polynomial expansion
Training set mean squared error: 7.5671
Testing set mean squared error: 524.3705
Results for linear regression model with degree-6 polynomial expansion
Training set mean squared error: 0.0000
Testing set mean squared error: 1034.3598

Experiment 2: Underfitting Ridge Regression with Large Weight Decay
Results for ridge regression model with weight decay of 1
Training set mean squared error: 44.1351
Testing set mean squared error: 44.3825
Results for ridge regression model with weight decay of 2
Training set mean squared error: 44.1368
Testing set mean squared error: 44.3786
Results for ridge regression model with weight decay of 4
Training set mean squared error: 44.1434
Testing set mean squared error: 44.3743
Results for ridge regression model with weight decay of 8
Training set mean squared error: 44.1681
Testing set mean squared error: 44.3783
Results for ridge regression model with weight decay of 16
Training set mean squared error: 44.2543
Testing set mean squared error: 44.4274
Results for ridge regression model with weight decay of 32
Training set mean squared error: 44.5238
Testing set mean squared error: 44.6363
Results for ridge regression model with weight decay of 64
Training set mean squared error: 45.2289
Testing set mean squared error: 45.2592
Results for ridge regression model with weight decay of 128
Training set mean squared error: 46.6933
Testing set mean squared error: 46.6519
Results for ridge regression model with weight decay of 256
Training set mean squared error: 49.2029
Testing set mean squared error: 49.1832
Results for ridge regression model with weight decay of 512
Training set mean squared error: 53.5784
Testing set mean squared error: 53.8016
Results for ridge regression model with weight decay of 1024
Training set mean squared error: 62.7933
Testing set mean squared error: 63.6086
Results for ridge regression model with weight decay of 2048
Training set mean squared error: 82.6230
```

```
Testing set mean squared error: 84.4319
Results for ridge regression model with weight decay of 4096
Training set mean squared error: 117.0844
Testing set mean squared error: 120.1688
Results for ridge regression model with weight decay of 8192
Training set mean squared error: 160.6582
Testing set mean squared error: 165.0118
Results for ridge regression model with weight decay of 16384
Training set mean squared error: 200.9311
Testing set mean squared error: 206.2856
Results for ridge regression model with weight decay of 32768
Training set mean squared error: 230.0365
Testing set mean squared error: 236.0493

Experiment 3: Ridge Regression with Weight Decay and Polynomial Expansion
Results for ridge regression model with weight decay of 1 for degree-6 polynomial expansion
Training set mean squared error: 30.4866
Testing set mean squared error: 41.3033
Results for ridge regression model with weight decay of 2 for degree-6 polynomial expansion
Training set mean squared error: 31.9424
Testing set mean squared error: 40.5021
Results for ridge regression model with weight decay of 4 for degree-6 polynomial expansion
Training set mean squared error: 33.2689
Testing set mean squared error: 40.0464
Results for ridge regression model with weight decay of 8 for degree-6 polynomial expansion
Training set mean squared error: 34.4947
Testing set mean squared error: 39.8071
Results for ridge regression model with weight decay of 16 for degree-6 polynomial expansion
Training set mean squared error: 35.7264
Testing set mean squared error: 39.8353
Results for ridge regression model with weight decay of 32 for degree-6 polynomial expansion
Training set mean squared error: 37.0339
Testing set mean squared error: 40.2288
Results for ridge regression model with weight decay of 64 for degree-6 polynomial expansion
Training set mean squared error: 38.4471
Testing set mean squared error: 41.0160
Results for ridge regression model with weight decay of 128 for degree-6 polynomial expansion
Training set mean squared error: 40.0672
Testing set mean squared error: 42.2313
Results for ridge regression model with weight decay of 256 for degree-6 polynomial expansion
Training set mean squared error: 42.0621
Testing set mean squared error: 43.9814
Results for ridge regression model with weight decay of 512 for degree-6 polynomial expansion
Training set mean squared error: 44.6313
Testing set mean squared error: 46.4660
Results for ridge regression model with weight decay of 1024 for degree-6 polynomial expansion
Training set mean squared error: 48.2339
Testing set mean squared error: 50.1140
```

```
Results for ridge regression model with weight decay of 2048 for degree-6 polynomial expansion
Training set mean squared error: 53.7208
Testing set mean squared error: 55.5798
Results for ridge regression model with weight decay of 4096 for degree-6 polynomial expansion
Training set mean squared error: 61.7790
Testing set mean squared error: 63.3484
Results for ridge regression model with weight decay of 8192 for degree-6 polynomial expansion
Training set mean squared error: 72.9443
Testing set mean squared error: 74.1290
Results for ridge regression model with weight decay of 16384 for degree-6 polynomial expansio
Training set mean squared error: 89.5759
Testing set mean squared error: 90.7650
Results for ridge regression model with weight decay of 32768 for degree-6 polynomial expansio
Training set mean squared error: 116.0515
Testing set mean squared error: 117.9325
```

3. List any collaborators.

`Collaborators: N/A`

Import packages

[16]:
```python
import numpy as np
import sklearn.datasets as skdata
import sklearn.metrics as skmetrics
import sklearn.preprocessing as skpreprocess
from sklearn.linear_model import LinearRegression as LinearRegressionSciKit
import warnings
from matplotlib import pyplot as plt

warnings.filterwarnings(action='ignore')
np.random.seed = 1
```

Implementation of Ridge Regression with Gradient Descent optimizer

[17]:
```python
class RidgeRegression(object):

    def __init__(self):
        # Define private variables
        self.__weights = None

    def __fit_normal_equation(self, X, y, weight_decay=0):
        '''
        Fits the model to x and y via normal equation

        Arg(s):
            X : numpy
                N x d feature vector
            y : numpy
```

```
            N x 1 ground-truth label
        weight_decay : float
            weight of weight decay term
    '''

    # DONE: Implement the __fit_normal_equation function
    # w* = (X.TX + \lambda I)^{-1} X.Ty
    X_t_X = np.matmul(X.T, X)

    # Identity
    I = np.eye(X.shape[-1])

    # (X.TX + \lambda I)^{-1}
    X_t_X_inverse = np.linalg.inv(X_t_X + weight_decay * I)

    # w* = (X.TX + \lambda I)^{-1} X.Ty
    self.__weights = np.matmul(np.matmul(X_t_X_inverse, X.T), y)

def fit(self, X, y, weight_decay=0, solver='normal_equation'):
    '''
    Fits the model to x and y by solving least squares
    using normal equation

    Arg(s):
        X : numpy[float32]
            N x d feature vector
        y : numpy[float32]
            N ground-truth label
        weight_decay : float
            weight of weight decay term
        solver : str
            solver types: normal_equation
    '''

    y = np.expand_dims(y, axis=1)

    # DONE: Implement the fit function

    if solver == 'normal_equation':
        self.__fit_normal_equation(X, y, weight_decay=weight_decay)
    else:
        raise ValueError('Encountered unsupported solver: {}'.
↪format(solver))

def predict(self, X):
    '''
    Predicts the real value for each feature vector x
```

```
        Arg(s):
            x : numpy[float32]
                N x d feature vector
        Returns:
            numpy[float32] : N x 1 real value vector (\hat{y})
        '''

        # DONE: Implement the predict function

        return np.matmul(X, self.__weights)
```

Helper function for plotting

[18]:
```
def plot_results(axis,
                 x_values,
                 y_values,
                 labels,
                 colors,
                 x_limits,
                 y_limits,
                 x_label,
                 y_label):
    '''
    Plots x and y values using line plot with labels and colors

    Args:
        axis :  pyplot.ax
            matplotlib subplot axis
        x_values : list[numpy[float32]]
            list of numpy array of x values
        y_values : list[numpy[float32]]
            list of numpy array of y values
        labels : str
            list of names for legend
        colors : str
            colors for each line
        x_limits : list[float32]
            min and max values of x axis
        y_limits : list[float32]
            min and max values of y axis
        x_label : list[float32]
            name of x axis
        y_label : list[float32]
            name of y axis
    '''
```

```
    # Iterate through x_values, y_values, labels, and colors and plot them
    # with associated legend
    for x, y, label, color in zip(x_values, y_values, labels, colors):
        axis.plot(x, y, marker='o', color=color, label=label)
        axis.legend(loc='best')

    # Set x and y limits
    axis.set_xlim(x_limits)
    axis.set_ylim(y_limits)

    # Set x and y labels
    axis.set_xlabel(x_label)
    axis.set_ylabel(y_label)
```

Load dataset

[19]:
```
# Create synthetic dataset
X, y = skdata.make_friedman1(n_samples=2000, n_features=8, noise=6)

# Shuffle the dataset based on sample indices
shuffled_indices = np.random.permutation(X.shape[0])

# Choose the first 80% as training set and the rest as testing
train_split_idx = int(0.80 * X.shape[0])

train_indices = shuffled_indices[0:train_split_idx]
test_indices = shuffled_indices[train_split_idx:]

# Select the examples from x and y to construct our training, validation,␣
↪testing sets
X_train, y_train = X[train_indices, :], y[train_indices]
X_test, y_test = X[test_indices, :], y[test_indices]
```

Experiment 1: Demonstrate that linear regression will overfit if we use high degrees of polynomial expansion

[20]:
```
print('Experiment 1: Overfitting Linear Regression with Polynomial Expansion')

# DONE: Initialize a list containing 1 to 6 as the degrees for polynomial␣
↪expansion
degrees = [p for p in range(1, 7)]

# Initialize empty lists to store scores for MSE
scores_mse_linear_overfit_train = []
scores_mse_linear_overfit_test = []

for degree in degrees:
```

```
    # DONE: Initialize polynomial expansion
    poly_transform = skpreprocess.PolynomialFeatures(degree=degree)

    # DONE: Compute the polynomial terms needed for the data
    poly_transform.fit(X_train)

    # DONE: Transform the data by nonlinear mapping
    X_poly_train = poly_transform.transform(X_train)
    X_poly_test = poly_transform.transform(X_test)

    # DONE: Initialize sci-kit linear regression model
    model_linear_overfit = LinearRegressionSciKit()

    # DONE: Train linear regression model
    model_linear_overfit.fit(X_poly_train, y_train)

    print('Results for linear regression model with degree-{} polynomial␣
↪expansion'.format(degree))

    # DONE: Test model on training set
    predictions_train = model_linear_overfit.predict(X_poly_train)
    score_mse_linear_overfit_train = skmetrics.mean_squared_error(y_train,␣
↪predictions_train)
    print('Training set mean squared error: {:.4f}'.
↪format(score_mse_linear_overfit_train))

    # DONE: Save MSE training scores
    scores_mse_linear_overfit_train.append(score_mse_linear_overfit_train)

    # DONE: Test model on testing set
    predictions_test = model_linear_overfit.predict(X_poly_test)
    score_mse_linear_overfit_test = skmetrics.mean_squared_error(y_test,␣
↪predictions_test)
    print('Testing set mean squared error: {:.4f}'.
↪format(score_mse_linear_overfit_test))

    # DONE: Save MSE testing scores
    scores_mse_linear_overfit_test.append(score_mse_linear_overfit_test)


# Convert each scores to NumPy arrays
scores_mse_linear_overfit_train = np.array(scores_mse_linear_overfit_train)
scores_mse_linear_overfit_test = np.array(scores_mse_linear_overfit_test)

# Create figure for training and testing scores for different features
n_experiments = scores_mse_linear_overfit_train.shape[0]
```

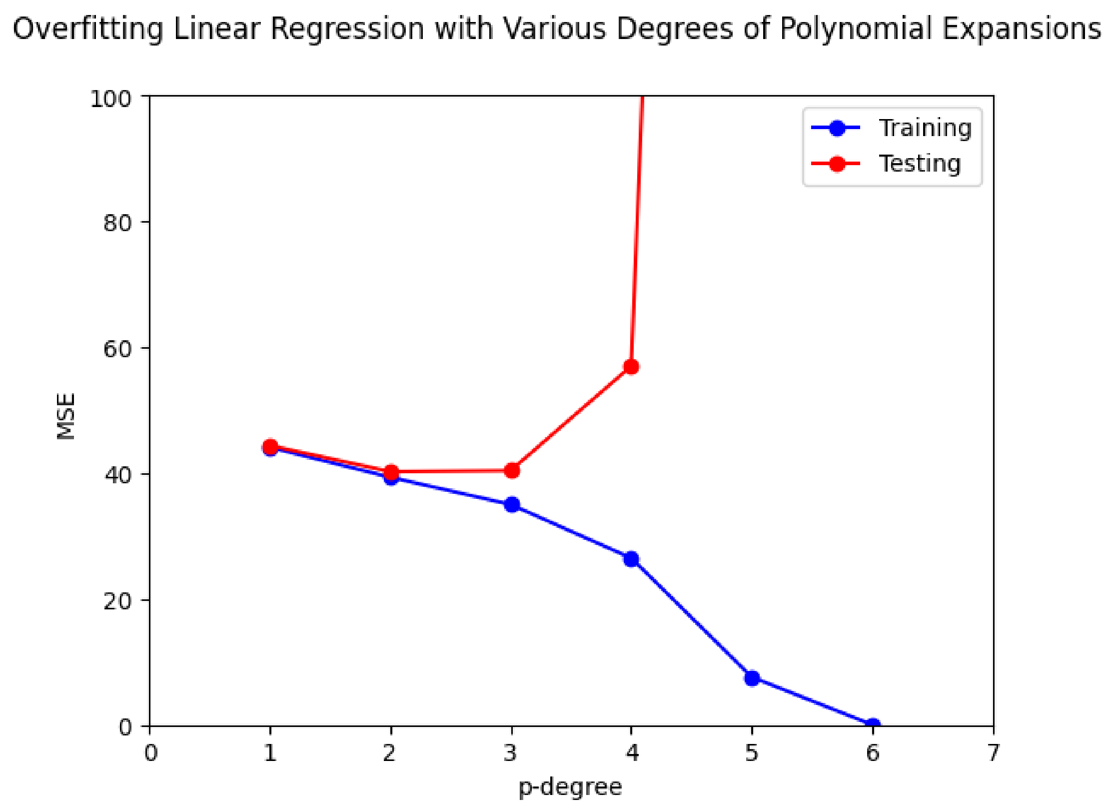
```
labels = ['Training', 'Testing']
colors = ['blue', 'red']

# DONE: Create a subplot of a 1 by 1 figure to plot MSE for training and testing
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

# DONE: Set x and y values
x_values = [range(1, n_experiments + 1)] * n_experiments
y_values = [
    scores_mse_linear_overfit_train,
    scores_mse_linear_overfit_test
]

# DONE: Plot MSE scores for training and testing sets
# Set labels to ['Training', 'Testing'] and colors based on colors defined above
# Set x limits to 0 to number of experiments + 1 and y limits between 0 and 100
# Set x label to 'p-degree' and y label to 'MSE'
plot_results(
    axis=ax,
    x_values=x_values,
    y_values=y_values,
    labels=labels,
    colors=colors,
    x_limits=[0, n_experiments + 1],
    y_limits=[0, 100.0],
    x_label='p-degree',
    y_label='MSE')

# TODO: Create plot title of 'Overfitting Linear Regression with Various␣
 ↪Degrees of Polynomial Expansions'
fig.suptitle('Overfitting Linear Regression with Various Degrees of Polynomial␣
 ↪Expansions')
```

```
Experiment 1: Overfitting Linear Regression with Polynomial Expansion
Results for linear regression model with degree-1 polynomial expansion
Training set mean squared error: 44.1064
Testing set mean squared error: 44.3818
Results for linear regression model with degree-2 polynomial expansion
Training set mean squared error: 39.3357
Testing set mean squared error: 40.2318
Results for linear regression model with degree-3 polynomial expansion
Training set mean squared error: 34.9943
Testing set mean squared error: 40.3880
Results for linear regression model with degree-4 polynomial expansion
Training set mean squared error: 26.4899
Testing set mean squared error: 56.9884
Results for linear regression model with degree-5 polynomial expansion
```

```
Training set mean squared error: 7.5671
Testing set mean squared error: 524.3705
Results for linear regression model with degree-6 polynomial expansion
Training set mean squared error: 0.0000
Testing set mean squared error: 1034.3598
```

[20]:

```
Text(0.5, 0.98, 'Overfitting Linear Regression with Various Degrees of
Polynomial Expansions')
```



Experiment 2: Demonstrate that ridge regression will underfit if we use large weight decay (λ)

[21]:

```python
print('Experiment 2: Underfitting Ridge Regression with Large Weight Decay')

# DONE: Initialize a list containing 1 to 2^15 as the weight for weight decay
weight_decays = [np.power(2, p) for p in range(16)]

# Initialize empty lists to store scores for MSE
scores_mse_ridge_underfit_train = []
scores_mse_ridge_underfit_test = []

for weight_decay in weight_decays:
```

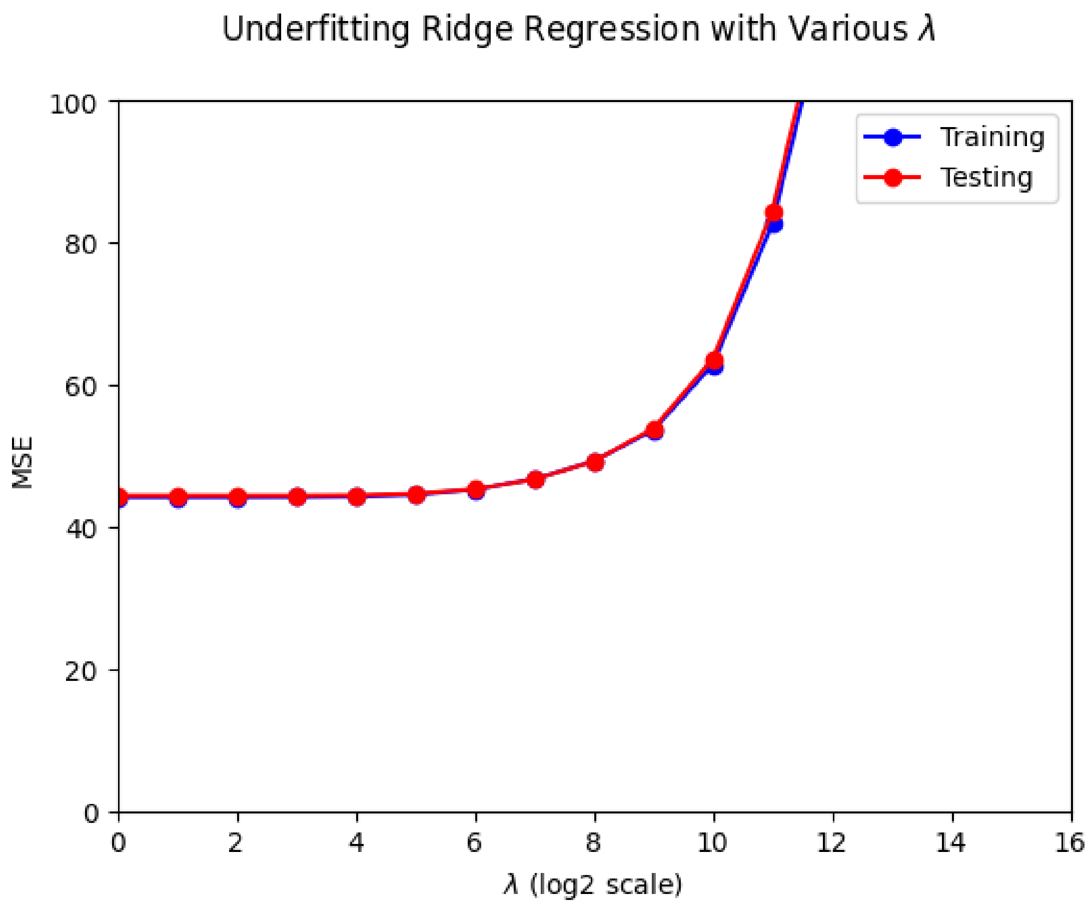
```
    # DONE: Initialize ridge regression model
    model_ridge_underfit = RidgeRegression()

    # DONE: Train ridge regression model
    model_ridge_underfit.fit(X_train, y_train, weight_decay=weight_decay)

    print('Results for ridge regression model with weight decay of {}'.
 ↪format(weight_decay))

    # DONE: Test model on training set
    predictions_train = model_ridge_underfit.predict(X_train)
    score_mse_ridge_underfit_train = skmetrics.mean_squared_error(y_train,␣
 ↪predictions_train)
    print('Training set mean squared error: {:.4f}'.
 ↪format(score_mse_ridge_underfit_train))

    # DONE: Save MSE training scores
    scores_mse_ridge_underfit_train.append(score_mse_ridge_underfit_train)

    # DONE: Test model on testing set
    predictions_test = model_ridge_underfit.predict(X_test)
    score_mse_ridge_underfit_test = skmetrics.mean_squared_error(y_test,␣
 ↪predictions_test)
    print('Testing set mean squared error: {:.4f}'.
 ↪format(score_mse_ridge_underfit_test))

    # DONE: Save MSE testing scores
    scores_mse_ridge_underfit_test.append(score_mse_ridge_underfit_test)


# Convert each scores to NumPy arrays
scores_mse_ridge_underfit_train = np.array(scores_mse_ridge_underfit_train)
scores_mse_ridge_underfit_test = np.array(scores_mse_ridge_underfit_test)

# Create figure for training, validation and testing scores for different␣
 ↪features
n_experiments = scores_mse_ridge_underfit_train.shape[0]

labels = ['Training', 'Testing']
colors = ['blue', 'red']

# DONE: Create a subplot of a 1 by 1 figure to plot MSE for training and testing
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

# DONE: Set x values (weight_decays in log base2 scale) and y values (MSE)
```

```python
x_values = [np.log2(weight_decays)] * 2
y_values = [
    scores_mse_ridge_underfit_train,
    scores_mse_ridge_underfit_test
]

# DONE: Plot MSE scores for training and testing sets
# Set labels to ['Training', 'Testing'] and colors based on colors defined above
# Set x limits to 0 to log of highest weight_decays + 1 and y limits between 0
 ↪and 100
# Set x label to r'$\lambda$ (log2 scale)' and y label to 'MSE'
plot_results(
    axis=ax,
    x_values=x_values,
    y_values=y_values,
    labels=labels,
    colors=colors,
    x_limits=[0, np.log2(weight_decays[-1]) + 1],
    y_limits=[0, 100.0],
    x_label=r'$\lambda$ (log2 scale)',
    y_label='MSE')


# DONE: Create plot title of r'Underfitting Ridge Regression with Various
 ↪$\lambda$'
fig.suptitle(r'Underfitting Ridge Regression with Various $\lambda$')
```

```
Experiment 2: Underfitting Ridge Regression with Large Weight Decay
Results for ridge regression model with weight decay of 1
Training set mean squared error: 44.1351
Testing set mean squared error: 44.3825
Results for ridge regression model with weight decay of 2
Training set mean squared error: 44.1368
Testing set mean squared error: 44.3786
Results for ridge regression model with weight decay of 4
Training set mean squared error: 44.1434
Testing set mean squared error: 44.3743
Results for ridge regression model with weight decay of 8
Training set mean squared error: 44.1681
Testing set mean squared error: 44.3783
Results for ridge regression model with weight decay of 16
Training set mean squared error: 44.2543
Testing set mean squared error: 44.4274
Results for ridge regression model with weight decay of 32
Training set mean squared error: 44.5238
Testing set mean squared error: 44.6363
Results for ridge regression model with weight decay of 64
```

```
Training set mean squared error: 45.2289
Testing set mean squared error: 45.2592
Results for ridge regression model with weight decay of 128
Training set mean squared error: 46.6933
Testing set mean squared error: 46.6519
Results for ridge regression model with weight decay of 256
Training set mean squared error: 49.2029
Testing set mean squared error: 49.1832
Results for ridge regression model with weight decay of 512
Training set mean squared error: 53.5784
Testing set mean squared error: 53.8016
Results for ridge regression model with weight decay of 1024
Training set mean squared error: 62.7933
Testing set mean squared error: 63.6086
Results for ridge regression model with weight decay of 2048
Training set mean squared error: 82.6230
Testing set mean squared error: 84.4319
Results for ridge regression model with weight decay of 4096
Training set mean squared error: 117.0844
Testing set mean squared error: 120.1688
Results for ridge regression model with weight decay of 8192
Training set mean squared error: 160.6582
Testing set mean squared error: 165.0118
Results for ridge regression model with weight decay of 16384
Training set mean squared error: 200.9311
Testing set mean squared error: 206.2856
Results for ridge regression model with weight decay of 32768
Training set mean squared error: 230.0365
Testing set mean squared error: 236.0493
```

```
[21]: Text(0.5, 0.98, 'Underfitting Ridge Regression with Various $\\lambda$')
```



Experiment 3: Demonstrate that ridge regression with various λ prevents overfitting when using polynomial expansion

[22]:
```
print(r'Experiment 3: Ridge Regression with Weight Decay and Polynomial␣
↪Expansion')

# Set polynomial expansion
degree = 6

# DONE: Initialize a list containing 1 to 2^15 as the weight for weight decay
weight_decays = [np.power(2, p) for p in range(16)]

# DONE: Initialize polynomial expansion
poly_transform = skpreprocess.PolynomialFeatures(degree=degree)

# DONE: Compute the polynomial terms needed for the data
poly_transform.fit(X_train)
```

```python
# DONE: Transform the data by nonlinear mapping
x_poly_train = poly_transform.transform(X_train)
x_poly_test = poly_transform.transform(X_test)

# Initialize empty lists to store scores for MSE
scores_mse_ridge_poly_train = []
scores_mse_ridge_poly_test = []

for weight_decay in weight_decays:

    # DONE: Initialize ridge regression model
    model_ridge_poly = RidgeRegression()

    # DONE: Train ridge regression model
    model_ridge_poly.fit(x_poly_train, y_train, weight_decay=weight_decay)

    print('Results for ridge regression model with weight decay of {} for␣
 ↪degree-{} polynomial expansion'.format(weight_decay, degree))

    # DONE: Test model on training set
    predictions_train = model_ridge_poly.predict(x_poly_train)
    score_mse_ridge_poly_train = skmetrics.mean_squared_error(y_train,␣
 ↪predictions_train)
    print('Training set mean squared error: {:.4f}'.
 ↪format(score_mse_ridge_poly_train))

    # DONE: Save MSE training scores
    scores_mse_ridge_poly_train.append(score_mse_ridge_poly_train)


    # DONE: Test model on testing set
    predictions_test = model_ridge_poly.predict(x_poly_test)
    score_mse_ridge_poly_test = skmetrics.mean_squared_error(y_test,␣
 ↪predictions_test)
    print('Testing set mean squared error: {:.4f}'.
 ↪format(score_mse_ridge_poly_test))

    # DONE: Save MSE testing scores
    scores_mse_ridge_poly_test.append(score_mse_ridge_poly_test)


# Convert each scores to NumPy arrays
scores_mse_ridge_poly_train = np.array(scores_mse_ridge_poly_train)
scores_mse_ridge_poly_test = np.array(scores_mse_ridge_poly_test)

# Create figure for training and testing scores for different features
n_experiments = scores_mse_ridge_poly_train.shape[0]
```

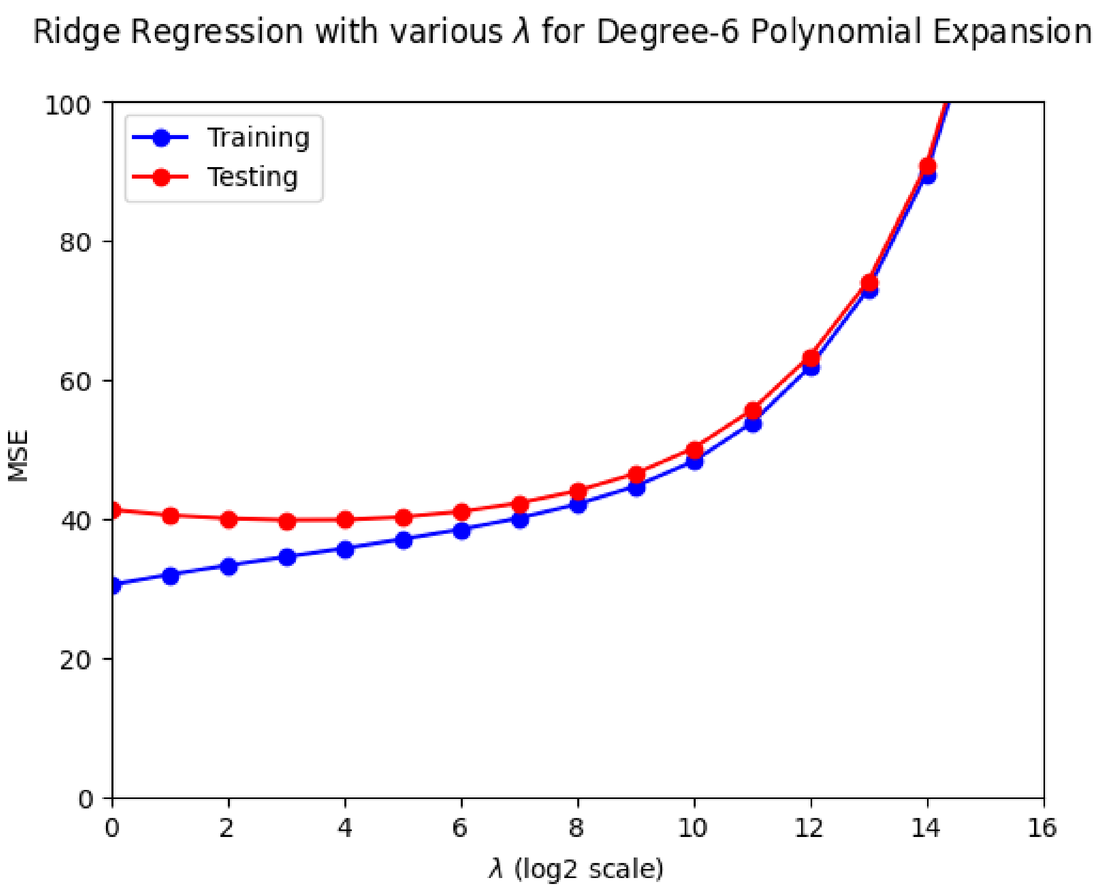
```
labels = ['Training', 'Testing']
colors = ['blue', 'red']

# DONE: Create the first subplot of a 1 by 1 figure to plot MSE for training␣
 ↪and testing
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)

# DONE: Set x values (weight_decays in log base2 scale) and y values (MSE)
x_values = [np.log2(weight_decays)] * 2
y_values = [
    scores_mse_ridge_poly_train,
    scores_mse_ridge_poly_test
]

# DONE: Plot MSE scores for training and testing sets
# Set labels to ['Training', 'Testing'] and colors based on colors defined above
# Set x limits to 0 to log of highest weight_decays + 1 and y limits between 0␣
 ↪and 100
# Set x label to r'$\lambda$ (log2 scale)' and y label to 'MSE'
plot_results(
    axis=ax,
    x_values=x_values,
    y_values=y_values,
    labels=labels,
    colors=colors,
    x_limits=[0, np.log2(weight_decays[-1]) + 1],
    y_limits=[0, 100.0],
    x_label=r'$\lambda$ (log2 scale)',
    y_label='MSE'
)


# DONE: Create plot title of r'Ridge Regression with various $\lambda$ for␣
 ↪Degree-{} Polynomial Expansion'.format(degree)
fig.suptitle(
    r'Ridge Regression with various $\lambda$ for Degree-{} Polynomial␣
 ↪Expansion'.format(degree)
)
```

```
Experiment 3: Ridge Regression with Weight Decay and Polynomial Expansion
Results for ridge regression model with weight decay of 1 for degree-6
polynomial expansion
Training set mean squared error: 30.4866
Testing set mean squared error: 41.3033
Results for ridge regression model with weight decay of 2 for degree-6
```

```
polynomial expansion
Training set mean squared error: 31.9424
Testing set mean squared error: 40.5021
Results for ridge regression model with weight decay of 4 for degree-6
polynomial expansion
Training set mean squared error: 33.2689
Testing set mean squared error: 40.0464
Results for ridge regression model with weight decay of 8 for degree-6
polynomial expansion
Training set mean squared error: 34.4947
Testing set mean squared error: 39.8071
Results for ridge regression model with weight decay of 16 for degree-6
polynomial expansion
Training set mean squared error: 35.7264
Testing set mean squared error: 39.8353
Results for ridge regression model with weight decay of 32 for degree-6
polynomial expansion
Training set mean squared error: 37.0339
Testing set mean squared error: 40.2288
Results for ridge regression model with weight decay of 64 for degree-6
polynomial expansion
Training set mean squared error: 38.4471
Testing set mean squared error: 41.0160
Results for ridge regression model with weight decay of 128 for degree-6
polynomial expansion
Training set mean squared error: 40.0672
Testing set mean squared error: 42.2313
Results for ridge regression model with weight decay of 256 for degree-6
polynomial expansion
Training set mean squared error: 42.0621
Testing set mean squared error: 43.9814
Results for ridge regression model with weight decay of 512 for degree-6
polynomial expansion
Training set mean squared error: 44.6313
Testing set mean squared error: 46.4660
Results for ridge regression model with weight decay of 1024 for degree-6
polynomial expansion
Training set mean squared error: 48.2339
Testing set mean squared error: 50.1140
Results for ridge regression model with weight decay of 2048 for degree-6
polynomial expansion
Training set mean squared error: 53.7208
Testing set mean squared error: 55.5798
Results for ridge regression model with weight decay of 4096 for degree-6
polynomial expansion
Training set mean squared error: 61.7790
Testing set mean squared error: 63.3484
Results for ridge regression model with weight decay of 8192 for degree-6
```

```
polynomial expansion
Training set mean squared error: 72.9443
Testing set mean squared error: 74.1290
Results for ridge regression model with weight decay of 16384 for degree-6
polynomial expansion
Training set mean squared error: 89.5759
Testing set mean squared error: 90.7650
Results for ridge regression model with weight decay of 32768 for degree-6
polynomial expansion
Training set mean squared error: 116.0515
Testing set mean squared error: 117.9325
```

[22]:
```
Text(0.5, 0.98, 'Ridge Regression with various $\\lambda$ for Degree-6
Polynomial Expansion')
```



[ ]: