Exercise 7: Ridge Regression and Polynomial Feature Expansion

CPSC 381/581: Machine Learning

Yale University

Instructor: Alex Wong

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_ridge_regression_poly_expansion.ipynb
```

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.

```
Report training and testing scores here.
```

Experiment 1: Overfitting Linear Regression with Polynomial Expansion Results for linear regression model with degree-1 polynomial expansion Training set mean squared error: 39.8309 Testing set mean squared error: 44.9218 Results for linear regression model with degree-2 polynomial expansion Training set mean squared error: 35.2415 Testing set mean squared error: 39.1237 Results for linear regression model with degree-3 polynomial expansion Training set mean squared error: 31.8169 Testing set mean squared error: 39.6756 Results for linear regression model with degree-4 polynomial expansion Training set mean squared error: 24.2178 Testing set mean squared error: 59.1120 Results for linear regression model with degree-5 polynomial expansion Training set mean squared error: 6.1305 Testing set mean squared error: 469.2289 Results for linear regression model with degree-6 polynomial expansion Training set mean squared error: 0.0000 Testing set mean squared error: 1108.0242

Experiment 2: Underfitting Ridge Regression with Large Weight Decay Results for ridge regression model with weight decay of 1 Training set mean squared error: 39.8482 Testing set mean squared error: 44.9514 Results for ridge regression model with weight decay of 2 Training set mean squared error: 39.8498 Testing set mean squared error: 44.9577 Results for ridge regression model with weight decay of 4 Training set mean squared error: 39.8558 Testing set mean squared error: 44.9728 Results for ridge regression model with weight decay of 8 Training set mean squared error: 39.8783 Testing set mean squared error: 45.0126 Results for ridge regression model with weight decay of 16 Training set mean squared error: 39.9575 Testing set mean squared error: 45.1229 Results for ridge regression model with weight decay of 32 Training set mean squared error: 40.2074 Testing set mean squared error: 45.4245 Results for ridge regression model with weight decay of 64 Training set mean squared error: 40.8711 Testing set mean squared error: 46.1633 Results for ridge regression model with weight decay of 128 Training set mean squared error: 42.2802 Testing set mean squared error: 47.6655 Results for ridge regression model with weight decay of 256 Training set mean squared error: 44.7632 Testing set mean squared error: 50.2544 Results for ridge regression model with weight decay of 512 Training set mean squared error: 49.2108 Testing set mean squared error: 54.8368 Results for ridge regression model with weight decay of 1024 Training set mean squared error: 58.6890 Testing set mean squared error: 64.5329 Results for ridge regression model with weight decay of 2048 Training set mean squared error: 79.0307 Testing set mean squared error: 85.2557 Results for ridge regression model with weight decay of 4096 Training set mean squared error: 114.0822 Testing set mean squared error: 120.8847 Results for ridge regression model with weight decay of 8192 Training set mean squared error: 157.9772 Testing set mean squared error: 165.4517 Results for ridge regression model with weight decay of 16384 Training set mean squared error: 198.2125 Testing set mean squared error: 206.2790 Results for ridge regression model with weight decay of 32768 Training set mean squared error: 227.1206 Testing set mean squared error: 235.6035

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: 28.0858 Testing set mean squared error: 40.4158 Results for ridge regression model with weight decay of 2 for degree-6 polynomial expansion Training set mean squared error: 29.5419 Testing set mean squared error: 39.8220 Results for ridge regression model with weight decay of 4 for degree-6 polynomial expansion Training set mean squared error: 30.8583 Testing set mean squared error: 39.6788 Results for ridge regression model with weight decay of 8 for degree-6 polynomial expansion Training set mean squared error: 32.0425 Testing set mean squared error: 39.7949 Results for ridge regression model with weight decay of 16 for degree-6 polynomial expansion Training set mean squared error: 33.1490 Testing set mean squared error: 40.0422 Results for ridge regression model with weight decay of 32 for degree-6 polynomial expansion Training set mean squared error: 34.2403 Testing set mean squared error: 40.4178 Results for ridge regression model with weight decay of 64 for degree-6 polynomial expansion Training set mean squared error: 35.4209 Testing set mean squared error: 41.0601 Results for ridge regression model with weight decay of 128 for degree-6 polynomial expansion Training set mean squared error: 36.8756 Testing set mean squared error: 42.1782 Results for ridge regression model with weight decay of 256 for degree-6 polynomial expansion Training set mean squared error: 38.7897 Testing set mean squared error: 43.9155 Results for ridge regression model with weight decay of 512 for degree-6 polynomial expansion Training set mean squared error: 41.3499 Testing set mean squared error: 46.3846 Results for ridge regression model with weight decay of 1024 for degree-6 polynomial expansion Training set mean squared error: 45.0579 Testing set mean squared error:

50.0273 Results for ridge regression model with weight decay of 2048 for degree-6 polynomial expansion Training set mean squared error: 50.7874 Testing set mean squared error: 55.6868 Results for ridge regression model with weight decay of 4096 for degree-6 polynomial expansion Training set mean squared error: 59.0832 Testing set mean squared error: 63.9740 Results for ridge regression model with weight decay of 8192 for degree-6 polynomial expansion Training set mean squared error: 70.3812 Testing set mean squared error: 75.5126 Results for ridge regression model with weight decay of 16384 for degree-6 polynomial expansion Training set mean squared error: 87.2439 Testing set mean squared error: 92.9971 Results for ridge regression model with weight decay of 32768 for degree-6 polynomial expansion Training set mean squared error: 114.1872 Testing set mean squared error: 120.8370 3. List any collaborators.

None

Import packages

In [ ]:
```
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

In [ ]:
```
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
        '''

        # TODO: 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)

        # TODO: 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})
        '''

        # TODO: Implement the predict function

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

Helper function for plotting

In [ ]:

```
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

In [ ]:
```
# 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 set
X_train, y_train = X[train_indices, :], y[train_indices]
X_test, y_test = X[test_indices, :], y[test_indices]
```

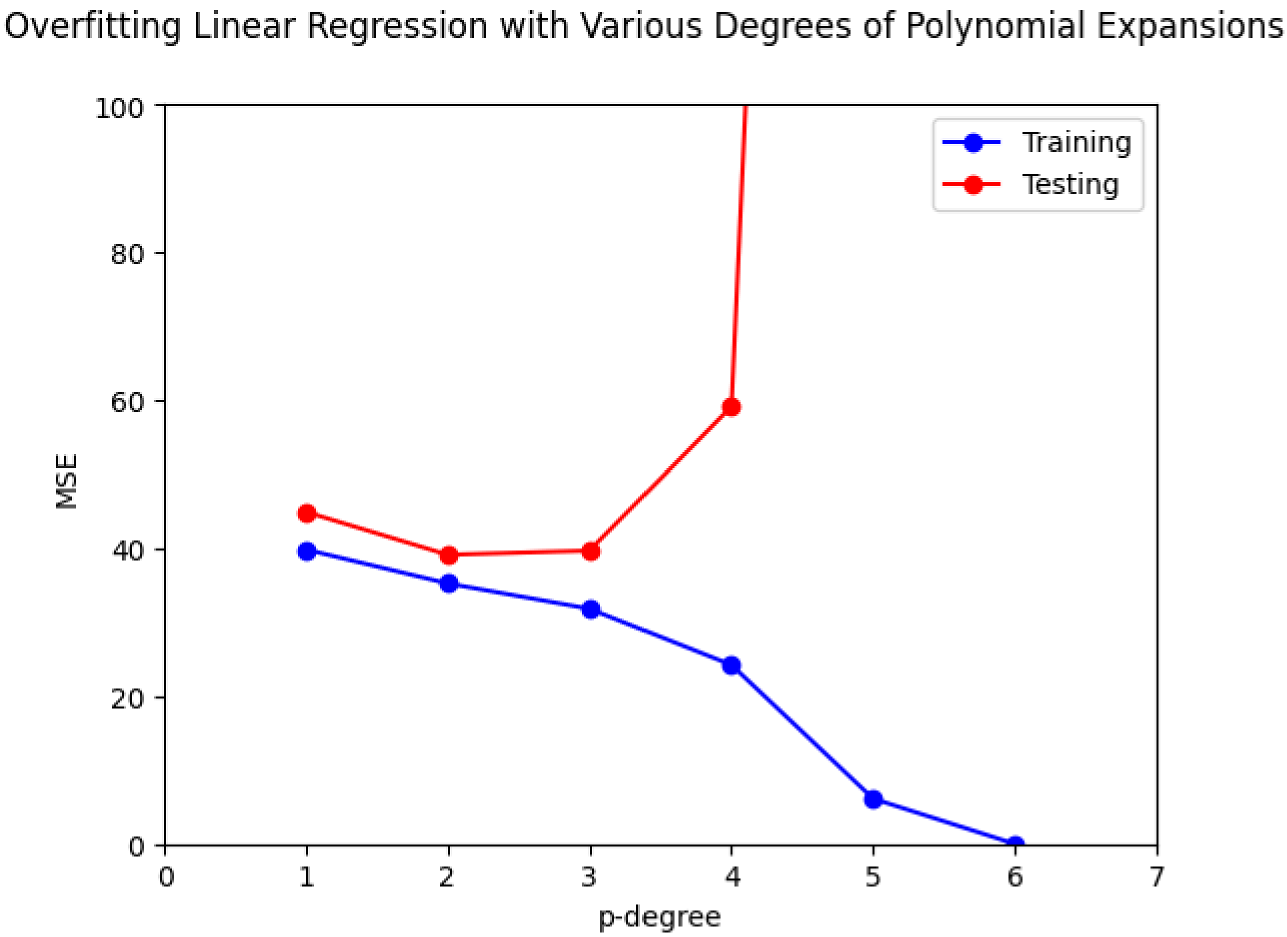
Experiment 1: Demonstrate that linear regression will overfit if we use high degrees of polynomial expansion

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

# TODO: 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:

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

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

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

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

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

    print('Results for linear regression model with degree-{} polynomial expansion'.f

    # TODO: 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, prediction
    print('Training set mean squared error: {:.4f}'.format(score_mse_linear_overfit_t

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

    # TODO: 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_
    print('Testing set mean squared error: {:.4f}'.format(score_mse_linear_overfit_te

    # TODO: 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']

# TODO: 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)

# TODO: 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
]

# TODO: 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 P
fig.suptitle('Overfitting Linear Regression with Various Degrees of Polynomial Expans
```

```
Experiment 1: Overfitting Linear Regression with Polynomial Expansion
Results for linear regression model with degree-1 polynomial expansion
Training set mean squared error: 39.8309
Testing set mean squared error: 44.9218
Results for linear regression model with degree-2 polynomial expansion
Training set mean squared error: 35.2415
Testing set mean squared error: 39.1237
Results for linear regression model with degree-3 polynomial expansion
Training set mean squared error: 31.8169
Testing set mean squared error: 39.6756
Results for linear regression model with degree-4 polynomial expansion
Training set mean squared error: 24.2178
Testing set mean squared error: 59.1120
Results for linear regression model with degree-5 polynomial expansion
Training set mean squared error: 6.1305
Testing set mean squared error: 469.2289
Results for linear regression model with degree-6 polynomial expansion
Training set mean squared error: 0.0000
Testing set mean squared error: 1108.0242
```

Out[ ]:
```
Text(0.5, 0.98, 'Overfitting Linear Regression with Various Degrees of Polynomial Ex
pansions')
```



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

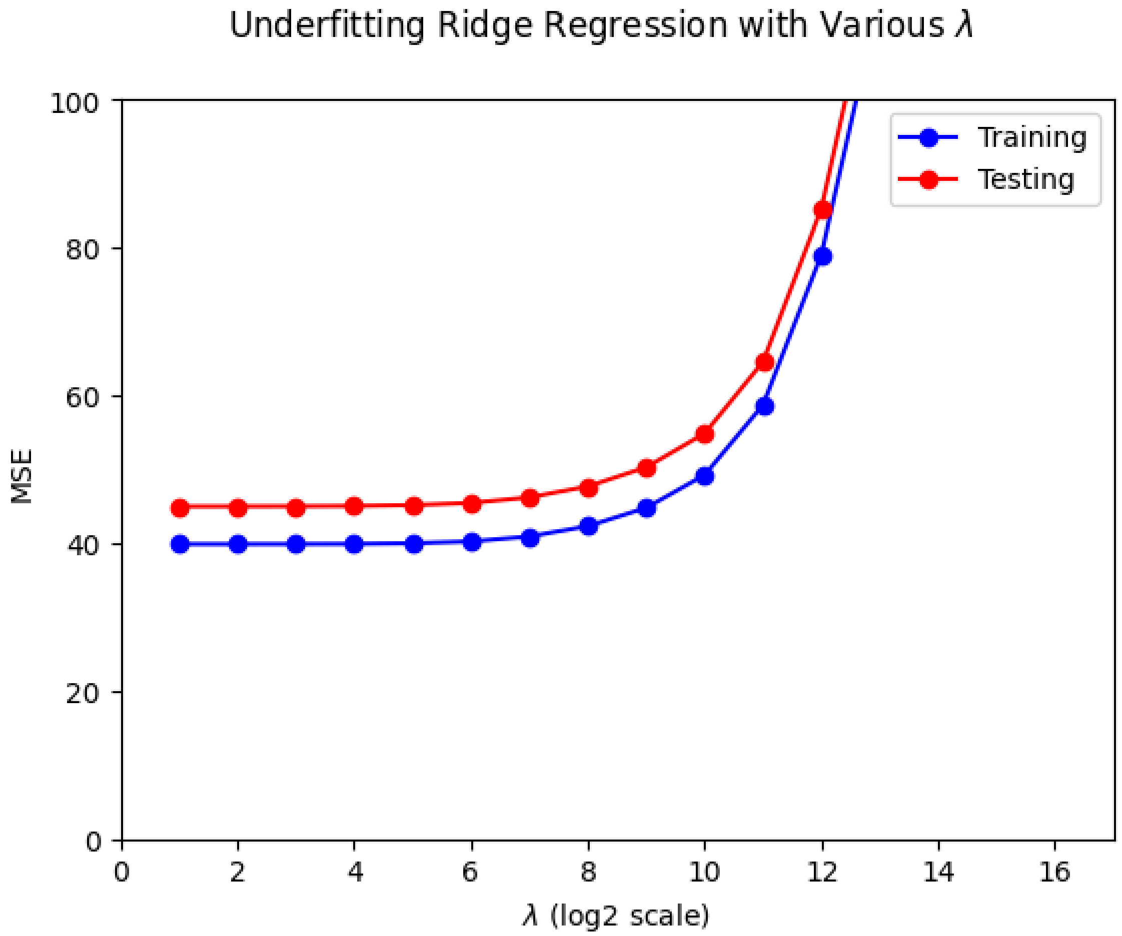
In [ ]:
```
print('Experiment 2: Underfitting Ridge Regression with Large Weight Decay')

# TODO: 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:

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

```
    # TODO: 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_

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

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


    # TODO: 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_
    print('Testing set mean squared error: {:.4f}'.format(score_mse_ridge_underfit_te

    # TODO: 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']

# TODO: 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)

# TODO: Set x values (weight_decays in log base2 scale) and y values (MSE)
x_values = [range(1, n_experiments + 1)] * n_experiments
y_values = [
    scores_mse_ridge_underfit_train,
    scores_mse_ridge_underfit_test
]


# TODO: 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 10
# 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, n_experiments + 1],
    y_limits=[0, 100.0],
    x_label=r'$\lambda$ (log2 scale)',
    y_label='MSE'
)
```

```python
# TODO: 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: 39.8482
Testing set mean squared error: 44.9514
Results for ridge regression model with weight decay of 2
Training set mean squared error: 39.8498
Testing set mean squared error: 44.9577
Results for ridge regression model with weight decay of 4
Training set mean squared error: 39.8558
Testing set mean squared error: 44.9728
Results for ridge regression model with weight decay of 8
Training set mean squared error: 39.8783
Testing set mean squared error: 45.0126
Results for ridge regression model with weight decay of 16
Training set mean squared error: 39.9575
Testing set mean squared error: 45.1229
Results for ridge regression model with weight decay of 32
Training set mean squared error: 40.2074
Testing set mean squared error: 45.4245
Results for ridge regression model with weight decay of 64
Training set mean squared error: 40.8711
Testing set mean squared error: 46.1633
Results for ridge regression model with weight decay of 128
Training set mean squared error: 42.2802
Testing set mean squared error: 47.6655
Results for ridge regression model with weight decay of 256
Training set mean squared error: 44.7632
Testing set mean squared error: 50.2544
Results for ridge regression model with weight decay of 512
Training set mean squared error: 49.2108
Testing set mean squared error: 54.8368
Results for ridge regression model with weight decay of 1024
Training set mean squared error: 58.6890
Testing set mean squared error: 64.5329
Results for ridge regression model with weight decay of 2048
Training set mean squared error: 79.0307
Testing set mean squared error: 85.2557
Results for ridge regression model with weight decay of 4096
Training set mean squared error: 114.0822
Testing set mean squared error: 120.8847
Results for ridge regression model with weight decay of 8192
Training set mean squared error: 157.9772
Testing set mean squared error: 165.4517
Results for ridge regression model with weight decay of 16384
Training set mean squared error: 198.2125
Testing set mean squared error: 206.2790
Results for ridge regression model with weight decay of 32768
Training set mean squared error: 227.1206
Testing set mean squared error: 235.6035
```

```
Out[ ]: 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

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

# Set polynomial expansion
degree = 6

# TODO: 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)]


# TODO: Initialize polynomial expansion
poly_transform = []

# TODO: Compute the polynomial terms needed for the data
poly_transform = skpreprocess.PolynomialFeatures(6)

# TODO: Transform the data by nonlinear mapping
poly_transform.fit(X_train)
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:

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

    # TODO: Train ridge regression model
    model_ridge_poly.fit(X_poly_train, y_train, weight_decay=weight_decay, solver='no
```

```
    print('Results for ridge regression model with weight decay of {} for degree-{} p

    # TODO: 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_tr
    print('Training set mean squared error: {:.4f}'.format(score_mse_ridge_poly_train

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


    # TODO: 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))

    # TODO: 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']

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

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

# TODO: 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 10
# 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'
)

# TODO: Create plot title of r'Ridge Regression with various $\lambda$ for Degree-{} 
fig.suptitle(r'Ridge Regression with various $\lambda$ for Degree-{} Polynomial Expan
```

```
Experiment 3: Ridge Regression with Weight Decay and Polynomial Expansion
Results for ridge regression model with weight decay of 1 for degree-6 polynomial expa
nsion
Training set mean squared error: 28.0858
Testing set mean squared error: 40.4158
Results for ridge regression model with weight decay of 2 for degree-6 polynomial expa
nsion
Training set mean squared error: 29.5419
Testing set mean squared error: 39.8220
Results for ridge regression model with weight decay of 4 for degree-6 polynomial expa
nsion
Training set mean squared error: 30.8583
Testing set mean squared error: 39.6788
Results for ridge regression model with weight decay of 8 for degree-6 polynomial expa
nsion
Training set mean squared error: 32.0425
Testing set mean squared error: 39.7949
Results for ridge regression model with weight decay of 16 for degree-6 polynomial exp
ansion
Training set mean squared error: 33.1490
Testing set mean squared error: 40.0422
Results for ridge regression model with weight decay of 32 for degree-6 polynomial exp
ansion
Training set mean squared error: 34.2403
Testing set mean squared error: 40.4178
Results for ridge regression model with weight decay of 64 for degree-6 polynomial exp
ansion
Training set mean squared error: 35.4209
Testing set mean squared error: 41.0601
Results for ridge regression model with weight decay of 128 for degree-6 polynomial ex
pansion
Training set mean squared error: 36.8756
Testing set mean squared error: 42.1782
Results for ridge regression model with weight decay of 256 for degree-6 polynomial ex
pansion
Training set mean squared error: 38.7897
Testing set mean squared error: 43.9155
Results for ridge regression model with weight decay of 512 for degree-6 polynomial ex
pansion
Training set mean squared error: 41.3499
Testing set mean squared error: 46.3846
Results for ridge regression model with weight decay of 1024 for degree-6 polynomial e
xpansion
Training set mean squared error: 45.0579
Testing set mean squared error: 50.0273
Results for ridge regression model with weight decay of 2048 for degree-6 polynomial e
xpansion
Training set mean squared error: 50.7874
Testing set mean squared error: 55.6868
Results for ridge regression model with weight decay of 4096 for degree-6 polynomial e
xpansion
Training set mean squared error: 59.0832
Testing set mean squared error: 63.9740
Results for ridge regression model with weight decay of 8192 for degree-6 polynomial e
xpansion
Training set mean squared error: 70.3812
Testing set mean squared error: 75.5126
Results for ridge regression model with weight decay of 16384 for degree-6 polynomial
expansion
Training set mean squared error: 87.2439
Testing set mean squared error: 92.9971
Results for ridge regression model with weight decay of 32768 for degree-6 polynomial
expansion
Training set mean squared error: 114.1872
Testing set mean squared error: 120.8370
```

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