Prerequisites

If you're new to NumPy, please read https://numpy.org/doc/stable/user/quickstart.html. You will notice that `for` loops practically never appear!—if you find yourself writing `for` loops, you should look for a vectorized solution. You might also want to watch the video, "Losing your Loops: Fast Numerical Computing with NumPy" by one of the NumPy core developers https://www.youtube.com/watch?v=EEUXKG97YRw&ab_channel=PyCon2015 and see Figure 1 in https://www.nature.com/articles/s41586-020-2649-2 for a discussion of views, broadcasting, and the `axis` parameter.

Practical 1 : Linear Regression

In this practical, we'll implement Linear Regression using the Least Squares method. For the first part, the task is to implement the linear model directly using the `numpy` package. For the more advanced (optional) tasks, you can make use of a package called scikit-learn.

For the purpose of testing, we'll use the winequality dataset. The dataset is available here: https://archive.ics.uci.edu/ml/datasets/Wine+Quality In order to make it easier to import the dataset, we've converted the data to the numpy array format and shuffled it so that you can start the practical directly. There are two different files, for white wines and red wines. They are available on the practicals page of the course website.

We'll focus on the white wine data for the most part, which is a larger dataset. You can load the data from the files as follows:

```
import _pickle as cp
import numpy as np

X, y = cp.load(open('winequality-white.pickle', 'rb'))
```

In order to get consistent results, all students should use the same 80% of the data as training data. We'll use the remaining as test data. To achieve this split run the following:

```
N, D = X.shape
N_train = int(0.8 * N)
N_test = N - N_train

X_train = X[:N_train]
y_train = y[:N_train]
X_test = X[N_train:]
y_test = y[N_train:]
```

We'll not touch the test set except for reporting the errors of our learned models.

Understanding What We're Predicting

Before we get to training a linear model on the data and using it to make predictions, let's look at the spread of y values on the training set. The values are integers between 3 and 9, indicating the quality of the wine.

Handin 1: Make a bar chart showing the distribution of y values appearing in the training set.

The most trivial predictor would simply use the average value of y on the training set as the prediction for every datapoint.

Handin 2: Report the mean squared error, i.e., the average of the squared residuals, using this simplest of predictors on the training and test data. We should hope that our models beat at least this baseline.

Linear Model Using Least Squares

Let us first fit a linear regression model and then calculate the training and test error. We'll actually use the closed form solution of the least squares estimate for the linear model. Although it's not strictly necessary (why?) for the linear model using the least squares method, let's standardise the data, i.e., make every feature have mean 0 and variance 1.

We do the standardisation using the training data, so we need to remember the means and the standard deviations, so that they can be applied to the test data as well. Apply the standardisation so that every feature in the training data has mean 0 and variance 1. Apply the same transformation to the test data. (Note that the features in the test data will have mean approximately 0 and variance approximately 1, though not exactly.)

Handin 3: Report the mean squared error using the linear model on the training and test data.

Learning Curves

Let us see if the linear model is overfitting or underfitting. Since the dataset is somewhat large and there are only 11 features, our guess should be that it may either be underfitting or be about right.

Starting with 20 datapoints, we'll use training datasets of increasing size, in increments of 20, up to about 600 datapoints. For each case train the linear model only using the first n elements of the training data. Calculate the training error (on the data used) and the test error (on the full test set). Plot the training error and test error as a function of the size of the dataset used for training.

Handin 4: Report the learning curves plot. Also, explain whether you think the model is underfitting or not, and how much data you need before getting the optimal test error.

(Optional) Polynomial Basis Expansion with Ridge and Lasso

For this part use the following from the scikit-learn package. Read the documentation available here: http://scikit-learn.org/stable/modules/classes.html

You'll need to make use of the following:

- `linear_model.Ridge`
- `linear_model.Lasso`
- `preprocessing.PolynomialFeatures`
- `preprocessing.StandardScaler`
- `pipeline.make_pipeline`

Try 5 powers of 10 for λ from 10^{-2} to 10^{2} and use degree 2 basis expansion. Fit Ridge and Lasso using degree 2 polynomial expansion with these values of λ. You should pick the optimal values for λ using a validation set. Set the last 20% of the training set for the purpose of validation. However, do not use the test set at this time.

Once you've obtained the optimal values for λ for Ridge and Lasso (they will typically be different), train these models using these hyperparameters on the full training set. Then report the training and test error.

(Super-optional) Larger Degrees

Try using higher degree basis expansion. You may want to use k-fold cross validation to determine the values of hyperparameters rather than just keeping a validation set. There are tools in `scikit-learn` to let you do this automatically.