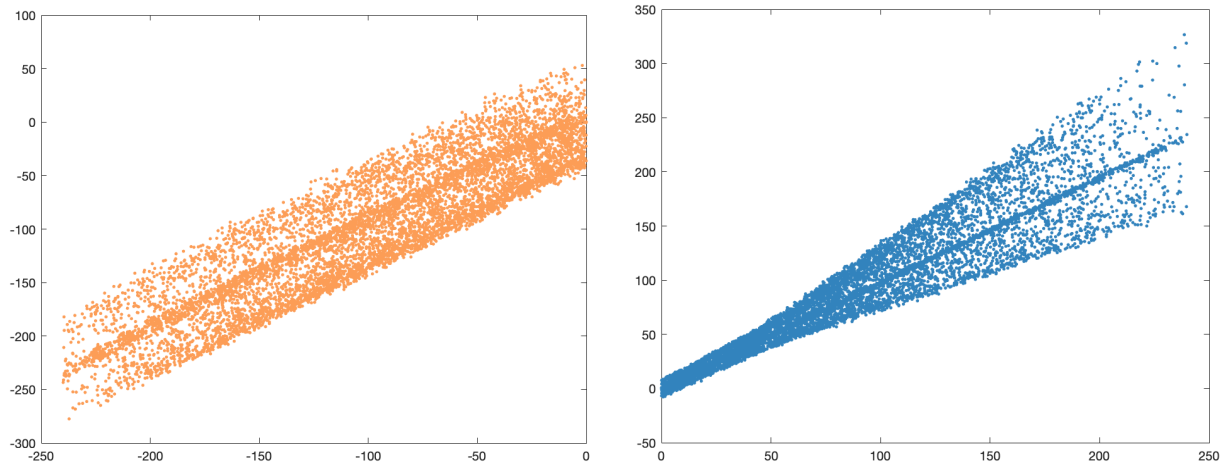
Topics In Scientific Computing Week 3

Regression Task

This task consists of finding a curve fit for a specific dataset. It is limited to two dimensional data. The idea of this code is to find correlations in datasets and then estimate the relation between the two subsets. In this case to find the correlations within the Electric Motor dataset, I chose to plot all inputs against all outputs and visually check if there are any correlations. Another way to do this is by calculating the covariance matrix for the data set and finding the highest values which correspond to the highest correlations between subsets. Both of these methods showed that the first and second inputs were correlated respectively to the first and second outputs. Indeed these correlations make sense, because according to the specifications of the data, input one is the measured d-current at k and its correlated output is also the d-current at k+1. It makes sense that the currents passing through two subsequent points are correlated.



Plots of $i_{d,k}$ vs $i_{d,k+1}$ and $i_{q,k}$ vs $i_{q,k+1}$. There is indeed a correlation visible.

Using the get params function I wrote, which uses the method from the lecture to get the parameters of the best fitting line through minimizing the mean squared error between the fitting curve and the dataset. In the getparams function I can specify what order curve to try and fit. Doing both linear and quadratic regressions leads to similar coefficients, where the quadratic one has x^2 term very close to zero as the correlation is almost perfectly linear for the first case at least.

Linear regression:

- $i_{d,k+1} = 0.9427\, i_{d,k} - 5.3076$ R = 81.3834
- $i_{q,k+1} = 0.9801\, i_{q,k} + 3.0356$ R = 89.3556

Quadratic regression:

- $i_{d,k+1} = 0.0000\, i_{d,k}^2 + 0.9486\, i_{d,k} - 5.0909$ R = 81.3836

- $i_{q,k+1} = -0.0009\, i_{q,k}^2 + 1.1660\, i_{q,k} - 3.3601$ R = 89.6275

Judging by the correlation coefficients R we can tell there is indeed a correlation between the two sets of input and outputs. However, to test the strength of the parameters, we can divide the data set into two parts: a training set and a testing set. The point is to get parameters of the training set and then look at the mean squared error (MSE) between the curve fit with those parameters and the new testing set. If the MSE of the testing set is comparable to the MSE of the training set then the parameters obtained can be said to have some predictive power.

Linear regression

```
mse_train1 = 391.7399
mse_test1 = 387.1322
mse_train2 = 202.0431
mse_test2 = 189.4897
```

Quadratic regression

```
mse_train1 = 391.7346
mse_test1 = 387.1611
mse_train2 = 196.8833
mse_test2 = 188.7034
```

Judging from the data above, the two MSE are quite similar between the linear and quadratic models, yet the quadratic models appear to have a smaller SME which implies that the quadratic curve fit is a better model.

Principal component analysis

This function is meant to reduce the dimensional complexity of high dimensional data. The example given of mice protein expressions is displayed in 68 dimensions. To do this, we find along which dimensions most of the data is expressed and transform the data to be represented only on this basis. The way to do this is using the eigenvalues and eigenvectors of the covariance matrix of the data. By normalizing the eigenvalues of the covariance matrix, they represent how much that dimension contributes to the variance of the data. Then we can redefine the data based on the eigenvalues, and corresponding eigenvectors to reduce the dimensional complexity. To do this, we need to compute the eigenvalues and functions then find which eigenvalues form 95% of the variance and then create a transformation matrix of the corresponding eigenvectors which we can use to transform the original previously centered data. To find which eigenvectors make up 95% of the variance, I ordered their normalized matrix in descending order and looked at the cumulative sum of the entries and found at what index

0.95 reached. The hardest part of this task was rewriting the code for different formats of data. For example the test data given in DIY 7 were row vectors and the mice protein data were column vectors. In the end, using the example data of the mice gene expression, I managed to reduce the data from 68 dimensions to 9 dimensions. However I cannot tell for sure if the reduced data is correct as there is no way to visualize it.