Aston University
Machine Learning

Dimensionality Reduction (1)

Learning Outcomes:
In the current unit, we studied the Principal Component Analysis (PCA) algorithm for reducing the dimensionality of a dataset. In this lab, we will deepen our understanding of the algorithm by implementing it and applying it to a test data set.

Instructions:
Attached to the lab task on Blackboard is a data.csv. This dataset is generated according to the same scheme as the one shown in lectures.

Task 1:
Familiarise yourself with the scikit-learn implementation of PCA which is documented here: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. Start by fitting a PCA model to the data using the same number of components as you have explanatory variables.

Once you have done this, inspect the properties **`explained_variance_ratio_`**. What does this tell you about the number of components you could use to explain the majority of the variance in the dataset?

Now look at **`components_`**. What do the components corresponding to high values of the explained variance ratios look like? How does this accord with your understanding of the data?

Finally, use PCA to project the data down into 2D space (the easiest way to do this is to fit a new PCA model with 2 components and using the model to **`transform`** the data). Plot a scatter graph of the data. What you see should be close to the graph given in the lecture slides.

Task 2:
To further build up your understanding of PCA, you should now implement it. The PCA algorithm is given in the lecture slides and is fairly straightforward, but you will need to be careful about the alignment of your data. For this you can use the **`numpy`** library (you will need to make sure that your data is in a numpy array. If you have read the data in using the pandas function **`read_csv`**, then the **`values`** property of the result will be a numpy array). Functions of particular interest will be:

- **`np.ndarray.mean(X,axis=0)`**: This will give you the mean row value of matrix **`X`**.
- **`np.subtract(X,v)`**: If **`v`** is a vector of the same size as a row of **`X`**, this will subtract **`v`** from each row of **`X`**.
- **`np.transpose(X)`**: This calculates the transpose of a matrix **`X`** (i.e. $\boldsymbol{X}^T$ in our lecture notes).
- **`np.matmul(X,Y)`**: This multiplies matrix **`X`** by matrix **`Y`** (i.e. $\boldsymbol{XY}$ in our lecture notes).

- **`np.linalg.eig(X)`**: This returns two values: the first being the eigenvalues of **`X`** and the second being the corresponding eigenvectors. Note that this will return columns of eigenvectors, not rows of them.

Take care at each step that your result has the dimensionality that you were expecting. For instance:

- the covariance matrix should be of size D by D, where D is the number of explanatory variables,
- the matrix of eigenvectors should also be of this size. You should have one eigenvalue for each eigenvector.

Apply your function to the lab data. Your eigenvectors should be the same as the PCA components calculated during task 1. If you normalise the eigenvalues (i.e. divide the vector by the sum of its elements to give you a new vector with elements which sum to one), they should match the explained variance ratio.

Finally, use the two eigenvectors with the highest eigenvalues to project the data into 2D space and plot a scatter graph. It should match the graph you created during step 1.