7COM1033 Semester B 2020/21
Jan T Kim, `j.t.kim@herts.ac.uk` 15 March 2021

Neural Networks & Machine Learning: Worksheet 3

Aim

This worksheet continues exploration of unsupervised learning. By completing the tasks on this sheet, you will

- firm up and extend your ability to use the Matlab documentation for systematically developing solutions,
- practically explore key constituent parts of Principal Component Analysis,
- gain experience in using the PCA functionality provided by Netlab,
- generate test data for Machine Learning algorithms using pseudo random numbers.

Developing Machine Learning solutions requires (1) identifying and formally stating an approach, and (2) on that basis finding the programming techniques and functions required to implement that approach. The focus of practical labs is on the second aspect. Therefore, take time to study elements of the Matlab system, and to learn how to recognise their use for the task at hand. The main reference for Matlab is the online documentation. You can access that via the "Documentation" window, or by typing `help`, followed by the name of the command or function in question.

The tasks on this sheet include specific guidance on Matlab programming and functions that are recommended to solve them. Please read the documentation to the extent you require to exactly understand how to use them, and consult your lab leaders if you have any questions or encounter problems.

Please remember to develop your work in files (scripts and functions), and make sure you save these where you can access them in the future.

This worksheet is not marked and no submission is required.

Tasks

For this worksheet you need the files `sparrows.dat` and `sparrows_labels.dat` which are available on Canvas. You find them in the same directory from which this sheet is available.

Place these files in a suitable directory and use Matlab's `addpath` function to add that directory to the search path. Check that Matlab finds these files by running the commands

```
load sparrows.dat;
load sparrows_labels.dat;
size(sparrows)
size(sparrows_labels)
```

The last two commands should show that both datasets have 49 rows, where `sparrows` has 5 columns and `sparrows_labels` has 1 column.

1. **Manual PCA using Covariances, Eigenvectors and Eigenvalues.** The objective of this task is to reproduce the PCA shown in the slides entitled "the first example" from the lecture. Create the small dataset as follows:

   ```
   data = [-2, 2; -4, -4; 4, 4; 2, -2; 0, 0];
   ```

   Verify that the scatterplot of your data looks like that shown in Fig. 1.

   Review the documentation on the `cov` function and use that to compute the covariance matrix of the data. After inspecting the covariance matrix, use the `eig` function to compute its eigenvectors and eigenvalues.

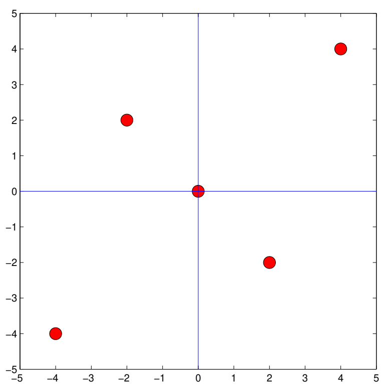

Figure 1: Plot of the data to be used for exploring PCA.

The `eig` function returns eigenvectors and eigenvalues in an unspecified order. However, recall that for PCA, the eigenvalues are required in descending order. Therefore, use the `sort` function to order the eigenvalues. Use the index supplied by `sort` to order the eigenvectors accordingly.

Carry out a matrix multiplication of the data matrix and the matrix of sorted eigenvectors. Use a scatterplot to verify the rotation of the original data.

Verify your work further by checking that the trace of the covariance of the rotated data is the same as that of the covariance of the original data. Also check that the rotated data is decorrelated, i.e. that the elements outside of the diagonals are 0.

2. **PCA of the Sparrows Dataset.** Apply the method for manually conducting PCA manually on the `sparrows` dataset. Notice that the measurements pertain to different biometric aspects of the birds, and that therefore, the data should be normalised to zero means and unit standard deviation.

   Subsequently, review the documentation of Netlab's `pca` function and use that to to carry out PCA on the `sparrows` data. Do you get similar results from both methods? If you notice any major discrepancies, investigate these. The results you obtain should also closely resemble those in the lecture slides. You may use the `sparrows_labels` data to reproduce the survivor / non-survivor labelling.

3. **Creating 2D Datasets of Random Values.** Use the `randn` function to create a 50×2 matrices of values from a normal Gaussian distribution. Plot these values using the `scatter` function. Then, add offsets $x_0 = 15$ and $y_0 = 5$ to these points, and plot them again. Notice how the plot area does not change much, the change of values is mostly reflected in the axes. Consult the documentation of the `axis` function and use it to show the range $[-5, 25]$ on both axes, regardless of the values plotted.

   Develop your code into a function that takes the number of rows, and the offsets along the x- and y-axis as parameters and returns a `numRows` $\times 2$ matrix of values. Start this code by writing

```
function m = gaussianPoints2D(numRows, x0, y0)
    % complete the line below:
    m =
end
```

   Tip: Take a look at the code you wrote to normalise data (Worksheet 1, task "Data Normalisation"). You can use the same technique that you employed for subtracting means to add offsets x_0 and y_0.

   After completing the `gaussianPoints2D` function, write a similar function `uniformPoints2D` to construct matrices of uniformly distributed values (using the `rand` function).

4. **Random Points in Two Clusters.** Review the Matlab documentation on concatenating matrices (you can find this section e.g. by typing "concatenating matrices" into the search box of the Documentation window). Create one matrix of 500 points from the uniform distribution and one matrix of 300 points from a Gaussian distribution centred on $(15, 5)$, and concatenate these into a 800×2 matrix. Show a scatterplot of this matrix, making sure that both axes show the same range.

5. **K-Means Analysis of the Two Cluster Dataset.** Review the code you wrote for the K-means task last week, and use that to carry out K-means analysis of the two-cluster dataset you created. Write code to randomly create initial codevectors. Run K-Means on your dataset a few times, starting with different codevectors. Show a scatterplot with the final codevectors and the data, using the `datacodeplot` function you wrote last week. Do you get different results when using $K = 2$ codevectors? What do you observe with $K = 3$ codevectors?

   You may explore these questions by running commands interactively in the Command Window. Then develop these commands into a script that uses a loop to show 10 results from different initial codevectors for $K = 2$ and for $K = 3$. Include a command to seed the pseudo-random number generator at the start of your script, so that running it gives you the same series of results each time. Refer to the documentation of the random number generator you use (e.g. `rand` or `randi`) to find out how to seed the generator.

6. **Further K-Means Experiments.** You now have developed facilities to generate clusters of data points (from a normal or a uniform distribution), and to combine them into datasets with two clusters. By concatenating further clusters, you can extend this to generate multi-cluster datasets. You also have the a facility to carry out batches (e.g. of 10) K-means analyses, each starting from different initial codevectors.

   Use these facilities to explore how K-Means performs in various situations. Ideas for questions to explore include:

   - What results do you get if the number of codevectors is smaller than the number of clusters?
   - What if the number of codevectors is larger than the number of clusters?
   - What happens if clusters are close together? In which conditions you are likely to find one codevector for each cluster? What makes it more likely to get only one codevector for both clusters?

   Try to anticipate the results in these scenarios before running the actual analyses, and explore scenarios that you design yourself.