MLD 2019 - Problem set 3

Introduction

This problem set looks at the use of Principal Component Analysis (PCA) and classification. The first problem goes through the process of calculating a PCA "by hand" so that you understand the steps that you need to take, but it also walks you through using the `sklearn` PCA routines.

If you are familiar with PCA, the problem 1 & 2 are probably not worth your time as they merely repeat much of what was done in the lecture. Focus on problem 3 which is realistic application to astronomical data.

Problem 4 uses some of the tools you have learned in this course.

Problem 5 is based around the classification example in the lecture and if you felt that was clear there is no need to do this, but if you want to understand it better going through the Jupyter notebook.

Finally problem 6 is a more complex application of PCA. This is similar to what I showed in the lectures about PCA of stellar spectra - but here applied to a set of models of white dwarf spectra. Again this is based on a Jupyter notebook.

1. PCA of x & y data

The file `Datafiles/x-vs-y-for-PCA.csv` in the git repository contains an index and then x & y for a very simple dataset to do PCA on (similar to what I used in the lectures).

a) Standardise your data manually (not using the module listed below in g).

b) Calculate the covariance matrix of the standardised data (see the math reminder document if you are unsure how to do this). I will call this covariance matrix C_x in the following. Compare the covariance matrix with a plot of the data and see whether it makes sense to you.

c) Calculate the eigenvalues and eigenvectors of C_x. How many eigenvalues are significant do you feel? The first eigenvector for me was (0.7071, 0.7071), check that you get something similar.

   Hint: a convenient way to extract vectors from a matrix is to use: `ev1, ev2 = zip(*M)`

d) Plot the eigenvectors over the original (not the standardised) data.

e) Calculate the eigen-components of the data. To do this, you need to combine the eigenvectors you want to use into a matrix (in this case all), and then matrix multiply this with the coordinates of each point to get the projected points.

   For simplicity you can do this in a loop, in which case my loop looks like (v is my matrix of eigenvectors and xw and yw are the standardized data).

```
for i in range(len(xw)):
       pcs[:, i] = np.matmul(v.T, np.array([xw[i], yw[i]]).T)
```

   Plot the resulting PC1 and PC2 axes - does this look like what you expected?

f) Project the data onto only one axis. In this case you use only one eigenvector in your matrix, but otherwise it works the same as e). Let us call these transformed data PC1. Now project PC1 back to the original space - to do that, you multiply PC1 with the matrix you used for the projection. So if your eigenvector is `ev1`, you have `x = PC1*ev1[0]` and `y=PC1*ev1[1]` (plus you need to undo the standardizing but that should be straightforward.

g) Finally, we can do the same using the PCA module in `sklearn`. To do that we have to import two packages:

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

We make a PCA object - note that we need to specify the number of components at the outset:

```
pca = PCA(whiten=False, n_components=2)
```

we create a X variable and standardize it:

```
X = np.vstack([x, y]).T
scl = StandardScaler()
Xs = scl.fit_transform(X)
```

and run the PCA on these data:

```
pca.fit(Xs)
```

and the eigenvectors are now stored in the `pca.components_` element:

```
ev1_1, ev2_1 = zip(*pca.components_.T)
ev1_1, np.array(ev1)*std_x
```

Now you can do the same as problem f) using this package - you will most likely want to use the inverse_transform function in `pca` and also in `scl` to get back to the original space.

2. Exploring higher dimensional PCA.

The file `Datafiles/xyzw-for-PCA.csv` contains index, x, y, z & w variables for a simple synthetic dataset. The data was created using the code in code-for-xyzw.py and the unrotated data in `Datafiles/xyzw-for-PCA-no-rotation.csv` shows what it looks like if the coordinate axes are well chosen.

a) Run PCA on these data after standardizing them. What is the effective dimension of the data. Plot the transformed data in 4D and 2D.

b) Run PCA on these data *without* standardizing them. What does the results look like?

c) Compare the results of a) & b) - can you explain the differences?

3. PCA on the fundamental plane

The fundamental plane of elliptical galaxies is a relationship between the velocity dispersion, log σ, the effective radius, log Re, and the surface brightness, log μe. Here we will do a PCA analysis of a set of these data to see how well such an algorithm can find this plane.

a) Start by getting the “SMAC. III. Fundamental Plane catalog” from Hudson et al (2001) from Vizier - you want the standardised data catalogue with 699 rows.

b) The relevant columns are logRe (log R_e), <mu>e (log μ_e) and logsigma (log σ). There are many parametrisations of the fundamental plane. For concreteness we choose the one from Jørgensen, Franx & Kjærgard (1996, MNRAS, 280, 167): $\log R_e = 1.24 \log \sigma - 0.82 \log I_e +$ constant. Does your data fit this relation? What is scatter in $\log R_e$ around this plane? [recall that $\mu_e = -2.5 \log I_e$]

c) Carry out a PCA of these data. Which eigenvector should you choose to define the plane (recall that a plane is defined up to a constant by its normal vector).? Does your PCA solution match the one from Jørgensen et al? Calculate the spread in re around this plane - is it smaller or larger than the one found using the Jørgensen parametrisation?

4. How many peaks?

The file `Datafiles/mysterious-peaks.csv` contains data drawn from a distribution with multiple peaks. How many peaks are there?

5. Galaxy classification using Bayesian approaches.

This problem is implemented as a IPython notebook. It goes through the classification problem discussed in the lecture and asks you to classify galaxies & calculate mis-classification rates. The problem is found in the `Classification of galaxies.ipynb` file in the `Problem set 3` directory in the git repository.

6. PCA of white dwarf models

This problem asks you to do PCA of a grid of models of white dwarf model spectra. The whole problem set is put in a jupyter notebook named `White dwarf PCA for Git.ipynb`. You do of course not need to solve the problem in a notebook if you prefer not to - but the relevant instructions and code are there.