Applying ML Tutorial
COMS3007

Benjamin Rosman, Devon Jarvis

Use your notes and any resources you find online to answer the following questions. This does not need to be submitted.

1. In this tutorial, we will look at playing with something that more closely resembles real-world data. Two classification datasets have been provided for you: `ThoracicSurgery.txt` which describes attributes for a number of surgery patients and attempts to predict how many survived afterwards, and `Data_Cortex_Nuclear.csv` which describes a number of attributes relating to gene expression and behaviours in mice and aims to predict the class of the mice (from 8 classes). Description files are provided for both datasets. Work through the following steps for BOTH datasets. These are very different, and so the same steps can be followed, giving very different results.

   This tut follows a simple approach to exploratory data analysis. There are many questions you could ask about your data, and this will guide you through some of the common ones. You really want the answers to these questions to quide your investigation further. Be curious, and try understand what is going on!

   (a) Load the data. This in itself may require you to manually open the file and reformat it into something that is easy to load.

   (b) Look at the dataset as a whole. Pay attention to the numbers of features and numbers of records. From this first glance, do you think there's enough data?

   (c) Look at the features. What do they each mean? Are they continuous or discrete valued? What are the ranges of the values? What kind of preprocessing or normalisation may be required here? Are there the expect numbers of different values for each feature? If there are more, this may be from typos or NANs. Are there any missing values?

   (d) Plot histograms of the features. Are there any peculiar outliers? What could these mean? What kinds of distributions do you see in these values?

   (e) From your initial investigation, perform any appropriate normalisation.

   (f) Plot a scatterplot matrix. You may just want to do this on a subset of the data and the features. Do you notice anything interesting? When you colour-code the points by class, does this give you any indication of features that may be particularly useful?

   (g) Perform PCA on the data. Remember, this involves computing the $m \times m$ covariance matrix (where m is the number of features), and then computing the eigenvalues and eigenvectors of the matrix. Produce a plot of the variation in the data that each vector is responsible for, by plotting the normalised eigenvalues. How many eigenvectors would you keep according to this?

   (h) Project your data down to 2D and 3D and produce these plots. What percentage variation have you kept in each case? Does this give you any idea of how separable the data is?

   (i) Perform clustering on your original data, into a number of clusters equal to the number of classes you are expecting to see. How well do these clusters align wit the true classes?

(j) Based on what you've seen here, what classification method would you use? Why? Use factors such as the types of variables, the correlation between them, etc., to answer this. If you use logistic regression, what model would you use? If you use a neural network, what architecture would you use?

(k) Divide your data randomly into training, validation, and testing data. Train your chosen model on the training data and optimise the hyperparameters on the validation data.

(l) Using the testing data, produce the confusion matrix. From this, compute the classification error, the accuracy, the precision and the recall. Do these values all make sense to you?

(m) Compare the performance with other people in the class. What approaches worked better? In retrospect, can you work out why?

2. For this question use the Olivetti Faces dataset from the file "olivetti_py3.pkz" or from sklearn data sets using "sklearn.datasets.fetch_olivetti_faces()". Perform PCA on this data. Save the first 5 principal components as images. Submit your code and the 5 images on moodle.

3. A jupyter notebook as been included in the handout to let you visualise your results (this part is not to be submitted). Use your PCA code from above to complete this notebook. You should be able to tranfer your code directly. Make sure of the following to ensure the notebook works:

   (a) Store the data in a variable named "images" where each row is a flattened image (so "images" will be a matrix with shape (400, 4096)).

   (b) Store the eigen-values from PCA in a variable named "eig_value" and store the eigen-values in ascending order.

   (c) Store the principal components (eigen-vectors from PCA) in a variable named "eig_vectors" where each column is an eigen-vector and the eigen-vectors are in ascending order (matching the eigen-values.)

   If the notebook works you will see two following parts to the notebook. You can play around with the rest of the notebook to understand PCA and image compression with PCA.

   Firstly you will see a section with 3 sliders. The first slider allows you to select which of the 400 images to use. The second slider allows you to select which of the 4096 eigen-vectors to use, and finally the third slider lets you select how much of the eigen-vector is added to the orginal image. This will create a new image using the eigen-vector to transform the original image and increase the variance along the principal component.

   The second part of the notebook has two sliders. The first again selects which of the 400 images to use. The second slider picks how many of the 4096 principal components to keep and prints the percentage of variation being kept as a result. This produces a new compressed image.