Letter Recognition using KNN

The first thing in this task was is to import the .csv file provided. It is then seen that it contains letters in one column and 16 columns for each feature of the letter. It contains 20000 letters and their features. Since the machine learning algorithms in python take numbers instead of characters (letters in this case), they are converted into the integers using the LabelEncoder() function. It encodes the letters such as A=1, B=2 and so on. This way any machine learning algorithm can take input and give the output. If there is need to see the outputs in a meaningful way, then they can be converted back to letters. Then comes the part for dimensionality reduction. For this, Principal Component Analysis (PCA) function is used to reduce the number of features from 16 to 8 features that means reduced by half. The PCA is an excellent algorithm to use because it projects the information on n spaces where n is the number of reduced features.

Firstly, the PCA needs the standardized data. The standardization rescales the data to range of between 0 and 1. This process is important because data with bigger difference will contribute more to the dominating factor for the final selection of the feature. This can be done mathematically using the formula:

$$z = \frac{value - mean}{standard\ deviation}$$

Where z is the standardized data value. This process is repeated for every feature of all the letters. Then the covariance matrix is plotted of the data to identify the correlations. Since this dataset has 16 features therefore 16x16 matrix is plotted then their eigen vectors and eigen values are calculated. The principal components means that the components with the highest variance are selected until specified limit is reached (8 features in this case), the eigen values and their vectors with highest variance are selected. These are the principal components. The higher the variance, more the information that component contains. The aim is to select components with the highest information in descending order. Then the feature vector is plotted which is simply the matrix of the eigen vectors that are chosen in the previous step.

The next step is to join the array of letters converted into numbers with the array containing the reduced features. Then the data is split into the training and test sets with 25% of the data being the test set. Then machine learning model is then used to classify the letters. The K-Nearest Neighbours is used for this purpose. It is a supervised algorithm which relies on using the labelled data for learning. It means it uses data and the targets both for the training data and then it can predict the test data. Using default parameters, the KNN algorithm uses Euclidean distance to calculate the distance between point under consideration with all the other points.

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Where d(p,q) is the distance between two points. This distance is with all the other points are then ordered in an ascending order. This way first distance shows the nearest point. Then the first k (where k=5 in this case) points are selected (features of the letters) and their labels (the letters) are returned. From these labels the mode of labels is selected as the class. The weight to each prediction is given equal (uniform) in the algorithm although it can be changed. Then overall accuracy is 87 %. The cross validation is used with 3 folds to evaluate the result and the results were 76.4%, 78% and 75.8%. The advantage of using cross validation is that is shuffles the data and splits it into k folds (3 folds in this case). At last, the confusion matrix is plotted using the Sklearn library built-in function. The confusion matrix below in figure file "confusion_matrix_KNN" shows the clearer picture of each letter recognition. It is seen that least number of correctly recognized letter is H whereas the greatest number of correctly recognized letter is A.

In conclusion, this is a basic implementation of letter recognition. There are some reasons that it does not perform very accurately or like those algorithms used in the industry. Firstly, it uses a small dataset whereas industrial machines use comparatively huge dataset and consume up-to couple of days of learning. Secondly, the industrial solutions now mostly consist of neural networks either solely or as a part of a bigger algorithm nowadays.