Perceptron, Ensemble Methods, Classifier Evaluation, Clustering,

Text Classification: Data Mining exercise 2

Lichuan Xiang – 1650783

Introduction

This work is aimed to using R and Python to implement clustering or classification method to handle several different practical problems, and using proper evaluation method to find optimal classifiers or best clustering methods. The main body of this essay is form of three major parts, each parts focus on one perspective of data mining and includes several problems. The solution and evaluation of each problems will be presented in each part. We will start will an implementation of a perceptron classifier, followed by a section that discovering on clustering. And solutions for multi-label and multi-output test classification will be illustrated before the conclusion. Several packages and libraries has been adapted in certain works, the detail of those requirement will be illustrated on README file in each code folder.

1、 Perceptron

Perceptron can be treated as a shallow neural networks with an input layer, an equal bias hidden layer and one neuron output layer. It takes a real values vector as input, and by calculating it's linear combination, it give a result that can be evaluate by output layer. If this value higher than threshold value, the output will be 1 else output will be -1. More precisely definition in mathematics can be illustrate as follow:

$$o(x_1,x_2,....,x_n)=\begin{cases}1 & if \quad w_0+w_1x_1+w_2x_2+...+w_nx_n>0\\ -1 & Otherwise\end{cases}$$

Where the hypothesis space of this perceptron is the set of all weight: $H=\{\vec{w}\,|\,\vec{w}\in\xi^{n+1}\}$.

To training this perceptron, we start initial all weights as zero, a better way is that initial them with random small values. The training process followed perceptron training rule, which modified weights vector based on input on every step. For every step, the weight will be updated as:

$$w_i \leftarrow w_i+\Delta w_i$$

Where:

$$\Delta w_i=\eta(t-o)x_i$$

The η is represent as a learning rate that controls the step of weight update, and t is training target and o is output of perceptron.

As Minsky and Papert (1969 quoted by Mitchell 2003) has introduced, if the training samples are linear dividable and the learning rate is small enough, the model always converged after finite iterations. To explore that, this works include a grid search function, that searching converged states from two dimension space of parameters bias and learning rate, the range of bias is [-1, 1], while learning rate stay on [0.01, 20]. And we searched 22000 times with different

parameters in this space, 0 times not converged.

2、 Classification Evaluation and Clustering

2.1 95% Interval for the Expected Error

As we have already observe sample error at 6.67% from 145 instance, we can estimate true error by calculating equation as follow

$$error \in error_S(h) \pm z_N \sqrt{\frac{error_S(h)(1-error_S(h))}{n}}$$

As confidence interval is 95%, so Z_n = 1.96, so we get:

$$error \in 0.0667 \pm 1.96 \times \sqrt{\frac{0.0667 \times (1\text{-}0.0667)}{145}} = 0.0667 \pm (1.96 \times 0.0207)$$

The 95% interval for the expected error is [0.0261, 0.1073].

2.2 Classification Comparison

Compared the performance of two classifiers, can following the procedures showing below:

1、 Divided data in to k folder:

2、 For i from 1 to k, set folder i as test folder:

Training and predict at test samples.

Get $error_i^A$ and $error_i^B$

Calculate it sample difference: $\delta_i \leftarrow error_i^A - error_i^B$

3、 Calculate mean sample difference: $\bar{\delta} \equiv \frac{1}{k} \sum_{i=1}^{k} \delta_i$

4、 Difference of true error can be estimated as: $\bar{\delta} \pm t_{N,\ k-1} s_{\bar{\delta}}$, where $s_{\bar{\delta}} \equiv \sqrt{\frac{1}{k(k-1)} \sum_{i=1}^{k} (\delta_i - \bar{\delta})^2}$

As the information provided in question, k = 10, and we can calculated difference on each folder:

CV Fold	Algorithm 1	Algorithm 2	difference
1	91.11	90.7	0.41
2	90.48	90.52	-0.04
3	91.87	90.88	0.99
4	90.52	90.87	-0.35
5	89.88	90.02	-0.14
6	89.77	88.99	0.78
7	91.44	90.98	0.46
8	90.88	91.44	-0.56
9	90.77	90.77	0
10	90.89	90.92	-0.03
mean			0.152

So, we can get $s_{\bar{\delta}} = 0.1559$, the true error difference can be estimated in range

$0.152 \pm (t_{N,k-1} \times 0.1559)$. And we search from t table and we can get $0.152 \pm (0.883 \times 0.1559)$, at 60% confidence level that we can accept algorithm 1 will outperform algorithm 2.

2.3 Clustering

In this section, we have implemented three clustering methods that had been introduced on lectures, which are DBSCAN, K-means and agglomerative hierarchical clustering method. The code is write under R language, and several packages and libraries have been used to provide clustering function and data visualization function. To evaluate clustering quality, the clustering plot has been used to be compared with ground truth and average silhouette value also been used to judge clustering quality.

Before implement these clustering method, normalization, a necessary pre-processing has been adopted to eliminate scale difference between elements from difference dimensions.

DBSCAN, also known as Density-based spatial clustering of application with noise, is a density-based clustering algorithm. This algorithm can be illustrate as following steps: 1、Find the certain neighbours below eps distance for every points, and identify the core point with more than minPoints neighbours. 2、Find the connected components of core points on neighbour graph. 3、Assign all non-core points in to their nearest eps distance neighbour, otherwise treat as noise.

One of the key points of DBSCAN is to find optimal eps value, in this work, we used knndisplot function to explore k-nearest neighbour distance distribution:

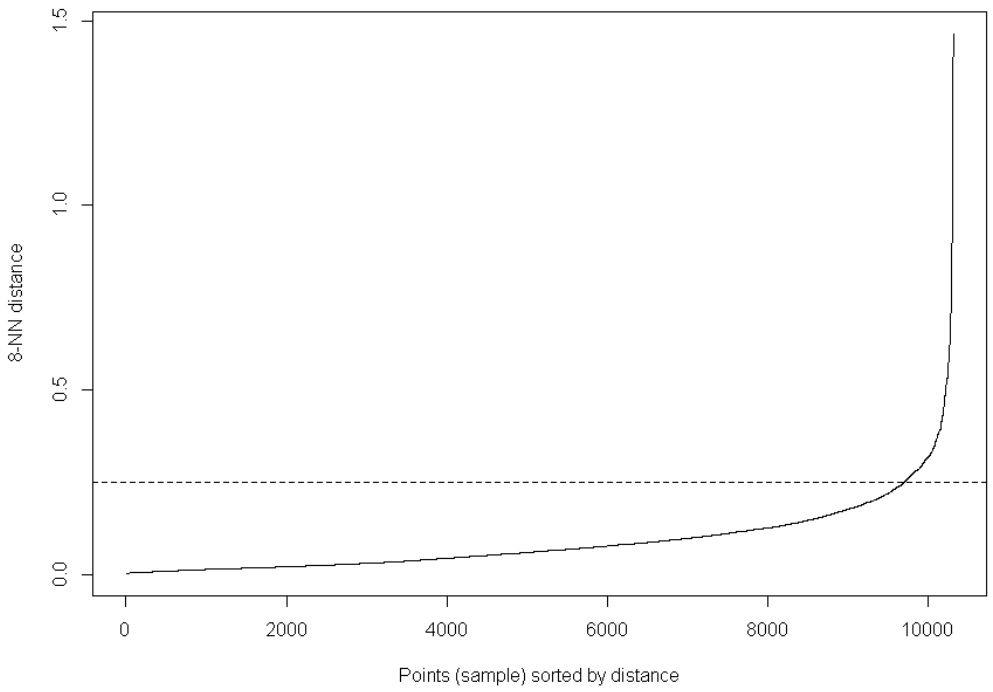

Figure 1-Knn distance count plot

We looking for the “angle” of this plot graph, and it is the optimal eps value which represent majority inner distance of points in clusters.

After determined eps value, we have used minPoints = 2 and minPoints = 5 to discover the clustering ability of DBSCAN methods on this data set, and results are present as follow:

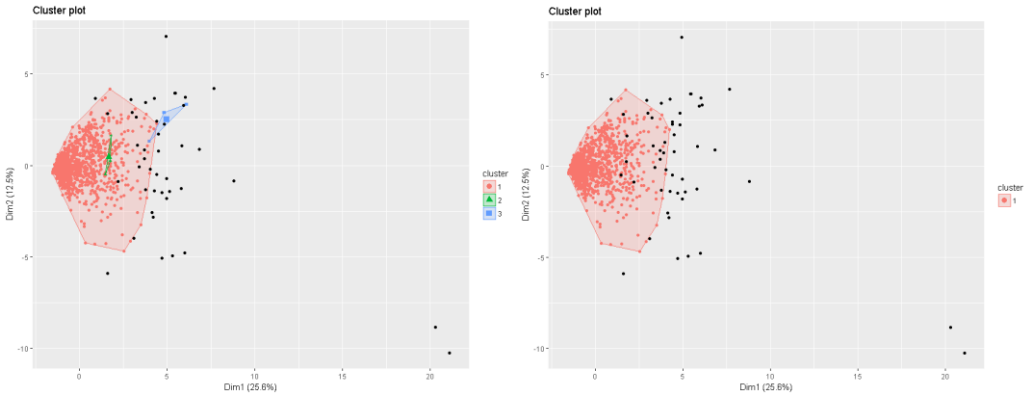

Figure 2: DBSCAN with minPoints =2 (left), DBSCAN with minPoints = 3 (right)

It seems that DBSCAN trend to cluster all points in to one cluster, which far away from ground truth 8 clusters. The major reason for this result is that DBSCAN method using Euclidean distance to calculate density which cause problem when tackle on high dimensions data due to some cluster may divide in certain dimension but closet in rest major dimension would trend to have high density, and DBSCAN will clustering them in to same cluster. One way to tackle this problem is using principle components analysis method to reduce data dimension if the data are linear dividable in certain dimensions.

Under this concept, we perform principle components analysis methods on original data, and plot it variances cross pc dimension. It shows the 3 dimension principle components obtain more than 98% variance. So we transformed 11 dimensions data into 3 dimensions principle component space. And we used transformed data to implement DBSCAN and obtained result as follow:

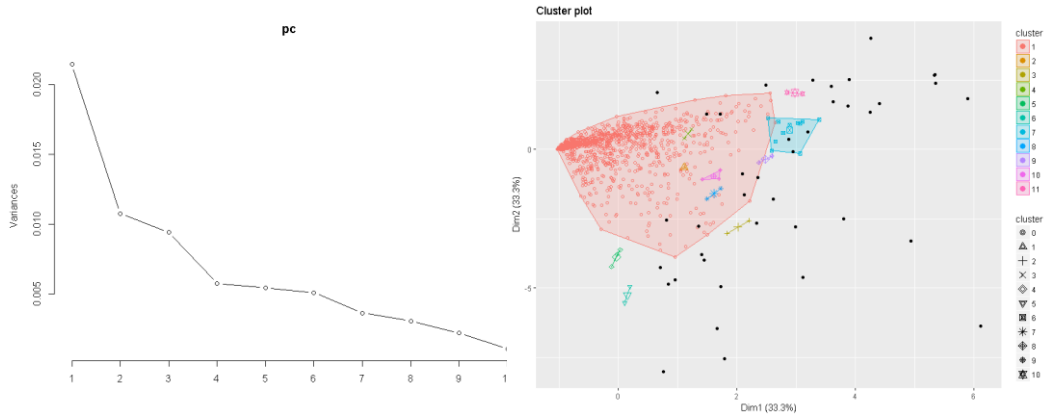

Figure 3: Variance plot across PC dimensions (left), DBSCAN with principle component (right)

The result seems better than previous one, however still far away from the ground truth. And the reason for this situation is that those cluster has many overlapping area cross multiple dimensions, makes data not linear dividable.

K-means algorithm is another common used clustering method. K-means method start with randomly assign k most separate points as initial centroids, and then in each round, assign rest point in to nearest centroid. After all points has been assigned, update new cluster centroid as

cluster mean points. K-means will keep this loop until centroid not change or point' assignment not change.

As it has been known about data that it includes 8 class, so we started with using k=8 to explore the K-means method. In this work, a self-designed k-means function has been provide in scripts, however, due to the performance issue, we will using kmeans function from pre-installed R package. The results is showing below:

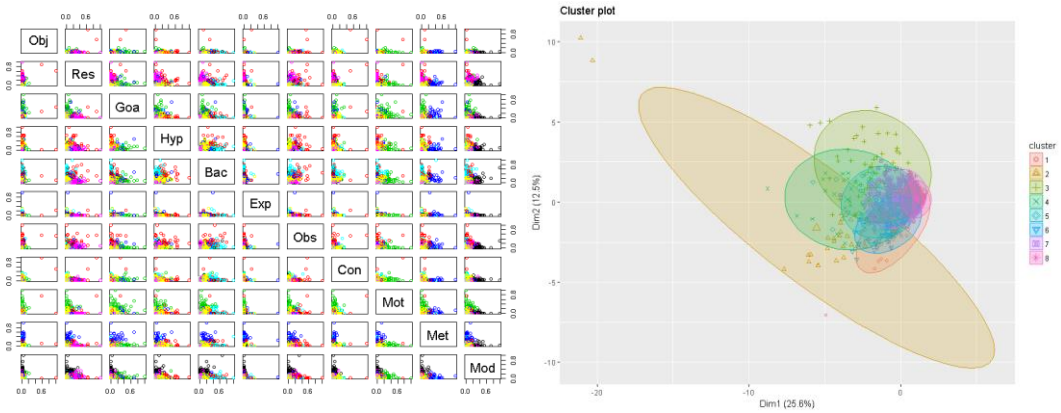

Figure 4: Cluster result in each pair of dimension (left), K-means cluster result overview (right)

The graph illustrate K-means clustering results in each dimensions and overview space. The concept that has been mentioned in last part that data has many overlap cross dimension has been proved from left figure. By compared clustering result with ground truth, we have table below:

	Corr	Essay	Opinion	Perspective	Research	Review	View point	Case study	silhouette
Truth	100	200	93	200	200	314	74	109	
Kmeans	56	142	46	176	329	427	45	69	0.151

This table shows that K-means algorithm can clustering 8 clusters properly represent majority tendency of the rate of points' numbers. However, due to the data overlapping cross dimensions, the points tend to be assigned to major cluster then minor cluster, which course over count on major clusters and under count on minor clusters. And we perform silhouette calculation and the result indicate that k=8 k-mean cluster on this data set, has a very weak cluster structure.

In order to find best k, we search different k with their silhouette value, and result showing below:

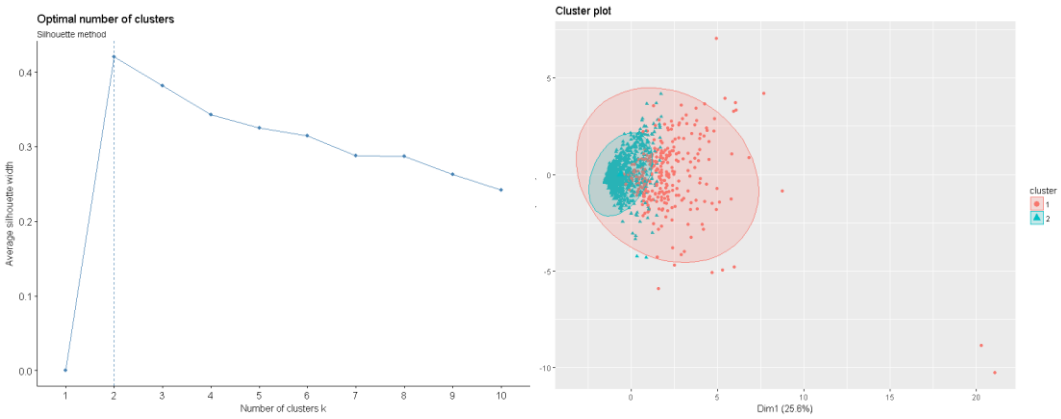

Figure 5: Silhouette plot under different K (left), K=2 K-means clustering result (right)

The silhouette values indicate that optimal cluster number for K-means is 2, and we perform K-means under k=2, the silhouette value boost significantly, at 0.42. Which indicate a reliable structure has been found by clusters.

The final clustering method that been implemented was agglomerative hierarchical clustering method. It is works in a bottom-up manner. Each points is initially considered as a leaf (single-element cluster). At each step, it will combine two most similar cluster in to a new bigger cluster. And it until all data has been assign into 1 cluster.

We using hclust method as cluster function and choose link-method as ward.D2, the result is showing below, silhouette at 0.374:

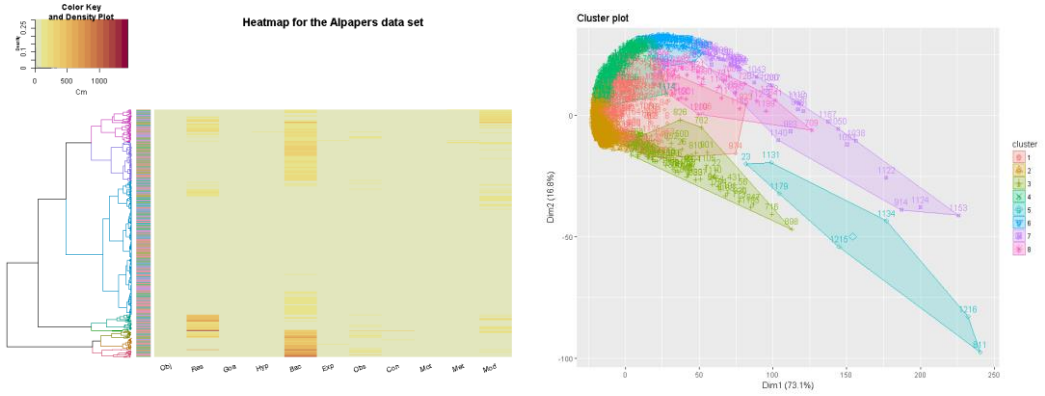

Figure 6: Hierarchical clustering structure (left), Hierarchical clustering result (right)

3、 Text Classification

3.1 Pre-processing

The pre-processing methods is following what have been done at nature language processing assignment 1. We first tokenized the sentence, and then in each sentence we remove all pure number, non-alphabetic characters and short than 3 words. After that we reconnect all sentence in to a new sample.

3.2 Features

TF-IDF weighting is the term frequency and inverse document frequency, to produce a composite weight for each term in each documents. To calculate TF-IDF feature, we first need to calculate invers document frequency, given N as numbers of documents in a collection and

df as document frequency, the inverse document frequency for a term t as follow:

$$idf_t = \log(\frac{N}{df_t})$$

And $\text{tf-idf}_{t,d}$ represent term frequency and inverse document frequency for a term (word or ngram0) in a documents as:

$$tf - idf_{t,d} = tf_{t,d} \times id_f$$

The highest TF-IDF value means term t occurs many times within small number of documents, which leading to higher discriminating power to those documents. In that case, we could use it as feature of each documents. Hear is top ten TF-IDF weights from training set:

mln	dlrs	tonnes	gas	pct
0.038091	0.030969	0.030816	0.030795	0.029533
yen	year	palm	japan	trade
0.029533	0.024978	0.024842	0.022657	0.021973

If we calculate TF-IDF cross all documents for all terms, the feature should have a great dimension which we should perform Latent semantic analysis (LSA) to reduce dimensions on features. As a dimensionality reduction techniques that can projects document to a lower-dimensional semantic space, singular value decomposition (SVD) is the key for this methods.

As we have built a co-occurrence matrix $(|D| \times |V|)$, we decompose this matrix as:

$$X = USV^T$$

Where: U and V are unitary matrices and S is a diagonal matrix. And U is the exactly reduced document vectors we needed.

3.3 Models

In this work, we have used three types of classifiers, which decision trees, K-nearest neighbours and full connected neural networks. The basic concept of decision trees, K-nearest neighbours method has already been introduced in previous work , due to the limitation of pages on this paper, we will only introduced the detail of implementation on full connect neural networks.

A four layers neural networks has been implement as classifier, this structure includes one input layer which contains 50 neuron same as the size of input feature vectors. And 2 hidden layer with size 100 and 50 respectively, in those hidden layer rectified linear unit （Relu） function has been set as activation function to accelerate learning speed due to better gradient propagation than other activation function.:

$$\text{ReLU} = \max(0, x)$$

Those hidden layer created two hyper space for data and it hyper feature that relevant to output. Automatically update optimal parameters for global optimal result.

And output layer includes contains 10 sigmoid neuron to output probability vector for 10 labels. The output range is on [0, 1].

$$sigmoid = \frac{1}{1+e^{-x}}$$

The neural networks training based on gradient decent method, and the key for the descending is to calculate derivate for certain point on the gradient space. The procedures are called as forward propagation and backward propagation.

In forward propagation, each layers contains two main parts, first a linear forward propagation, which represent as:

$$Z^{[L]} = W^{[L]} + b^{[L]}$$

And activation forward propagation：

$$A^{[L]} = g^{[L]}(Z^{[L]})$$

And back propagation can be illustrated as:

$$dZ^{[L]} = A^{[L]} - Y$$

$$dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L]T}$$

$$db^{[L]} = \frac{1}{M} np.sum(dZ^{[L]}, axis = 1, keep\,\dim = TRUE)$$

$$dZ^{[L-1]} = dW^{[L]T} dZ^{[L]} g'^{[L]}(Z^{[L-1]})$$

3.4 Evaluation

Before evaluation, we should be clarified about basic concept of classification result evaluation. As we have predicted result and it ground truth labels, we can count a confusion matrix which has 2 rows and 2 columns respectively. In convenient to presentation, we set tp, tn, fp, fn as true positive, ture negative, false positive and false negative. So we can define, precision and recall as:

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

And F-score to count both on precision and recall:

$$F - score(\beta) = \frac{(1+\beta^2)PR}{\beta^2 P + R}$$

Where, P as precision and R as recall.

Because we have multi-label and multi-output for each samples, we must leverage between micro average and macro average. And in this work, we choose micro average. Because output predicted result is a sparse matrix, some labels are all negative. In that case the calculation of precision will tackle problems which influence f-score calculation.

The ten cross folder validation has been implement in this work, the 10 f-scores for each classifier are showing below:

	Decision Tree	5-NN	Neural networks
1	0.941	0.939	0.958
2	0.960	1.0	1.0
3	1.0	0.92	1.0
4	0.824	0.880	0.923
5	0.824	0.939	0.980
6	0.920	0.980	1.0
7	0.88	0.960	0.960
8	0.875	0.917	1.0
9	0.917	0.958	0.957
10	0.958	1.0	1.0

As we can discovered from this table that neural networks is at 99% confidence level that outperform than other 2 classifiers.

In order to test the neural networks classifier ability, we further test model on extract test sets. And result as follow:

	Neural networks
micro-precision	0.968
micro-recall	0.975
micro-fscore	0.971

The neural networks works as great performance on the test data sets. With both proper bias and variance.

Conclusion

In this work, we have explored supervised learning and unsupervised learning method from different practical problem, and implement properly both in R and Python. Average silhouette value and ground truth comparison for evaluate clustering has been introduced, followed by classification comparison and evaluation method, we have focused on discussion on use if micro and macro average in certain situation. As result of the final experiment, we found out that neural networks has optimal ability on multi-label and multi-output classification problem.

Appendix:

Code instruction: Each part of code required different packages for function or visualization. Before run code to test works, please make sure you have read the README file in each folder individually.