Zibo Meng
Nov. 5th, 2015
Report to assignment 4 for CS570

1) Introduction

This project implemented and evaluated the k-means clustering algorithm in JAVA and IntelliJ IDEA (IDE). In this program, arff files are required as the data input files, and it can take and handle both numeric and nominal features and classes. Besides, it can take parameters to decide whether the normalization will be used in the program, how much iteration for each clustering process and how many clusters will be generated.

To execute this program, you'd better import this project into IntelliJ IDEA 14.1.5 and use JDK 1.8.0_60. Besides, there is an external .jar library, jcommander-1.27.jar, which is used to handle multiple parameters. Please include it as your external library.

Parameter details as follows:

```
public class paraReader {
    @Parameter(names = {"-i", "-indir"}, description = "Input file directory")
    public String inDir;

    @Parameter(names = {"-o", "-outdir"}, description = "Output file directory")
    public String outDir;

    @Parameter(names = {"-d", "-delimiter"}, description = "Delimiter symbol in input file")
    public String deli;

    @Parameter(names = {"-k"}, description = "The clusters number")
    public int clusterNum = 3;

    @Parameter(names = {"-sr"}, description = "The clusters number range start point")
    public int startRange = 3;

    @Parameter(names = {"-er"}, description = "The clusters number range end point")
    public int endRange = 3;

    @Parameter(names = {"-iter"}, description = "The iteration for each clusters number")
    public int iter = 100;

    @Parameter(names = {"-m"}, description = "The maximum iteration to find centroid")
    public int maxIter = 100;

    @Parameter(names = {"-n"}, description = "Normalization or not, 0: non-normalization, 1: normalization")
    public int norm = 0;
}
```

Usage:

Set parameters like

-i dat/baskball.arff –o output.txt –sr 2 –er 20 –iter 100 –n 0

Recompile, run and wait for result.
Happy done.

2) Dataset choosing and preprocessing
   a. dat/baskball.arff
      5 features without class and all these features are numeric
      96 instances
   b. dat/cloud.arff
      7 features without class and part of these features are numeric, others are nominal
      108 instances
   c. dat/mushroom.arff
      22 features with class and all these features are nominal
      7423 instances

There is no missing data in all of these datasets above and there is no obvious correlation between features in these datasets. Thus, we don't need to do the data preprocessing before clustering.

3) Evaluation methods and quality metrics:
   For dataset without class (unsupervised):
   Calculate and compare their Sum of Squared Error (SSE) and Between cluster Sum of Squares (BSS) value

   For dataset with class (supervised):
   a. Calculate and compare their SSE and BSS values
   b. Calculate their accuracy, precision and recall values

4) Results, comparing and analysis
   In this process, *SimpleKMeans* and *EM* in Weka are chosen as the comparator to get the result comparison.

   a. Results and compare for unsupervised:

For baskball.arff, non-Normalization, best in 100 iteration for myKMeans, 1 iteration result for KMeans in Weka

		K=2	K=3	K=4	K=5	K=7	K=10	K=13	K=16	K=19
myKMeans	SSE	7199	5402	4182	3493	2485	1806	1376	1132	948
	BSS	230	695	1557	4419	8543	16210	26446	41194	64378
KMeans Weka	SSE	7207	5699	4185	3701	2951	2139	1674	1452	1128
	BSS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

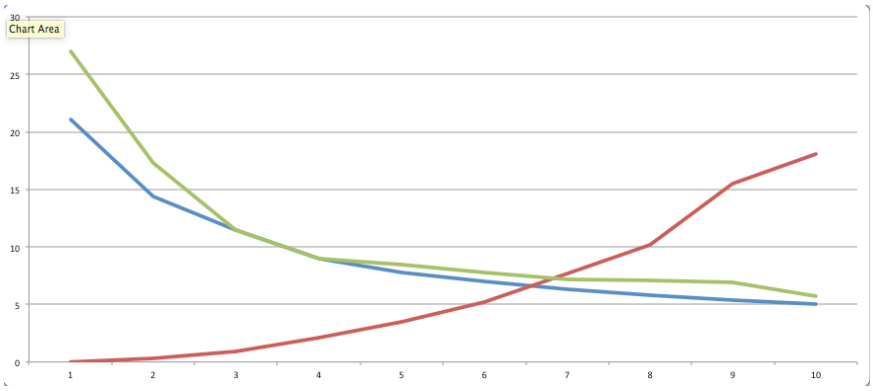
For baskball.arff, Normalization, best in 100 iteration for myKMeans, 1 iteration result for KMeans in Weka

	K=2	K=3	K=4	K=5	K=7	K=10	K=13	K=16	K=19

myKMeans	SSE	14.36	11.47	9.01	7.79	6.35	5.02	4.355	3.884	3.324
	BSS	0.282	0.945	2.11	3.52	7.72	18.6	33.0	51.09	76.17
KMeans Weka	SSE	17.28	11.49	9.01	8.46	7.205	5.683	4.953	4.324	3.563
	BSS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Analyze:

For the baskball dataset, we can compare the results of myKMeans and SimpleKMeans and generate the conclusion that myResults are very close and a little better than the WekaResults. I think it is because I repeat the KMeans 100 times to find the optimal results, which is close to the global optimal results. However, the Weka only run 1 time and will be easy to get the local optimal results. Then, we can only the normalization results to decide the optimal cluster number, because the non-normalization results and normalization results are similar.



This is the KMeans clustering SSE and BSS trend graph. In this graph, x-axis is the k value, y-axis is the SSE or BSS value. The red line is the BSS value of myKMeans, blue line is the SSE of myKMeans and green line is the SimpleKMeans of Weka. Based on the elbow principle, we can choose k=4 as the most proper cluster number.

For cloud.arff, non-Normalization, best in 100 iteration for myKMeans, 1 iteration result for KMeans in Weka

		K=2	K=3	K=4	K=5	K=7	K=10	K=13	K=16	K=19
myKMeans	SSE	102298	33731	9934.2	5289.5	3282.3	1852.0	1205.0	1002.6	826.20
	BSS	10209	31014	67621	88037	138606	357712	586748	984928	1056762
KMeans Weka	SSE	102298	33730	29086	28215	4248.4	2805.0	2457.0	2352.2	1314.9
	BSS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

For cloud.arff, Normalization, best in 100 iteration for myKMeans, 1 iteration result for KMeans in Weka

		K=2	K=3	K=4	K=5	K=7	K=10	K=13	K=16	K=19
myKMeans	SSE	101.6	84.86	71.05	59.17	35.30	19.25	15.88	13.19	11.64
	BSS	2.005	6.301	12.70	20.99	43.00	106.9	189.6	289.3	434.3
KMeans	SSE	102.3	88.48	72.68	59.13	44.36	30.50	16.71	15.79	15.45

Weka	BSS	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

For cloud dataset, it's very similar to the baskball dataset except the nominal features in dataset. But we still can generate the SSE and BSS value trends and get the most proper cluster number is 7.

b. Results and compare for supervised:

For mushroom.arff, normalization, best in 100 iteration for myKMeans

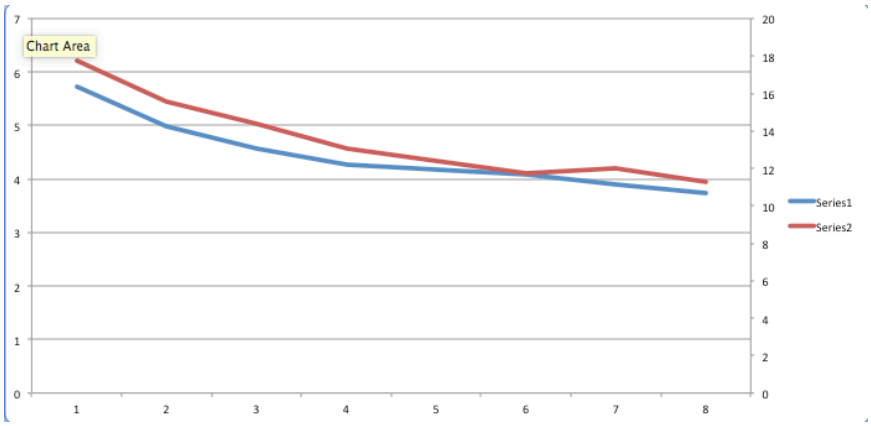
		K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9
myKMeans	SSE/10^4	5.72	4.98	4.56	4.27	4.17	4.08	3.89	3.73
	BSS	22	66	132	220	330	462	616	792
	Ave. Precision	89.5%	89.7%	90.2%	96.8%	95.2%	92.5%	89.9%	90.7%
	Ave. Recall	89.1%	89.3%	89.9%	96.7%	91.0%	92.3%	89.6%	90.8%

For mushroom.arff, 1 iteration for EM in Weka.

EM	K=2	K=3	K=4	K=5	K=6	K=7	K=8	K=9
Log Likelihood	-17.76	-15.59	-14.40	-13.03	-12.42	-11.74	-11.98	-11.28

Analyze:

For the mushroom dataset, its features are all nominal and it contains the class. From the results of myKMeans and the EM results from Weka, we can discover the most proper cluster number of mushroom is k = 5, because its average precision and recall are largest when k = 5. To check its correctness, we can get the SSE, BSS and EM Log likelihood value trend graph.



The blue line is the SSE/10^4 value from myKMeans and the red line is the Log Likelihood * (-1) value from the EM in Weka. We can easily find that the most proper cluster number is k = 5 by using the elbow principle.