Data Mining and Machine Learning.

Coursework 2.

Ivan Gee

Daniyar Nazarbayev

https://drive.google.com/open?id=16Ho902Pd7uXBMbr42giGNgCw70QwY-JR

Introduction.

First of all, I want to draw a comparison between supervised and unsupervised learning. At first I did not understand the difference between one or another, but now I think I have got a grasp on it.

In KNN (K Nearest Neighbours), the algorithm would have an x amount of points (that it got from its instances), and they points have coordinates (that are calculated with the attributes). The data set is divided into training and testing sets. The training set keeps its class attribute, while in our testing we assume that it has no class. Then we start calculating the K closest neighbors for each instance in the test set, to our data that already has a class (the training set). In the end, we compare the values we got in the test set to the real values of the class attribute. The point is, you have to know your class attribute in order to make a model.

In unsupervised, you do not need that. The main assumption is that you do not know the class, and you are trying to find out how to group your data set into classes. By the way, bayes nets are primarily supervised, though there are ways to use them in unsupervised manner. The whole point of bayes nets is whether the attribute x has an effect on its parent, and usually the root parent is the class attribute.

But clustering is pure unsupervised, and while you can keep the class attributes to check for your accuracy, in the end, in the real world, you will have none.

K-mean clustering algorithm is very similar to KNN by the way. You calculate all your instances and their coordinates, but then, you put a set amount of centroids, with random coordinates. The points get assigned to the nearest centroid, and when the whole process is done, centroids' coordinates are re-positioned (by finding the mean of all the points in its cluster), and then all the points are recalculated again, and get assigned to the closest centroid. That continues until the centroids' mean stops changing. The process has to be done multiple times since the centroids' starting positions are random.

Part 1 and 2.

Naive Bayes Nets.

I used the same weka file from the previous coursework, with around 1000 attributes. The run time was around 10 seconds. To think of it, I feel that the naive bayes nets are basically bayes nets with parent being equal to 1, meaning that each attribute is independent of each other, which is unrealistic, hence naive.

1000 attributes, with split 66% and 34% gave me accuracy 22%.

Then I tested the full dataset (2304 attributes), with the same split, and it gave me 21%.

Then I created a smaller dataset (which I used for the next task), which consisted of around 250 attributes. It gave me accuracy 27%. I am going to explain what methods I used for feature reduction in the next part.

After that I kept using that dataset.

The naive bayes net with top 14 attributes gave 26%. Top 35 gave 25% and top 70 gave 24%.

Bayes Nets.

The 1000 attribute dataset would not even get past the model making stage in Weka, when the parent parameter was higher than 1.

With parent=1, it would give 24% accuracy.

Anyway, I wanted to implement some new feature reduction algorithms, specifically geared for bayes nets. For that I used classifier attribute evaluation, where the classifier was set to bayes net (k2, p=1). I read the theory and it said, that it removes each attribute, one at a time and runs the same classifier algorithm over and over again. It then states which attributes affected the accuracy in both positive and negative way, and which ones did not, in comparison to the full data set accuracy. Then, I took the first 250 attributes (up to 0.0004).

The reduced data set actually was a big improvement. **Here are the results:**

28% on k2, p=1.

32% on k2, p=2.

34% on k2, p=3.

35% on k2, p=4.

32% on TAN.

28% on hill climbing, p=1.

I would have done more tests, but a lot of these would take at least 10 minutes to run.

Top 14 attributes results are:

26% on k2, p=1.

29% on k2, p=2.

29% on k2, p=4.

30% on TAN.

26% on hill climbing, p=1.

28% on hill climbing, p=4.

Top 35 attributes results are:

26% on k2, p=1.

30% on k2, p=2.

29% on k2, p=4.

30% on TAN.

26% on hill climbing, p=1.

Top 70 attributes results are:

26% on k2, p=1.

30% on k2, p=2.

31% on k2, p=4.

26% on hill climbing, p=1.

30% on hill climbing, p=2.

31% on TAN.

Conclusions.

Bayes net k2 or hill climbing, with parent=1 is basically the same as naive bayes net.

K2 is somewhat better than hill climbing. It is said that they are the same algorithm, but one uses ordering (K2) and the other does not (hill climbing). But I did not notice any huge difference in performance or accuracy.

TAN automatically finds the right amount of parents. At most TAN would give 2 parents in all of my cases.

Task 5

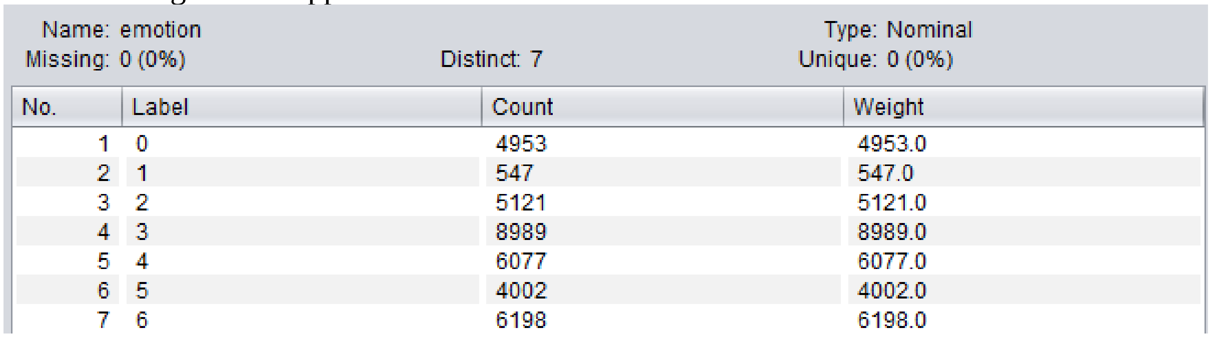
For task 5 we ran the K means algorithm on our entire randomised data set. First using the classes to clusters option on the emotion attribute. We ran the test with 10 different seeds to change the values of the randomised initial centroids. This helped us produce 10 inaccuracies for our data which proved to be all very similar on our tests.

Seed number	Percentage of Clustered Instances per emotion for different seed values
1	0 5648 (16%) 1 5188 (14%) 2 4656 (13%) 3 4633 (13%) 4 5178 (14%) 5 5083 (14%) 6 5501 (15%) Incorrectly clustered instances : 29121.0 81.1464 %
2	0 5020 (14%) 1 5677 (16%) 2 4623 (13%) 3 5209 (15%) 4 5171 (14%) 5 4645 (13%) 6 5542 (15%) Incorrectly clustered instances : 29116.0 81.1324 %
3	0 5643 (16%) 1 5190 (14%) 2 4642 (13%) 3 5532 (15%) 4 4643 (13%) 5 5211 (15%) 6 5026 (14%) Incorrectly clustered instances : 29110.0 81.1157 %
4	0 5642 (16%) 1 4641 (13%) 2 5504 (15%) 3 5231 (15%) 4 5190 (14%) 5 5025 (14%) 6 4654 (13%) Incorrectly clustered instances : 29109.0 81.1129 %
5	0 5490 (15%) 1 4693 (13%) 2 5652 (16%) 3 5051 (14%) 4 4538 (13%) 5 5189 (14%) 6 5274 (15%) Incorrectly clustered instances : 29130.0 81.1715 %
6	0 4305 (12%) 1 5047 (14%) 2 5747 (16%) 3 5694 (16%) 4 4657 (13%) 5 5006 (14%) 6 5431 (15%) Incorrectly clustered instances : 29126.0 81.1603 %

7	0 4804 (13%) 1 4780 (13%) 2 4592 (13%) 3 5449 (15%) 4 5565 (16%) 5 5176 (14%) 6 5521 (15%) Incorrectly clustered instances : 29151.0 81.23 %
8	0 4688 (13%) 1 5258 (15%) 2 4648 (13%) 3 4891 (14%) 4 5290 (15%) 5 5538 (15%) 6 5574 (16%) Incorrectly clustered instances : 29154.0 81.2383 %
9	0 5649 (16%) 1 4638 (13%) 2 4643 (13%) 3 5535 (15%) 4 5168 (14%) 5 5029 (14%) 6 5225 (15%) Incorrectly clustered instances : 29106.0 81.1046 %
10	0 5095 (14%) 1 4582 (13%) 2 5184 (14%) 3 4664 (13%) 4 5548 (15%) 5 5661 (16%) 6 5153 (14%) Incorrectly clustered instances : 29135.0 81.1854 %
Max Min Average	81.2383 % 81.1046 % 81.15975 %

Task 6

The following is the snippet of the labelled instances from the full data set.



Name: emotion Type: Nominal
Missing: 0 (0%) Distinct: 7 Unique: 0 (0%)

No.	Label	Count	Weight
1	0	4953	4953.0
2	1	547	547.0
3	2	5121	5121.0
4	3	8989	8989.0
5	4	6077	6077.0
6	5	4002	4002.0
7	6	6198	6198.0

If we compare the running of the K means algorithm to the snippet we can see definitely that there is a lot of inaccuracy in the data compared to the percentages of clustered instances given from each different seed. The most distinctive one would be label 1. The data set describes it as having a value of only 547. While the average of this emotion across all of the K means instances was 4969. This could definitely contribute largely to the overall inaccuracy. As even a small percentage change of 1% would be an inaccuracy of 358.87 instances each time.

Task 7

We ran the destiny based algorithm on our full data set. Its results were as follows:

```
Clustered Instances
0      18145 ( 51%)
1      17742 ( 49%)
Class attribute: emotion
Classes to Clusters:
    0    1  <-- assigned to cluster
 2368 2585 | 0
  297  250 | 1
 3048 2073 | 2
 4376 4613 | 3
 2411 3666 | 4
 2908 1094 | 5
 2737 3461 | 6

Cluster 0 <-- 3
Cluster 1 <-- 4

Incorrectly clustered instances :        27845.0          77.5908 %
```

This algorithm used 2 clusters to divide the data set into 49% and 51% respectively. Within this set we can see the split of how many each instance were classed to each cluster. This was done off of the emotion attribute.

The next test we ran was manually changing the number of clusters for the K means algorithm. We decided to run the algorithm with 5 and 9 clusters to see if we had any changes in accuracy.

5 clusters	9 clusters
Clustered Instances	Clustered Instances
0 8292 (23%)	0 3840 (11%)
1 7499 (21%)	1 3721 (10%)
2 6180 (17%)	2 4293 (12%)
3 6688 (19%)	3 3761 (10%)
4 7228 (20%)	4 4883 (14%)
	5 4287 (12%)
	6 2961 (8%)
Class attribute: emotion	7 3646 (10%)
Classes to Clusters:	8 4495 (13%)
Cluster 0 <-- 5	Class attribute: emotion
Cluster 1 <-- 6	Classes to Clusters:
Cluster 2 <-- 0	
Cluster 3 <-- 4	Cluster 0 <-- 0
Cluster 4 <-- 3	Cluster 1 <-- 6

Incorrectly clustered instances : 28040.0 78.1341 %	Cluster 2 <-- 4 Cluster 3 <-- 2 Cluster 4 <-- 3 Cluster 5 <-- 1 Cluster 6 <-- No class Cluster 7 <-- No class Cluster 8 <-- 5 Incorrectly clustered instances : 30120.0 83.9301 %

Here we have 2 separate k means results for testing the optimum number of clusters. From our two tests we can see that we have a better accuracy for the lower amount of clusters by a small margin of roughly 3% compared to the average accuracy we had in task 5 0f 81%. Having 9 clusters proved to make our data even more inaccurate as it has had the highest inaccuracy of 83.9%.

Task 8

After exploring multiple algorithms on our data set we have found that the algorithms all produce reasonable results. We feel that having more time and running a lot more trials on the data set would help produce more accurate results. Bayes nets on average produced a higher accuracy of correctly defined instances of the 7 emotions given as part of the data set. The algorithms can be very intensive on the computer. Which makes it difficult to run the tests effectively on the large data set. The data set that we have been provided with tends to have a very high inaccuracy overall across all of the algorithms, which shows us that there is definitely room for change and improvement in the classifications of the data's emotions.

The one nice feature we found from using the destiny based algorithm was the std deviation and mean value for all of the pixels. With some more experimentation of using this algorithm and the data set of classed emotions we could probably improve the classification great