## Context Aware within the Same Application

### The python package InfoBeyond :

According to the theory, I build a python package which is called InfoBeyond (C:\Users\Christina).In this package, there are 5 modules:

1. affinity\_propagation: the classic affinity propagation algorithm
2. init\_affinity\_propagation: process the first batch of data
3. updated\_affinity\_progapagation: process the current batch of data with respect to the previous batch of data
4. updated\_exemp\_preference: calculate preference of the exemplars of current batch of data considering the previous data
5. update\_w\_d: update the exemplar weight and the weighted dissimilarity.

### Test on Synthetic Dataset

The synthetic data was created with different Gaussian distributions while the mean and variance of data distribution vary during data generation. We decided to present a two-dimensional data set for the sake of visualization of the context aware clustering results.

The streaming data are generated so as to follow three Gaussian distributions. The following figures (from Figure 1 to Figure 10) describe the evolution of the two-dimensional streaming data as they arrive in batches. We assume that a new batch of data arrive with 300 items at every epoch. Also, data distribution changes as new batches of data arrives (You can find the detailed information in the figures .Here we set the variance are all 1).

From the following 10 figures, we can find that our approach succeed in identifying the new emerging cluster at different epochs, while it also takes the previous defined clusters into account. The new clusters may appear or existing clusters may disappear.

Table 1 shows the weights of clusters identified at each epoch when lambda is set to 5(it is better if we can also try on 0.5 and then compare the difference later). Lambda is the decay factor which controls the impact of the historical data on the clustering process. The larger the lambda is, the less emphasis we put on historical data comparing to the most recent ones. First, we can see that when the mean and variance of Gaussian distribution just change a little bit comparing to the previous epoch, the clusters will almost stay the same. For instance, the first four epochs have the same exemplars while they have different means. Second, it can also catch the new evolving information with a view to the previously arrived data. For example, the data in the sixth epoch have four exemplars and three of them come from the previous epoch. The data items that belong to different data clusters will be assigned to different learners for classification according to the availability and efficiency of the learners.

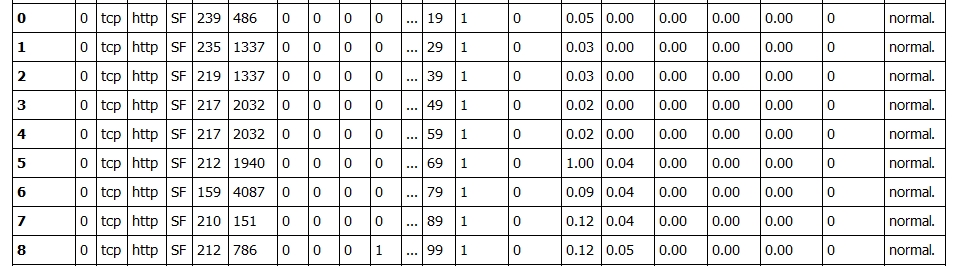
In order to improve the effectiveness of the data process system and tradeoff between the delay of data processing and the speed of data flowing, we also record the time delay along with the number of data items of every epoch in Figure 11. Obviously, the time cost gradually increases over the number of data items. It seems that the increase rate becomes lower and lower which means that our algorithms have a lower time complexity. Since I just change the number of data items from 0 to 1000, this analysis may not be convincing. Hence it is better if we can test on larger number in the further work.

The synthetic data shows the effectiveness of our approach to capture data evolution and identify the emerging clusters as data flows.

Refer to script test\_normal\_dist.ipynb and test\_delay\_number.ipynb in the final report for more detailed information.

### Test on the Real Dataset

The real dataset is the KDD CUP’99 Network Intrusion Detection data set. The data website is [http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. We use the kddcup.data](http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.%20We%20use%20the%20kddcup.data) 10 percent.gz since this data set has been labeled and we can check the clustering result later. (the text version of data set is kddcup42.txt in the final report) .

The kddcup data has 42 features and the last column is the labels. There are 7 symbolic features and 34 continuous features. First, I convert the strings into factors and then take the whole 42 features into consideration. The format of the data is in the following figure.

Every time we randomly select 1000 data items from the whole data set as a new batch. From the script of test\_kdd.ipynb in the final report, the results manifest that the exemplars only change a little bit between epochs which may mean the data itself doesn’t evolve or our approach cannot catch the new emerging clusters. What is more, the number of the exemplars is about 30 which are much more than we expect. In addition, I find that our algorithms are not robust and are sensitive to the outliers. The clustering result may change a lot while the data only change a little bit. <http://wscg.zcu.cz/wscg2012/poster/B29-full.pdf> this paper may be helpful in the further work.

With respect to the analysis of the clustering result, I think we can try to enhance the result through several other ways.

1. Selecting the 1000 data items randomly may not be appropriate. We can try to convert the data set into data stream by taking the data input order as order of streaming.
2. We should not take all the features as the input since some of the features are discrete and some features are much more important than the other. In this case, we may take the continuous features out of the total 42 variable as the input or try to select some features according to the importance.
3. We can also try to change the preference estimation method.

4)It is better if we can calculate the clustering purity.

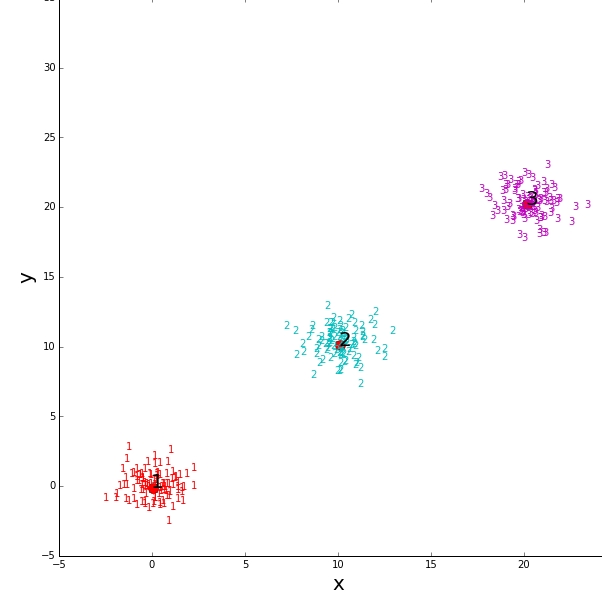


Figure 1: exemplars of epoch 1 with means (0,0),(10,10),(20,20)

1 Exemplar:(0.034745,-0.114246)

2 Exemplar:(10.114102,10.076634)

3 Exemplar:(20.154809,20.189719)

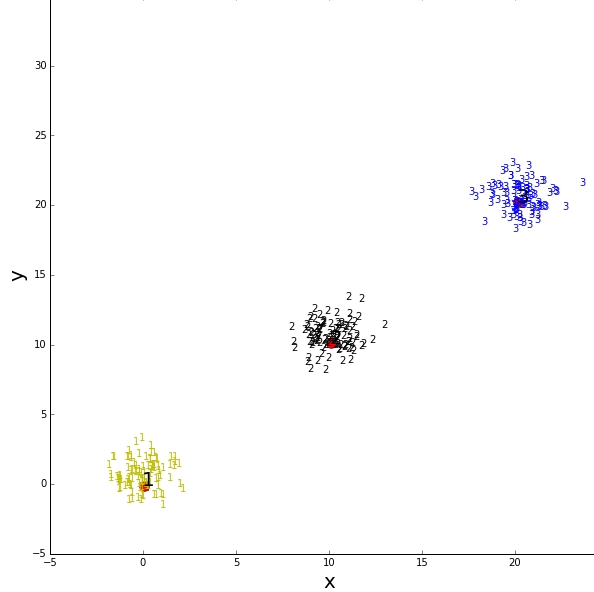


Figure 2: exemplars of epoch 2 with means (0,0.5),(10,10.5),(20,20.5)

1 Exemplar:(0.034745,-0.114246)

2 Exemplar:(10.114102,10.076634)

3 Exemplar:(20.154809,20.189719)

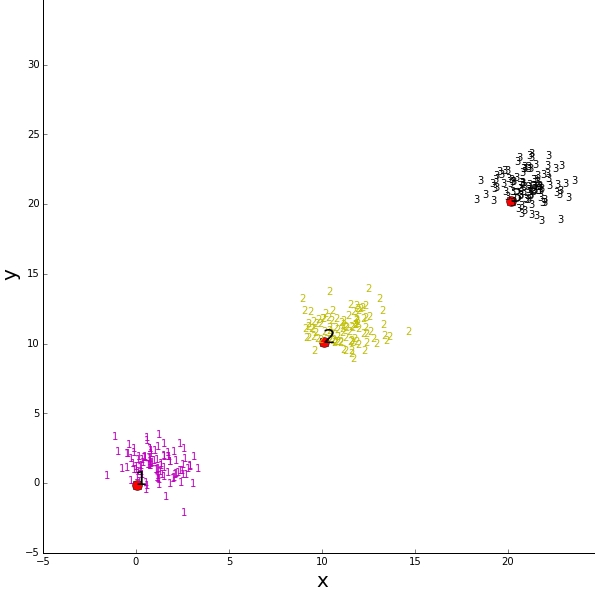


Figure 3: exemplars of epoch 3 with means (0,1),(10,11),(20,21)

1 Exemplar:(0.034745,-0.114246)

2 Exemplar:(10.114102,10.076634)

3 Exemplar:(20.154809,20.189719)

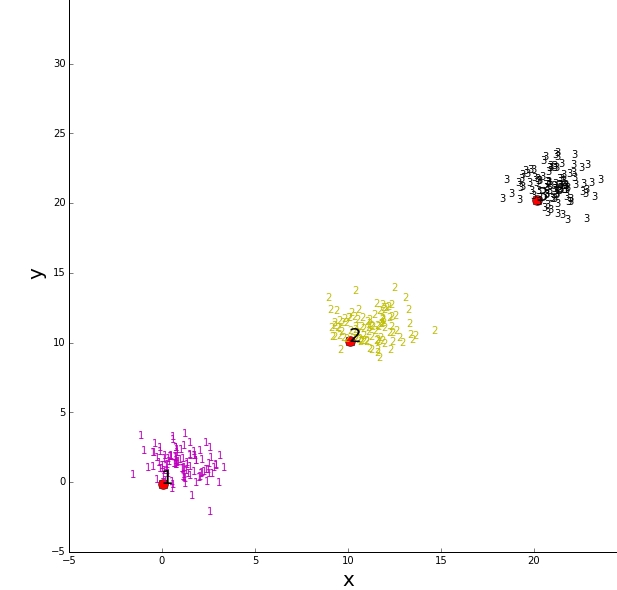


Figure 4: exemplars of epoch 4 with means (1,1),(11,11),(21,21)

1 Exemplar:(0.034745,-0.114246)

2 Exemplar:(10.114102,10.076634)

3 Exemplar:(20.154809,20.189719)

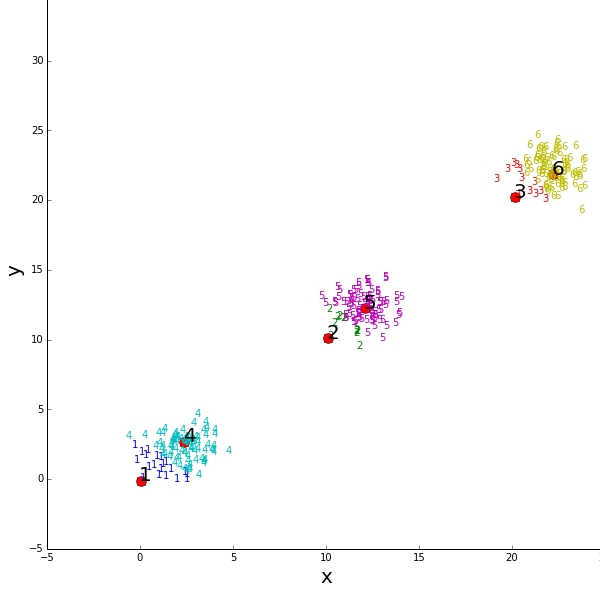


Figure 5: exemplars of epoch 5 with means (2,2),(12,12),(22,22)

1 Exemplar:(0.034745,-0.114246)

2 Exemplar:(10.114102,10.076634)

3 Exemplar:(20.154809,20.189719)

4 Exemplar:(2.343358,2.681385)

5 Exemplar:(12.113146,12.233894)

6 Exemplar:(22.206723,21.841792)

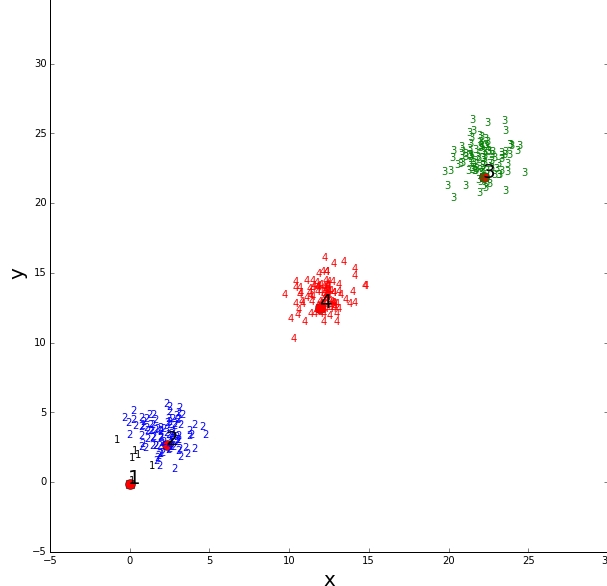


Figure 6: exemplars of epoch 6 with means (2,3),(12,13),(22,23)

1 Exemplar:(0.034745,-0.114246)

2 Exemplar:(2.343358,2.681385)

3 Exemplar:(22.206723,21.841792)

4 Exemplar:(11.968712,12.496992)

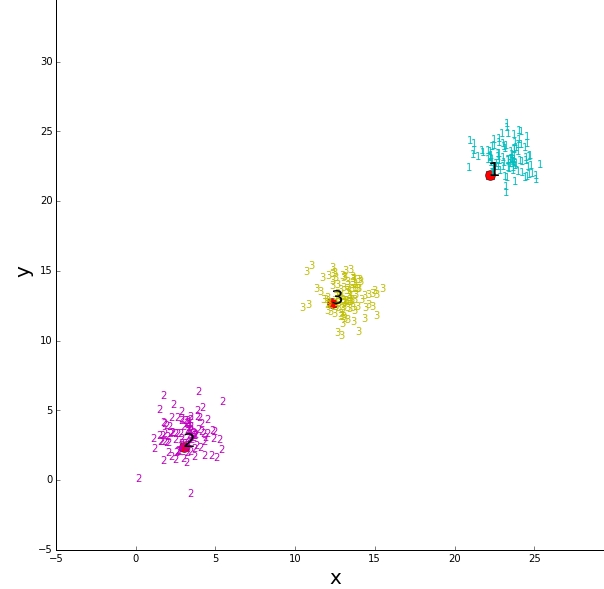


Figure 7: exemplars of epoch 7 with means (3,3),(13,13),(23,23)

1 Exemplar:(22.206723,21.841792)

2 Exemplar:(3.033131,2.404589)

3 Exemplar:(12.320900,12.666277)

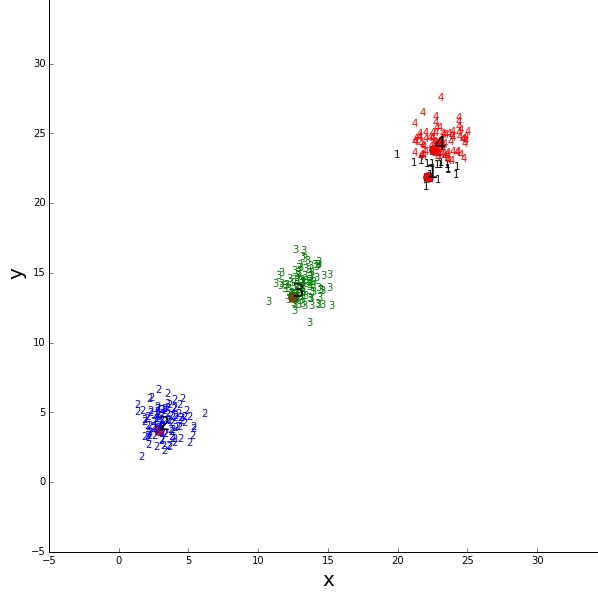


Figure 8: exemplars of epoch 8 with means (3,4),(13,14),(23,24)

1 Exemplar:(22.206723,21.841792)

2 Exemplar:(2.874167,3.753515)

3 Exemplar:(12.479956,13.292202)

4 Exemplar:(22.625526,23.811635)

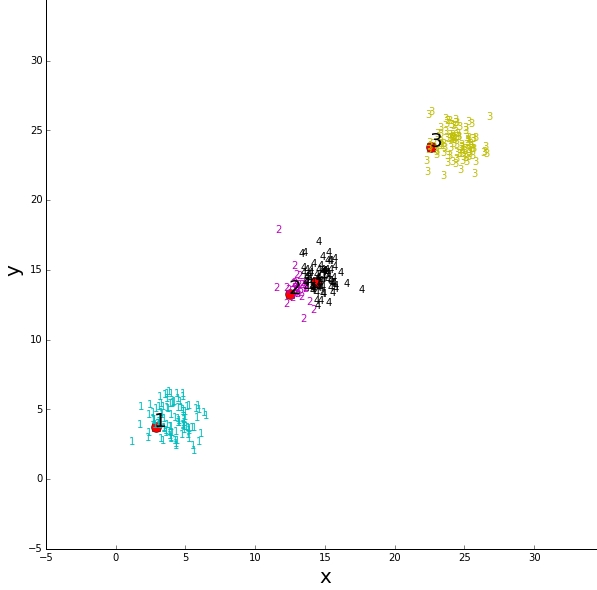


Figure 9: exemplars of epoch 9 with means (4,4),(14,14),(24,24)

1 Exemplar:(2.874167,3.753515)

2 Exemplar:(12.479956,13.292202)

3 Exemplar:(22.625526,23.811635)

4 Exemplar:(14.354309,14.132556)

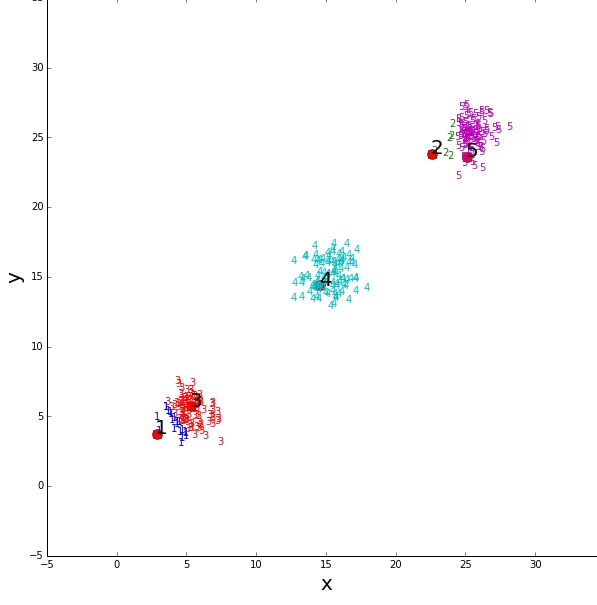


Figure 10: exemplars of epoch 10 with means (5,5),(15,15),(25,25)

1 Exemplar:(2.874167,3.753515)

2 Exemplar:(22.625526,23.811635)

3 Exemplar:(5.333827,5.716739)

4 Exemplar:(14.596032,14.387504)

5 Exemplar:(25.127873,23.585210)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **C1** | **C2** | **C3** | **C4** | **C5** | **C6** | **C7** | **C8** | **C9** | **C10** | **C11** | **C12** | **C13** | **C14** | **C15** | **C16** |
| **1** | **100** | **100** | **100** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2** | **101** | **101** | **101** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3** | **101** | **101** | **101** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **4** | **101** | **101** | **101** |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **5** | **20** | **12** | **12** | **81** | **89** | **89** |  |  |  |  |  |  |  |  |  |  |
| **6** | **6** |  |  | **96** |  | **102** | **102** |  |  |  |  |  |  |  |  |  |
| **7** |  |  |  |  |  | **101** |  | **102** | **101** |  |  |  |  |  |  |  |
| **8** |  |  |  |  |  | **16** |  |  |  | **101** | **101** | **85** |  |  |  |  |
| **9** |  |  |  |  |  |  |  |  |  | **101** | **31** | **102** | **70** |  |  |  |
| **10** |  |  |  |  |  |  |  |  |  | **21** |  | **6** |  | **80** | **102** | **95** |

Table I: cluster weight (number of data items belong to every cluster) of every epoch

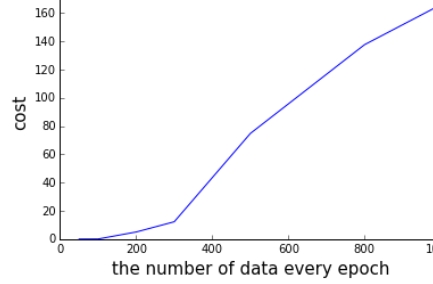


Figure 11: the time delay(s) VS the number of data items every epoch