

Lec 10

Thursday, 7 November 2024 19:33

1. Elbow method for K picking in K means. WSS - sum of squared distances of records to picked centroids. Minimizing while wss changes insignificantly. Better recall from lecture

Diagnostics – Evaluating the Model



- Do the clusters look separated in at least some of the plots when you do pair-wise plots of the clusters?
 - ▶ Pair-wise plots can be used when there are not many variables
- Do you have any clusters with few data points?
 - ▶ Try decreasing the value of K
- 2.
 - Are there splits on variables that you would expect, but don't see?
 - ▶ Try increasing the value K
 - Do any of the centroids seem too close to each other?
 - ▶ Try decreasing the value of K

K-Means Clustering - Reasons to Choose (+) and Cautions (-)



3.

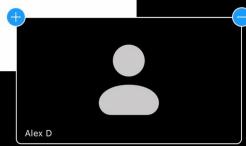
Reasons to Choose (+)	Cautions (-)
Easy to implement	Doesn't handle categorical variables
Easy to assign new data to existing clusters Which is the nearest cluster center?	Sensitive to initialization (first guess)
Concise output Coordinates the K cluster centers	Variables should all be measured on similar or compatible scales Not scale-invariant!
	K (the number of clusters) must be known or decided a priori Wrong guess: possibly poor results
	Tends to produce "round" equi-sized clusters. Not always desirable

1. The map phase
2. map_of_points Map(centroid_coordinates, all_points) - returns a map of key-value pairs that represent assignment of points to closest centroids
3. new_centroid_coordinates Reduce(map_of_points) - returns new recalculated centroids based on every cluster formed

4.

PK-means (K-means for Map-Reduce): Mapper

```
map (key, value)
Input: Global variable centers, the offset key, the sample value
Output: <key', value'> pair, where the key' is the index of the closest center point and value' is a
string comprise of sample information
1. Construct the sample instance from value;
2. minDis = Double.MAX VALUE;
3. index = -1;
4. For i=0 to centers.length do
    dis= ComputeDist(instance, centers[i]);
    If dis < minDis {
        minDis = dis;
        index = i;
    }
5. End For
6. Take index as key';
7. Construct value' as a string comprise of the values of different dimensions;
8. output <key', value'> pair;
9. End
*
```

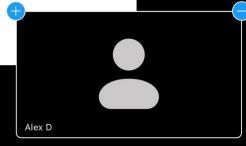


5. Combiner is sort of middleware in this pipeline that partially takes up computation from reducer

6.

PK-means (K-means for Map-Reduce): Combiner

```
combine (key, V)
Input: key is the index of the cluster, V is the list of the samples assigned to the same cluster
Output: <key', value'> pair, where the key' is the index of the cluster, value' is a string comprised
of sum of the samples in the same cluster and the sample number
1. Initialize one array to record the sum of value of each dimensions of the samples contained in
the same cluster, i.e. the samples in the list V ;
2. Initialize a counter num as 0 to record the sum of sample number in the same cluster;
3. while(V.hasNext()){
    Construct the sample instance from V.next();
    Add the values of different dimensions of instance to the array
    num++;
}
5. Take key as key';
6. Construct value' as a string comprised of the sum values of different dimensions and num;
7. output <key', value'> pair;
8. End
```



19:49 Thu 7 Nov

Zoom

Unmute Start video Share content Participants More Leave

PK-means (K-means for Map-Reduce): Reducer

```

reduce (key, V)
Input: key is the index of the cluster, V is the list of the partial sums from different host
Output: <key', value'> pair, where the key' is the index of the cluster, value' is a string representing the new center
1. Initialize one array record the sum of value of each dimensions of the samples contained in the same cluster, e.g. the samples in the list V;
2. Initialize a counter NUM as 0 to record the sum of sample number in the same cluster;
3. while(V.hasNext())/
    Construct the sample instance from V.next();
    Add the values of different dimensions of instance to the array
    NUM += num;
4. }
5. Divide the entries of the array by NUM to get the new center's coordinates;
6. Take key as key';
7. Construct value' as a string comprise of the center's coordinates;
8. output <key', value'> pair;
9. End

```

Alex D's screen

19:56 Thu 7 Nov

Zoom

Unmute Start video Share content Participants More Leave

Hierarchical Clustering Algorithm

The approach in words:

- Start with each point in its own cluster.
- Identify the closest two clusters and merge them.
- Repeat.
- Ends when all points are in a single cluster.

Dendrogram

Alex D's screen

Partially based on Statistical Learning @ Stanford teaching material

9. Hierarchical clustering is like agario lol
10. What linkage to choose to determine distance between clusters?

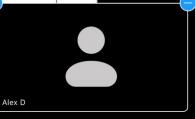
11.

Types of Linkage

A diagram illustrating four types of linkage:

- Complete:** Maximal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
- Single:** Minimal inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities.
- Average:** Mean inter-cluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
- Centroid:** Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversion*.

Partially based on Statistical Learning @ Stanford teaching material



12. Better not use with lots of data. In first 3 the matrix of distances would be too big, in the last one recalculations of centroids on every iteration will take lot of time

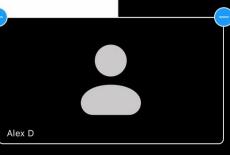
13.

Choice of Dissimilarity Measure

- So far have used Euclidean distance.
- An alternative is *correlation-based distance* which considers two observations to be similar if their features are highly correlated.
- This is an unusual use of correlation, which is normally computed between variables; here it is computed between the observation profiles for each pair of observations.

A line graph showing three observation profiles (Observation 1, Observation 2, Observation 3) plotted against time. The y-axis ranges from 0 to 20, and the x-axis shows time points 0, 5, and 10. The profiles fluctuate, with Observation 2 generally having the highest values and Observation 3 the lowest.

Partially based on Statistical Learning @ Stanford teaching material



14. DBSCAN

15. Точка опорная если её количество соседей больше чем заявленное в эпсилон окрестности

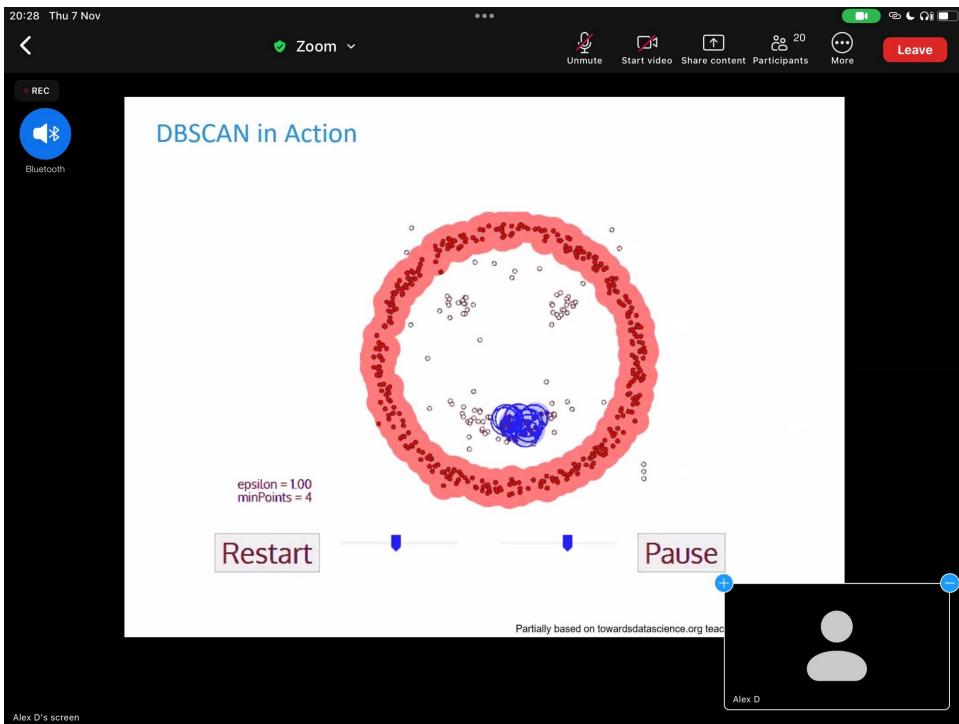
16. Связываем её опорных соседей к тому же кластеру что и она

17. Затем добавляем все точки которые захватываются опорными, но сами опорными не являются(на них формирование кластера останавливается)

18. И так волной для граничных точек, рассматривая их как опорные пока такие точки не закончатся

19. Закончится всё на граничных точках - точках которые лежат в эпсилон окрестности опорной, но сами опорной не являются

20. Шумовая точка - не граничная и не опорная



22.

Reasons to Choose (+)	Cautions (-)
No need of prior knowledge for K	Doesn't handle categorical variables
Arbitrary-shaped clusters	Non-deterministic for some (border) points
Can detect outliers	Need to choose ϵ and min_points parameters based on data domain
Can be used with databases, by utilizing indexes	Not work well with high dimensions and different clusters structures
With choose of optimal parameters and right data-structures can reach log-linear complexity	

Partially based on Dell EMC teaching

Alex D

23. Additional readings

20:36 Thu 7 Nov

Zoom

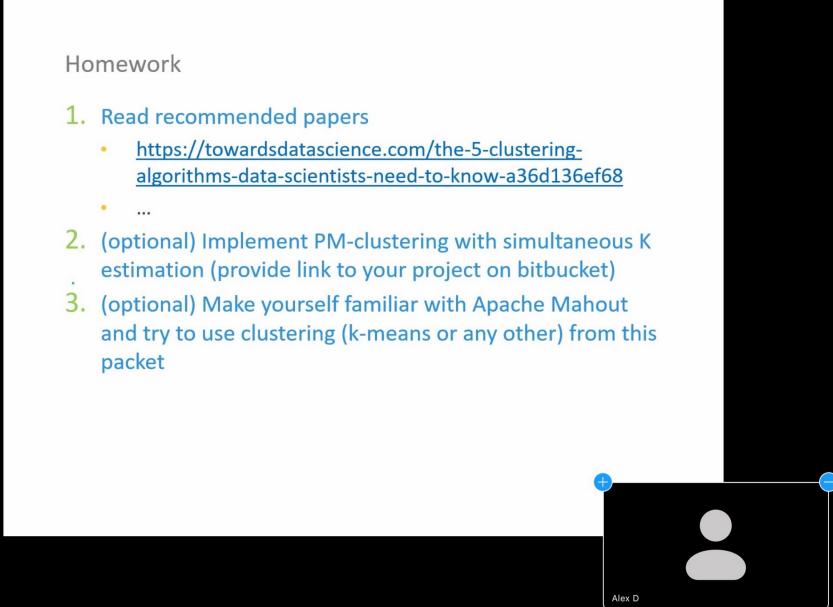
Unmute Start video Share content Participants More Leave

REC Bluetooth

Homework

1. Read recommended papers
 - <https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68>
 - ...
2. (optional) Implement PM-clustering with simultaneous K estimation (provide link to your project on bitbucket)
3. (optional) Make yourself familiar with Apache Mahout and try to use clustering (k-means or any other) from this packet

Alex D's screen



25. Assossiatin algos

26. Apriori

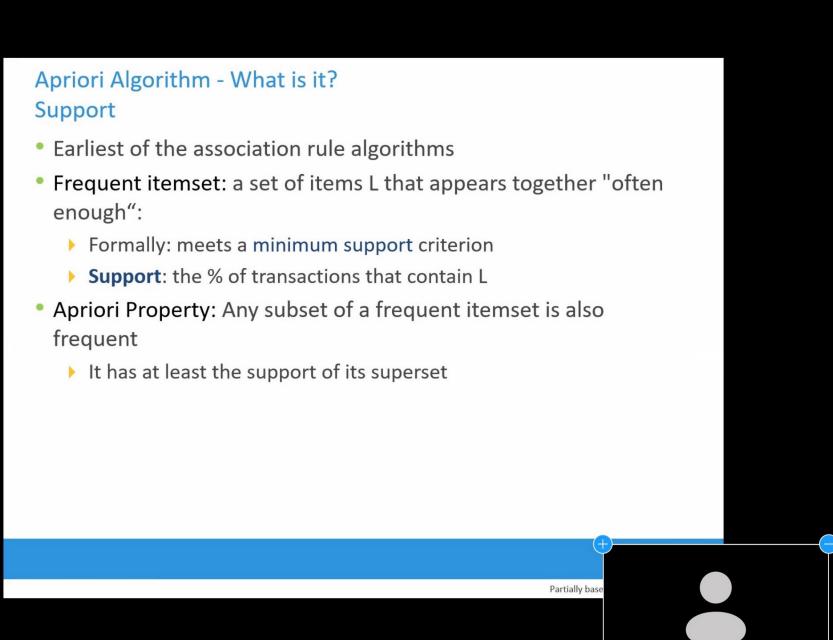
27.

Apriori Algorithm - What is it?

Support

- Earliest of the association rule algorithms
- Frequent itemset: a set of items L that appears together "often enough"
 - ▶ Formally: meets a **minimum support** criterion
 - ▶ **Support**: the % of transactions that contain L
- Apriori Property: Any subset of a frequent itemset is also frequent
 - ▶ It has at least the support of its superset

Alex D



28. **Support** - percent of occurrences

29. Superset have at least the same percentage of occurrences than set it comes from

30. More importantly - if some set isn't frequent enough than adding elements to it won't help, cause the percentage will fall

31. There is no need to go through all sets. Only through good enough, cause otherwise it'll be computationally hard

32. Forming available rules with minimum **confidence**(percentage of such occurrences)

33. But how to exclude consicencies?

Lift and Leverage

$$\text{Lift}(X \rightarrow Y) = \frac{\text{Support}(X \wedge Y)}{\text{Support}(X) * \text{Support}(Y)}$$

34.

$$\text{Leverage}(X \rightarrow Y) = \text{Support}(X \wedge Y) - \text{Support}(X) * \text{Support}(Y)$$



35. If Lift is large than there is something in it
36. Quite similar to the formula of probability of joint events



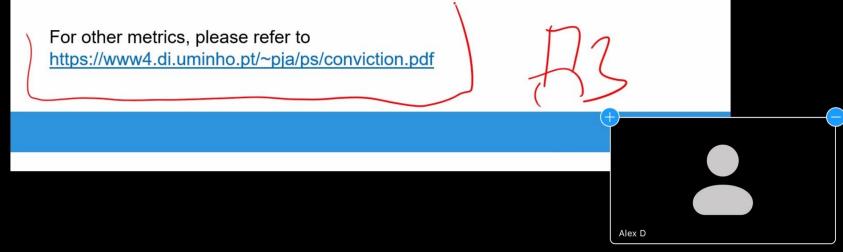
Conviction and others

$$\text{conv}(A \rightarrow C) = \frac{1 - \text{sup}(C)}{1 - \text{conf}(A \rightarrow C)}$$

37.

Conviction compares the probability that A appears without C if they were dependent with the actual frequency of the appearance of A without C. In that respect it is like Lift, however, it contrast to lift it is a directed measure. Alternative, Conviction (A → C) = P(A)P(not C)/P(A and not C)

For other metrics, please refer to
<https://www4.di.uminho.pt/~pjia/ps/conviction.pdf>



38.

A Sketch of the Algorithm

- If L_k is the set of frequent k -itemsets:
 - ▶ Generate the candidate set C_{k+1} by joining L_k to itself
 - ▶ Prune out the $(k+1)$ -itemsets that don't have minimum support
Now we have L_{k+1}
- We know this catches all the frequent $(k+1)$ -itemsets by the apriori property
 - ▶ a $(k+1)$ -itemset can't be frequent if any of its subsets aren't frequent
- Continue until we reach k_{\max} , or run out of support
- From the union of all the L_k , find all the rules with minimum confidence



39. Наращиваем порядок наборов пока не дойдём до порядка, на котором нет наборов с minimum support

Diagnostics



- Do the rules make sense?
 - ▶ What does the domain expert say?
- Make a "test set" from hold-out data:
 - ▶ Enter some market baskets with a few items missing (selected at random). Can the rules determine the missing items?
 - ▶ Remember, some of the test data may not cause a rule to fire.
- Evaluate the rules by lift or leverage.
 - ▶ Some associations may be coincidental (or obvious).

40.



41.

Apriori - Reasons to Choose (+) and Cautions (-)



Reasons to Choose (+)	Cautions (-)
Easy to implement	Requires many database scans
Uses a clever observation to prune the search space •Apriori property	Exponential time complexity
Easy to parallelize	Can mistakenly find spurious (or coincidental) relationships •Addressed with Lift and Leverage measures

Partially based on

Alex D

