CHAPTER 19. MIXTURE MODELS FOR UNSUPERVISED CLUSTERING

1) The Gaussian Mixture Model (GMM) Basis

Objective: Come up with a density estimation of X. Describe the probabilistic distribution of the data.

How? Relying on Multivariate Normal Distribution.

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})},$$

Where p(z) is the probability of belonging to one of the clusters:

$$p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}.$$

And p(x|z) is the probability of a given observation having some given values of its attributes knowing that it belongs to a given cluster:

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k=1}^{K} N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k},$$

If we suppose the data set is comprised of K clusters, then the probability distribution followed by the all the observations is the combination of the probability of an observation belonging to a given dataset and its probability to have given attribute values:

$$\begin{aligned} p(\boldsymbol{x}) &= \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}) \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \end{aligned}$$

Where π^k is the *weight* and N (x|mu, sigma) is the *Mixture Component.*

2) The Expectation Maximization Algorithm (EM) for GMM:

Once described the way GMM is structured, the Expectation Maximization Algorithm is the most efficient way to find the parameters required to provide the density estimation of the data X.

EM looks for the parameters:

$$\begin{aligned} \boldsymbol{\pi} &= [\pi_1 \cdots \pi_k] \\ \boldsymbol{\mu} &= \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K\} \\ \boldsymbol{\Sigma} &= \{\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K\} \end{aligned}$$

Which *maximizes* the log likelihood of the data:

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \log p(\boldsymbol{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{i=1}^{N} \log \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]$$

The probability of observation xi belonging to cluster K=1 is:

$$\begin{aligned} p(z_k = 1|\boldsymbol{x}_i) &= \frac{p(\boldsymbol{x}_i|z_k = 1)p(z_k = 1)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_i|z_{k'} = 1)p(z_{k'} = 1)} \\ &= \frac{\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k \boldsymbol{\Sigma}_k)\pi_k}{\sum_{k'=1}^{K} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_{k'} \boldsymbol{\Sigma}_{k'})\pi_{k'}} = \gamma_{ik} \end{aligned}$$

Total mass is the probability of all the observations belonging to a given cluster**.** And if we accumulate for all clusters, we have the dataset (since the observation must belong to some of the existing clusters it will add to 1):

$$\sum_{k=1}^{K} N_k = \sum_{k=1}^{K} \sum_{i=1}^{N} p(z_k = 1|\boldsymbol{x}_i) = \sum_{i=1}^{N} \left[\sum_{k=1}^{K} p(z_k = 1|\boldsymbol{x}_i) \right] = \sum_{i=1}^{N} 1 = N$$

Therefore, the EM algorithm can be summarized as:

Initialize: First we initialize $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ and π
Expectation step: Compute γ_{ik} for all i, k
Maximization step: Update $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ and π_k
Iterate: Repeat the two previous steps until the likelihood eq. (19.4) does not change.

Which leads to:

Algorithm 8: Expectation-Maximization algorithm

1: Initialize $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ and π_k for $k = 1, \ldots, K$
2: **while** The likelihood $\mathcal{L}$ changes **do**
3: Update $\gamma_{ik} = \frac{\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k \boldsymbol{\Sigma}_k)\pi_k}{\sum_{k'=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{k'} \boldsymbol{\Sigma}_{k'})\pi_{k'}}$ *(E-step)*
4: Update the parameter values in this order (where $N_k = \sum_{i=1}^{N} \gamma_{ik}$): *(M-step)*
5: $\mu_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} \boldsymbol{x}_i$
6: $\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T$
7: $\pi_k = \frac{N_k}{N}$
8: Compute the likelihood $\mathcal{L} = \sum_{i=1}^{N} \log \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]$
9: **end while**

- **Problems with EM algorithm:**

1) EM maximize the log likelihood [$loglikelihood\ L(\pi, \mu, \sum)$], but it's not guaranteed to behave well. It depends on the initialization parameters.
2) The algorithm can diverge when there is only one observation for a K=k cluster (Worse with random initialization of parameters ($\pi, \mu, \sum$). Therefore, its recommended to **initialize with K-means parameters**. (Can be compensated including a regularization parameter to $\sum k$.

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T + \lambda \boldsymbol{I}$$

3) The number of parameters required to run the algorithm is:

$$\underbrace{(K-1)}_{\boldsymbol{\pi}} + \underbrace{KM}_{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K} + \underbrace{K(M+1)M/2}_{\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K}$$

But K(M + 1)M/2 can be brought down by considering a covariance matrix to KM.

3) Cross-Validation and Gaussian Mixture Model (GMM)

Since GMM aims to maximize the log likelihood, a way to evaluate its performance is to measure the log likelihood in a test set X^{test}.

$$\begin{aligned}\mathcal{L}^{\text{test}}(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \log p(\boldsymbol{X}^{\text{test}} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \\ &= \sum_{i=1}^{N^{\text{test}}} \log \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i^{\text{test}} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right]\end{aligned}$$

To find the number of mixture components K we can apply Cross-Validation, and also, to select the regularization term in case of divergence.

CHAPTER 20. Density Estimation

Objective: Based on the estimated probabilistic distributions of the data computed with Gaussian Mixture Model density estimation measures try to find observations that are *different* form the others.

This can be done by looking for observations that lie in are with low density (areas where it's not expected to find observations). Therefore, the simplest way to differentiate observations is by estimating their density.

- Gaussian Mixture Model (GMM. Chapter 19) is the main tool for describing density estimations although it can be expensive for large datasets, affected by initialization values, difficulties treating areas with different densities.
- Kernel Density Estimation (Deterministic Method): Approximation to GMM by simplifying its parameters.
- Average Relative Density (Deterministic Method): Method not based in probabilities.

20.1) Kernel Density Estimator (KDE)

Since GMM has many parameters to be found to describe the density estimation and it depends on the initialization values.

GMM needs (number of clusters) K*3 parameters $(\pi, \mu, \sum)$

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

KDE assumes there are **K=N components**, **centred on each data point** and with **diagonal covariance matrix**.

$$\pi_k = \frac{1}{N}, \quad \mu_k = \boldsymbol{x}_k \quad \text{and} \quad \Sigma_k = \lambda^2 \boldsymbol{I}$$

Where λ is known as the *Kernel Width*. This leads to a probabilistic density of:

$$p_\lambda(\boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_i, \lambda^2 \boldsymbol{I}).$$

20.1.1) Selecting the Kernel Width (λ):

The kernel width can be chosen in a similar way as how GMM would select the number of clusters K, by cross-validating an comparing with a test set X^{test} and the log Likelihood of the data.

$$\begin{aligned}\mathcal{L}(\lambda) &= \log p(\boldsymbol{X}^{\text{test}}|\lambda) \\ &= \sum_{j=1}^{N^{\text{test}}} \log p(\boldsymbol{x}_j^{\text{test}}|\lambda) \\ &= \sum_{j=1}^{N^{\text{test}}} \log \left[\frac{1}{N}\sum_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_j^{\text{test}}|\boldsymbol{x}_i, \lambda^2\boldsymbol{I})\right]\end{aligned}$$

9

Leave-one-out (LOO) cross validation can be performed quickly by pre-computing for each pair of observations its density estimation knowing the other one is in the cluster:

$M_{i,j} = N(x_j | x_i, \lambda^2 I)$

$$\mathcal{L}(\lambda) = \frac{1}{N}\sum_{i=1}^{N} \log \left[\sum_{j\neq i} \frac{1}{N-1} M_{ij}\right]$$

***Uneven densities and GMM**

When trying to find outliers in the setting where we have 2 different uneven clusters, both KDE and GMM struggle to tell difference between outliers and may consider the wrong observation as candidate of outlier.

GMM may overfit leading us to rely on CV to find K, and may lead to artificially high-density regions. Making this density estimation model suitable for elongated observations along one given direction. Also is prone to spurious behaviour due to the way EN places the center of the clusters in each run. (Hence, when used for outlier detection GMM should have its parameters wisely chosen, CV and not include potential outliers in the training data).

Since KDE uses the same Kernel Width for all the data, it is unable to tell the difference in densities.

20.2) Average Relative Density (ARD)

This non probabilitically based method, tries to handle clusters with different densities, but with comparison between neighbours' relative density.

KNN (K-Nearest-Neighbour)

The K-observations closer to x:

$N_x(x,K)$= {The K observations in X which are nearest to x}

Then the average distance between neighbours (x') to x is:

$$\frac{1}{K}\sum_{\boldsymbol{x}'\in N_{\boldsymbol{X}}(\boldsymbol{x},K)} d(\boldsymbol{x},\boldsymbol{x}')$$

Where d' is a distance measure, for instance Euclidian distance: $d(x,y)= ||\sum(x-y)^2||^{1/2}$

If distance is high it means that there is low density and viceversa then , the density based on distance between observations is given by (the inverse of the sum of the distance between

neighbours, without taking on account the given observation upon which we are computing the density):

$$\text{density}_{\boldsymbol{X}_{\setminus i}}(\boldsymbol{x}_i, K) = \frac{1}{\frac{1}{K}\sum_{\boldsymbol{x}' \in N_{\boldsymbol{X}_{\setminus i}}(\boldsymbol{x}_i, K)} d(\boldsymbol{x}_i, \boldsymbol{x}')}$$

Average Relative Density ARD looks for those points whose density is smaller than its neighbors.

$$\text{ard}_{\boldsymbol{X}}(\boldsymbol{x}, K) = \frac{\text{density}_{\boldsymbol{X}}(\boldsymbol{x}, K)}{\frac{1}{K}\sum_{\boldsymbol{x}_j \in N_{\boldsymbol{X}}(\boldsymbol{x}, K)} \text{density}_{\boldsymbol{X}_{\setminus j}}(\boldsymbol{x}_j, K)}$$

Here K refers to KNN

EXAMPLE:

In case we only consider 1 neighbor (K=1). AverageRelativeDensity looks for the closer observation to x in the dataset and performs:

$$\text{ard}_{\boldsymbol{X}}(\boldsymbol{x}, 1) = \frac{\text{density}_{\boldsymbol{X}}(\boldsymbol{x}, 1)}{\text{density}_{\boldsymbol{X}_{\setminus j}}(\boldsymbol{x}_j, 1)}$$ *i.e.* $$\text{ard}_{\boldsymbol{X}}(\boldsymbol{x}, 1) = \frac{\frac{1}{d(\boldsymbol{x}, \boldsymbol{x}_j)}}{\frac{1}{d(\boldsymbol{x}_j, \boldsymbol{x}_k)}} = \frac{d(\boldsymbol{x}_j, \boldsymbol{x}_k)}{d(\boldsymbol{x}, \boldsymbol{x}_j)}$$

Where the denominator is the distance to the clolser observation without taking on account x (Observation of interest) to prevent from bias estimations.

CHAPTER 21. Association Rule Learning

Objective: Based on a specific dataset X where it is represented the transactions made by different clients (every row) and in every column the positive or negative purchasing of an item.

So X is an N (trades) x M (items) **binary** matrix.

Association mining relies on the Apriori Algorithm to predict the likelihood of a new buyer purchasing a given item based on another item they are buying.

21.1 BASIC CONCEPTS

To work with the association mining dataset, we will use *Set Notation.*

A given transaction is denoted by: $t_i = \{I_2, I_3\}$

Set notation

Given 2 sets **r={I_1,I_2,I_4}**, **s={I_2, I_3 , I_4 , I_5}**

Size	$\|r\| = 3$ and $\|s\| = 4$
Intersection	Both set share items $r \cap s = \{I_2, I_4\}$
Union	Combination of items in both elements (if repeated only show once): $r \cup s = \{I_1, I_2, I_3, I_4, I_5\}$
Empty set	$\emptyset = \{\}$
Disjoint sets	Elements don`t share any item Empty intersection $r \cap u = \emptyset$
Membership	An element is in set $x \in r$.
All elements contained	The subset of {I_2, I_4} is contained in set r:ç $\{I_2, I_4\} \subseteq r$
Remove elements in one set	Taking one set and removing the intersection set with another one: $r \setminus s = \{I_1\}, \quad \text{and} \quad s \setminus r = \{I_3, I_5\}$

21.1.1 Itemsets and Association Rules

Set of all items I={I_1,I_2,..I_M}. Set of all transactions T={T_1,T_2,..T_M}.

All elements in transactiom t_i are contained in the vector I of the items $t_i \subseteq I$. True, every transcation is composed of items, so since I vector contains all the possible ones, it is verified.

Not only the transactions are itemsets, we can creates separate ones, and they will still be subsets.

***ASSOCIATION RULE:**

$$X \to Y, \qquad X, Y \subseteq I \text{ and } X \cap Y = \emptyset$$

Where X and Y are subsets of I and they are disjoint. And it can be read as:

"People who buy X tend to buy Y." An association rule must follow the following cirteria:

- **<u>High Support:</u>** An association rule must refer to a joint event that happens often. Many people must buy X and Y**. Support is the fraction of transactions which contains X.**

$$\begin{aligned} \operatorname{supp}(X) &= \frac{\{\text{Number of transactions containing } X\}}{N} \\ &= \frac{|\{t \in T | X \subseteq t\}|}{N}. \end{aligned}$$

  "Set of transactions t in T such that X is contained as a subset of t."

  A support of an association rule (X->Y) means the support of both X and Y.

  Probabilistic singnificance of support is the empirical probability of a dataset.

$$\operatorname{supp}(X) = p(X) = p(I_3 = 1, I_5 = 1)$$

- **<u>High Confidence:</u>** The rule must be triggered. It is highly likely for someone to buy Y if they buy X. **The confidenceis the fraction of transactions where constumer buys X and Y:**

  Probabilistic meaning: conditional probability of Y given X: $\operatorname{conf}(X \to Y) = \frac{\operatorname{supp}(X \cup Y)}{\operatorname{supp}(X)}$

So to use an **association mining rule**, it should have **higher support** than a given value and **higher confidence** than another value.

21.1.2 APRIORI ALGORITHM

It's a way to discover association rules with high support. This is done by comparing the support of subsets of items to a *support* (threshold) *value* ϵ.

When this condition is fulfilled we say its is ***frequent***.

- *Downwards Closure Property:* If an itemset is frequent, so is any of its subsets. If a transaction that contains a subset X and it has highsupport, then any of thesubsets composed by elements of X is probable aswell.

By this same Upwards Closure Property, we can then discard any subset of observations in case one of the subsets which is contained in the itemset is ***unfrequent.***

The algorithm: *It finds (iteratively) frequent set of 1 item and scales up until there are no longer frequent itemsets:*

Algorithm 9: Apriori algorithm

Given N transactions and let $\epsilon > 0$ be the minimum support count
$L_1 = \{\{j\} | \text{supp}(\{j\}) \geq \epsilon\}$
for $k = 2, \ldots, M$ and $L_k \neq \emptyset$ **do**
  $C'_k = \{s \cup \{j\} | s \in L_{k-1}, j \notin s\}$
  Set $C_k = C'_k$
  for each $c \in C'_k$ **do**
    for each $s \subset c$ such that $|s| = k - 1$ **do**
      if s is not frequent, i.e. $s \notin L_{k-1}$ **then**
        $C_k = C_k \setminus \{c\}$ (Remove c from C_k)
      end if
    end for
  end for
  $L_k = \{c | c \in C_k, \text{supp}(c) \geq \epsilon\}$ (compute support)
end for
$L_1 \cup L_2 \cup \cdots \cup L_k$ are then all frequent itemsets

In every iteration:

- Generate candidate itemsets C_K, by adding all items to the last iteration frequent itemsets L_k.

$$C'_k = \{s \cup \{j\} | s \in L_{k-1}, j \notin s\}$$

- From the candidate list C_K, we remove those infrequent subsets (of size L_{k-1} which have been created when introducing the new item in the last iteration frequent itemsets)

$$\{s | s \subset c, |s| = k - 1\} \cap L_{k-1} = \emptyset \quad \text{->} \quad C_k = C'_k \setminus \{c | \{s | s \subset c, |s| = k - 1\} \cap L_{k-1} = \emptyset\}$$

- Compute the candidates which are considered frequent (support higher than support value E) which will be the new frequent itemsets L_k.

Example: *Apriori Algorithm*

milk	butter	beer	diapers
1	0	1	1
0	1	0	1
0	1	1	1
0	0	1	0
1	0	1	1

Initialization L1:

If the support value is ϵ = 0.15, then the items in every transaction with support higher than that will be considered frequent. Since there are 5 transactions, to consider an item frequent it should be purchased at least one time (1/5=0.2> ϵ).

Since all items occur at least once, the *frequent vector* L_1 should contain all the items in the dataset X (M).

$$L_1 = \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & 1 \end{bmatrix}$$

First Iteration K=2:

Now we will create the *candidates vector* C_2', out of the frequency vector from the last iteration L_1, and we add all items for all the components (but the one evauated).

$$C_2' = \begin{bmatrix} 1 & 1 & \cdot & \cdot \\ 1 & \cdot & 1 & \cdot \\ 1 & \cdot & \cdot & 1 \\ \cdot & 1 & 1 & \cdot \\ \cdot & 1 & \cdot & 1 \\ \cdot & \cdot & 1 & 1 \end{bmatrix}$$

For every component of the prime candidate vector, we have to obtain all the subgroups (of size k-1) in each row and verify if the subgroup was in the previous iteration. In such case, it will be considered for Candiate Vector C_2.

$$C_2' = \begin{bmatrix} 1 & 1 & \cdot & \cdot \\ 1 & \cdot & 1 & \cdot \\ 1 & \cdot & \cdot & 1 \\ \cdot & 1 & 1 & \cdot \\ \cdot & 1 & \cdot & 1 \\ \cdot & \cdot & 1 & 1 \end{bmatrix}$$ In this specific case, $C_2=C_2'$

Once this is done for every row and for every (k-1 element combination), the support of every row in the Candidate Vector is computed (Number of times the given combination of items is purchased). If the supoport value is met, the given row will finally be included in L_2.

$$L_2 = \begin{bmatrix} 1 & \cdot & 1 & \cdot \\ 1 & \cdot & \cdot & 1 \\ \cdot & 1 & 1 & \cdot \\ \cdot & 1 & \cdot & 1 \\ \cdot & \cdot & 1 & 1 \end{bmatrix}$$

<u>Second Iteration K=3:</u>

First step is to add all items to L_2, to obtain Prime Candidate Vector C3'. After this, we check for every (k-1) subsets if they were present in L2 to obatin C3.

$$C_3' = \begin{bmatrix} 1 & 1 & 1 & \cdot \\ 1 & \cdot & 1 & 1 \\ 1 & 1 & \cdot & 1 \\ \cdot & 1 & 1 & 1 \end{bmatrix} \qquad C_3 = \begin{bmatrix} 1 & \cdot & 1 & 1 \\ \cdot & 1 & 1 & 1 \end{bmatrix}$$

Note that only the subsets where all the possible (k-1) combinations are in Lk-1 are met, is introduced in C3.

To conclude the iteration, verify the support meets the support value condition:

$$L_3 = \begin{bmatrix} 1 & \cdot & 1 & 1 \\ \cdot & 1 & 1 & 1 \end{bmatrix}$$

<u>Third Iteration K=4:</u>

Again, consists in taking L3 and adding all items to all subsets, to generate the Prime Candidate Vector.

$$C_4' = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

Which is not met, then C4=0 and the algorithm is finished:

$$L = \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & 1 \\ 1 & \cdot & 1 & \cdot \\ 1 & \cdot & \cdot & 1 \\ \cdot & 1 & 1 & \cdot \\ \cdot & 1 & \cdot & 1 \\ \cdot & \cdot & 1 & 1 \\ 1 & \cdot & 1 & 1 \\ \cdot & 1 & 1 & 1 \end{bmatrix}$$

21.3 Apriori algorithm to find itemsets with High Confidence

Apriori Algorithm finds ASSOCIATION RULES (X->Y) itemsets with high support (appear in the dataset with a given minimum frequency), but we also want them to have a minimum level of conficende (Often, when X happens Y as well).

$\text{supp}(X \to Y) = \text{supp}(X \cup Y) \geq \epsilon$ but also $\text{conf}(X \to Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)} \geq \delta$

Hence we obtain that : $\frac{1}{\delta}\text{supp}(X \cup Y) \geq \text{supp}(X)$

Strategy:

1) Finds all itemsets **Z** with support higher than ϵ.
2) Finds subsets **X** of **Z** that meet the confidence limitation. $\text{supp}(X) \leq \frac{1}{\delta}\text{supp}(Z)$
3) Defining **Y** as the subgroup of **Z** that doesn't meet the confidence limitations. $Y = Z \setminus X$

*TWEAK: If W is contained on X (i.e is a subset of the itemset), then supp(w)>= supp(x). Then if X meets the confidence condition $\text{supp}(X) \leq \frac{1}{\delta}\text{supp}(Z)$ (CONFIDENCE LOWER THAN MINIMA). Then its not necessary to check smaller subsets of X, they will have **low confidence** as well. Then to find this subsets, by taking one element from the support condition susbset $X = Z \setminus \{i\}$ and only keep removing if the Association Rule has confidence greater than the confidence threshold.

21.4 PROBLEMS WITH APRIORI

The algorithm is efficient at finding frequent joint probabilities, but the algorithm only computes the confidence of those subsets denoted by having a high support (frequent), not taking on account lower support but high confidence sets.