Soft Clustering

Clustering has been one of the most popular tools in data mining. Even though hard clustering has been popularly studied and traditionally used, there are several important applications where soft partitioning is essential. Some of these applications include *text mining* and *social networks*. Soft partitioning is concerned with assigning a document in text mining or a person in a social network to more than one cluster. For example, the same document may belong to both *sports* and *politics*. In several cases softness has to be appropriately interpreted to make a hard decision; such a process permits us to delay the decision making. So, one of the major characteristics of softness is in delaying decision making as far as possible. Another notion is to acknowledge the assignment of a pattern to more than one category.

We can work out the number of soft partitions of n patterns into K clusters as follows:

- Let $X_1,\ X_2,\ \cdots, X_n$ be the n patterns and $C_1,\ C_2,\ \cdots, C_K$ be the K clusters. In soft clustering, a pattern may belong to one or more clusters. So, if pattern X_i belongs to cluster C_j then we record a 1 as the ij^{th} element of Cluster Indicator Matrix (CIM) otherwise we store a 0.

- Note that the number of 1s in the i^{th} row of CIM can vary between 1 to K. This can be chosen in $2^K - 1$ ways by ruling out the all zero vector because X_i has to be assigned to at least one of the K clusters.

- So for each row in CIM we have $2^K - 1$ choices; for all the n rows the number of possibilities is $(2^K - 1)^n$ which is of $O(2^{Kn})$. This is an upper bound because no column in CIM can be empty.

- Instead of storing a 1 or a 0 if we store one of P possible values to indicate the extent to which X_i belongs to C_j, then the number of possibilities is bounded by $(P^K - 1)^n$ as the number of distinct values each entry in CIM can assume is P. In some of the soft clustering algorithms the value of P could be very large; theoretically it could be infinite.

1 Soft Clustering Paradigms

There are a variety of soft clustering paradigms. Each of them has a distinct flavor. Some of the important ones are:

1. **Fuzzy Clustering:** Fuzzy clustering and related possibilistic clustering is the most popularly established and studied soft clustering paradigm. It is so popular that routinely soft clustering is interpreted as fuzzy clustering; again fuzzy c-Means clustering algorithm is the most frequently considered algorithm here. Each pattern X_i may be assigned to a cluster C_j with a membership value μ_{ij} which indicates the degree to which X_i belongs to C_j. It is assumed that $\mu_{ij} \in [0, 1]$ and $\sum_{j=1}^{K} \mu_{ij} = 1$.

2. **Rough Clustering:** When there is uncertainty in assigning patterns to clusters which means it is difficult to specify clusters as sets, then we can use *rough sets* to characterize the uncertainty. There could be patterns that clearly belong to only one cluster; other patterns may belong to two or more clusters. This is abstracted by a rough set which is represented using two sets; these sets are called *lower approximation* and *upper approximation*. A pattern can belong to at most one lower approximation. If a pattern does not belong to any lower approximation then it belongs to two or more upper approximations.

3. **Evolutionary Algorithms for Clustering:** Evolutionary approaches including *genetic algorithms*, *evolutionary schemes*, and *evolutionary programming* are three popular tools used in both hard and soft clustering. Unlike the other approaches, here a collection of possible solutions (*population*) is simultaneously processed leading to a more effective scheme for pursuing important solution directions. Genetic algorithms have been shown to lead to optimal hard clustering based on squared error criterion under some acceptable conditions.

4. **Statistical Clustering:** Here each pattern is viewed as a vector in the multi-dimensional space. So clusters are viewed as regions of points in the space. Hence this approach is also based on *geometric* or *vector space* models. Typically geometric models are popular in *computational geometry*, *graphics and visualization*. Similarly vector space model is the most popular in *information retrieval*. Because regions can overlap

or vectors can be associated with multiple regions, statistical models can be potentially soft.

Popular hard clustering algorithms like the K-Means algorithm can be modified to be soft. There are several soft versions of the K-Means algorithm which will be discussed in later sections. Leader algorithm is one of the most efficient hard clustering algorithms and it can be modified to perform soft clustering. Because of the similarity between leader and BIRCH it is possible to have a soft version BIRCH also.

5. **Neural Networks for Clustering:** Artificial Neural Networks (ANNs) have been popularly used in clustering; Self Organizing Map (SOM) is the most popular of them. Typically ANNs employ sigmoid type of activation functions which return positive real values in the range 0 to 1. So, based on the output values returned ANNs also can be used for soft clustering. ANNs may be viewed as statistical clustering tools. A major restriction of the ANNs is that they can deal with only numerical vectors. It is possible to view support vector machine (SVM) as the most popular and state-of-the-art artificial neural network.

6. **Probabilistic Clustering:** Conventionally vector space models and probabilistic models are treated as different in the area of information retrieval. Further, probabilistic models are typically viewed as *generative models*; once we have the model we can generate patterns of the corresponding cluster. In fact probabilistic models may be viewed as theoretical and their implementations are based on statistics. For example the Bayesian models are inherently probabilistic and without knowing the prior probabilities and likelihood values it is not possible to use them in practice; here statistics provides appropriate implementable approximations. This is achieved by employing estimation schemes. Probabilistic models are inherently soft. For example while using a Bayesian model in a two class problem, it is possible that none of the two posterior probabilities is zero; in such a case the pattern belongs to both the classes but to different degrees based on the posteriors.

   Expectation maximization is a well-known probabilistic clustering algorithm. It may be viewed as a probabilistic variant of the K-Means algorithm. It is the most popular soft clustering algorithm in the history

of pattern recognition and machine learning. Some of the other examples include *Probabilistic Latent Semantic Indexing (PLSI)* and *Latent Dirichlet Allocation (LDA)* which are popular in document clustering.

Among these, clustering based on *fuzzy sets*, *rough sets*, *evolutionary algorithms*, and *neural networks* are traditionally identified with the soft computing paradigm. Even though there is a significant amount of work done in these areas and reported in a variety of important journals and conference proceedings, statistical and probabilistic paradigms have gained a lot of prominence recently; they are fashionable currently. Because of the increasing interest in statistical learning theory, soft clustering based on expectation maximization and its variants is arguably the most popular now. We discuss each of the soft clustering paradigms in detail in the future sections.

2 Fuzzy Clustering

Typically each cluster is abstracted using one or more representative patterns. For example centroid and leader are popularly used representatives of a cluster. Let R_i be the set of representatives of a cluster C_i. Then a pattern X is assigned to one or more clusters based on the following:

- In hard clustering we assign X to cluster C_i if $Similarity(X, R_i) > Similarity(X, R_j)$ for all $j \neq i$ or equivalently $distance(X, R_i) < distance(X, R_j)$ for all $j \neq i$. It requires some appropriate definition of similarity and distance. In case of having a single representative for each cluster, for example the centroid of the cluster, euclidean distance between X and R_i is popularly used to characterize proximity where R_i is the centroid of the i^{th} cluster. Here the *winner-take-all* strategy is used to realize the hard clustering; this is based on assigning X to only one cluster (winner cluster gets the pattern) based on minimum distance or maximum similarity between X and the cluster representative.

- An alternative to winner-take-all is to assign X to more than one cluster based on the proximity between X and each R_i. Similarity between X and R_i is used to characterize the membership of X in C_i. This is typically used in *possibilistic clustering*; the similarity values are suitably normalized so that they add up to one in the case of *fuzzy clustering*.

Both hard K-means and Leader clustering algorithms can be suitably modified to realize their fuzzy counterparts; among them the Fuzzy K-Means algorithm is more popular which we describe next.

2.1 Fuzzy K-Means Algorithm

It goes through an iterative scheme like the K-Means algorithm. Starting with some initial assignment of points, it goes through the following steps:

1. **Obtain Membership Values:**
   $\mu_{ij} = \frac{||X_j - R_i||^{-2/(M-1)}}{\sum_{l=1}^{K}(||X_j - R_l||)^{-2/(M-1)}}$

2. **Compute the Fuzzy Cluster Centroids:**
   $Centroid_i = R_i = \frac{\sum_{j=1}^{n}(\mu_{ij})^M X_j}{\sum_{j=1}^{n}(\mu_{ij})^M}$

   Where

   - μ_{ij} = Membership value of pattern X_j in cluster C_i for $j = 1, 2, \cdots, n$ and $i = 1, 2, \cdots, K$.
   - $|| X_j - R_i ||$ typically is the Euclidean distance between X_j and R_i.
   - R_i = Centroid of the i^{th} cluster.
   - M is the fuzzifying constant. If $M = 1$ then the Fuzzy algorithm works like the hard K-Means algorithm. This is because $\mu_{ij} = 1$ when X_j is assigned to C_i; note that in the expression for μ_{ij}, the exponent of the distance between X_j and R_l in the denominator will be tending to $-\infty$ for each l. So, the smallest of them will be selected which is equal to the numerator $(|| X_j - R_i ||^{-2/(M-1)})$ and hence $\mu_{ij} = 1$. If the value of M is large or tends to ∞ then $\mu_{ij} = \frac{1}{K}$ as the exponents in both the numerator and denominator tend to 0.

3. The above two terms are updated iteratively by the Fuzzy K-Means algorithm; assign each pattern to its nearest fuzzy centroid and update the μ_{ij} and R_i values. The algorithm stops when there is no significant change in these values computed over two successive iterations.

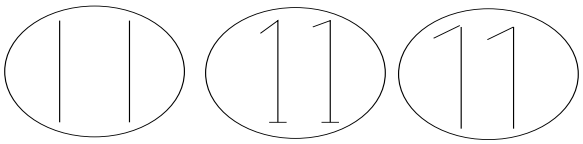
It is possible to show that the Fuzzy K-Means algorithm minimizes a variant of the squared error given by

$$\sum_{i=1}^{K}\sum_{i=1}^{n}(\mu_{ij})^M \mid\mid X_j - R_i \mid\mid^2$$

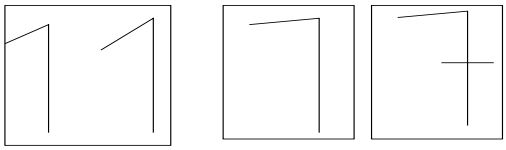
Both the hard and fuzzy K-Means algorithms can be guaranteed to reach the locally optimal solution of their respective objective functions. A similar form of softness can be extended to realize fuzzy versions of other hard clustering algorithms.

3 Rough Clustering

A rough set is characterized using the notion of the *indiscernability relation* that is defined based on equivalence classes of objects. Here, some patterns are known to definitely belong to a cluster. For example consider the patterns shown in Figure 1. There are two classes of character patterns in the figure;



(a) A Collection of Ones



(b) A Collection of Sevens

Figure 1: Examples of patterns from Two Classes

some are of character 1 (shown in Figure 1 (a)) and others are of character 7 (shown in Figure 1 (b)). There are four patterns that definitely belong to 1 and 4 patterns that definitely belong to 7. In the case of ones there are three equivalence classes bounded by circles; characters in each circle are similar to each other. Elements in the first two equivalence classes (considered from left to right) shown in Figure 1 (a) definitely belong to class of 1s. Similarly in

Figure 1 (b) there are 3 equivalence classes of sevens and these are bounded by rectangles in the figure; all of them definitely belong to class of 7s. However there is one equivalence class (third) in Figure 1 (a) where the patterns are either ones or sevens; these two patterns are possibly ones or sevens.

Patterns in these equivalence classes are *indiscernible*; note that 1s class consists of union of the two indiscernible equivalence classes and similarly all the three equivalence classes in Figure 1 (b) are also indiscernible and their union is a subset of class of 7s. The remaining equivalence class which has patterns that are possibly ones or sevens can be subset of both the classes. In a formal setting *indiscernability* is defined as follows:

Two patterns X_i and X_j are related or *indiscernible* if $X_{il} = X_{jl}$ for all l in a subset of features where X_{il} is the l^{th} feature value of X_i.

It is possible to show that such a relation is an *equivalence relation* and it partitions the set of patterns $\mathcal{X}$ into equivalence classes. In this set up two patterns in any equivalence class are *identical* on a chosen subset of features. It is possible to relax by insisting that the two patterns have values *falling in an interval*, for a feature, than the values being identical. In this setting the notion of similarity based on overlapping intervals instead of values could be exploited to form the equivalence classes.

Once we have the equivalence classes based on similarity on the set of features F we can define rough set using the *lower approximation* ($\underline{F}S$) and the *upper approximation* ($\overline{F}S$) where $S \subseteq \mathcal{X}$ as follows:

$$\underline{F}S \;=\; \cup_i E_i,\ where\ E_i \subseteq S \tag{1}$$

$$\overline{F}S \;=\; \cup_i E_i,\ where\ E_i \cap S \neq \phi \tag{2}$$

Note that in the lower approximation we insist that each equivalence be completely (or definitely) in S whereas in the upper approximation we consider equivalence classes that partially overlap with S. Hence the lower approximation contains patterns that definitely belong to S and the upper approximation has patterns that may possibly belong to S. Additionally we have:

- *Boundary Region of* S is given by $\overline{F}S - \underline{F}S$. If the boundary region has no elements then there is no roughness in S with respect to F.
- Lower approximation may be viewed as the core part of a cluster. Lower approximations of two clusters do not overlap. So, if a pattern X

(definitely) belongs to the lower approximation of a rough cluster then it will not be present in any other cluster.

- A lower approximation of a cluster is a subset of its upper approximation.
- If a pattern X belongs to the boundary region of a cluster, then it belongs to at least one more upper approximations.

We use these properties in designing rough clustering algorithms; a popular algorithm for rough clustering is the Rough K-Means algorithm which we discuss in the next subsection.

3.1 Rough K-Means Algorithm

Rough K-Means algorithm is the most popular among the rough clustering algorithms. Basically we have to characterize assignment of a pattern to a rough cluster and computation of the centroid of a rough cluster. We discuss these steps below:

1. **Assigning a Pattern:** A pattern X is assigned to one or more clusters. This may be done as follows:
   (a) **Initialize**: Let $C_1, C_2, \cdots, C_K$ be the K clusters with centroids $R_1, R_2, \cdots, R_K$ respectively. These centroids are obtained based on some initial assignment of patterns to clusters.
   (b) **Obtain the Nearest Centroid**: Obtain the nearest centroid R_j of pattern X based on
   $$d(X, R_j) = \min_{1 \leq i \leq K} d(X, R_i).$$
   (c) **Obtain Other Nearer Centroids**: Choose an ϵ and obtain all the clusters i ($\neq j$) such that $d(X, R_i) \leq (1 + \epsilon)d(X, R_j)$.
   (d) **Assign to Multiple Clusters**: If there are q ($2 \leq q \leq K - 1$) such i's given by $i_1, i_2, \cdots, i_q$, then assign X to $\overline{F}C_{i_1}, \overline{F}C_{i_2}, \cdots, \overline{F}C_{i_q}$.
   (e) **Assign to a Single Cluster j**: If there is no such i then assign X to the lower approximation of the rough cluster C_j, that is $\underline{F}C_j$. Note that by the property of rough sets, X is also assigned automatically to $\overline{F}C_j$.

2. **Updating Centroids:** Unlike the conventional K-Means algorithm, here centroids are updated by assigning different weightages to patterns based on whether they are in a lower approximation or in multiple upper approximations. Specifically centroids are updated as follows:

   (a) If $\underline{F}C_j = \overline{F}C_j$ and $\underline{F}C_j \neq \phi$ then its centroid R_j is $R_j = \frac{\sum_{X \in \underline{F}C_j} X}{|\underline{F}C_j|}$.

   (b) If $\underline{F}C_j = \phi$ and $\overline{F}C_j \neq \phi$ then $R_j = \frac{\sum_{X \in (\overline{F}C_j)} X}{|\overline{F}C_j|}$.

   (c) If $\underline{F}C_j \neq \phi$ and $\overline{F}C_j \neq \underline{F}C_j$ then $R_j = w_l \frac{\sum_{X \in \underline{F}C_j} X}{|\underline{F}C_j|} + w_u \frac{\sum_{X \in (\overline{F}C_j - \underline{F}C_j)} X}{|\overline{F}C_j - \underline{F}C_j|}$
   Here w_l and w_u correspond to the weightages or importance associated with the lower and upper approximations respectively. They add upto 1; that is $w_l + w_u = 1$.

The above two steps are iteratively used to realize the Rough K-Means algorithm.

4 Clustering Based on Evolutionary Algorithms

Evolutionary algorithms are inspired by some of the important properties of natural evolution; *survival of the fittest* and *exchange of genetic material* using crossover or recombination are two such functions captured by evolutionary operators. An evolutionary algorithm typically has the following steps:

1. **Initialization:** A collection of possible solutions, called *population*, is chosen.

2. **Evaluation:** The *fitness* of each element of the population is computed based on a fitness function.

3. **Selection:** A pair of solutions is selected at a time based on the fitness value assigned to each of the possible solutions in step 2. It is common to view selection as an *exploitation operator* where domain-knowledge based fitness values are exploited to select fit individuals for further processing. Recombination and mutation are helpful in exploring the

search space and are described next. Because of this judicial combination of exploitation and exploration, evolutionary algorithms may be considered as focused (exploitation) random-search (exploration) algorithms.

4. **Recombination:** Using the recombination operation on the two selected solutions, two offspring/children are generated.

5. **Mutation:** The two children generated using recombination are modified based on randomly chosen values in the string and mutating these values. The resulting solutions are placed in the next population. The process of selecting a pair of strings, recombination and mutation are repeated till the required number of solutions to fill the next population is generated.

6. **Termination:** Steps 2 to 5 are repeated till some termination condition is satisfied. Going through steps 2 to 5 once generates from the population at the current instance ($\mathcal{P}_t$) a new population corresponding to the next time instance ($\mathcal{P}_{t+1}$). Such a process of generating $\mathcal{P}_{t+1}$ from $\mathcal{P}_t$ is called a *generation*. In a practical setting the evolutionary algorithm terminates after a specified number of generations.

We briefly explain below how various steps listed above are realized in clustering a collection of n patterns. For the sake of illustration we consider a two-dimensional dataset: $\mathcal{X} = A : (1,1)^t;\ B : (1,2)^t;\ C : (2,2)^t;\ D : (6,2)^t;\ E : (7,2)^t;\ F : (6,6)^t;\ G : (7,6)^t$.

1. **Initialization:**

   - Let N be the size of a population; N is typically chosen to be even. The size of the population is chosen based on the size of the solution space and computational resources.
   - The N initial solutions are typically randomly chosen; it is the right thing to do in the absence of any other information. If additional knowledge about the problem is available, then it can be used to choose the initial solutions.
   - Each solution corresponds to a partition of the dataset and may be viewed as a string. Some of the possibilities are:

(a) In the case of genetic algorithms (GAs), each solution string is conventionally a binary string. We can represent the partition: $C_1 = \{A, B, C\}$; $C_2 = \{D, E\}$; $C_3 = \{F, G\}$ using the binary string
1 1 1 0 0 0 0 ; 0 0 0 1 1 0 0; 0 0 0 0 0 1 1
which has 3 substrings where each cluster is represented by a substring of size 7; if a pattern is in a cluster then we place a 1 in that location, otherwise a 0. There will be K substrings if there are K clusters and each substring is of length n. This view permits soft clustering. For example the binary string
1 1 1 0 0 0 0 ; 1 0 0 1 1 0 0; 0 0 0 0 1 1 1
indicates overlapping clusters where A belongs to both C_1 and C_2 and E is in both C_2 and C_3.

(b) Another popular representation employs string-of-group-numbers. Each character in the string represents a cluster (or group) number. For example:
1 1 1 2 2 3 3 represents a hard partition which indicates that $A, B,$ *and* C belong to cluster 1; D and E belong to cluster 2; patterns $F,\ G$ are in cluster 3. This characterizes the hard partition $C_1 = \{A, B, C\}; C_2 = \{D, E\}; C_3 = \{F, G\}$. This representation does not permit soft clustering using cluster numbers 1 to K.

(c) String-of-centroids representation employs a real vector to represent each location in the string. Specifically each string represents K cluster centers; each cluster center is a string of size d where d is the dimensionality of the vectors. For example consider the representation:
1.33, 1.66; 6.5, 2.0; 6.5, 6.0
which consists of 3 centroids each of them in a two-dimensional space which corresponds to the partition: $C_1 = \{A, B, C\}; C_2 = \{D, E\}; C_3 = \{F, G\}$ because A, B and C are closer to the centroid $(1.33, 1.66)^t$; D and E are nearer to the centroid $(6.5, 2.0)^t$; and F and G are closer to the centroid $(6.5, 6.0)^t$. Note that the string-of-centroids representation corresponds to the string-of-group-numbers representation given by
1 1 1 2 2 3 3; in fact there is a correspondence between these two representations in general. There is a bijection between

sets of equivalence classes of the two types defined as follows:

- Two string-of-group-numbers S_{g_i} and S_{g_j} are related by a binary relation R_g if they both correspond to the same partition of $\mathcal{X}$. For example consider the strings 1 1 1 2 2 3 3 and 2 2 2 1 1 3 3; both of them lead to the same partition of $\mathcal{X}$ given by $\{\{X_1, X_2, X_3\}, \{X_4, X_5\}, \{X_6, X_7\}\}$. Note that R_g is an equivalence relation on the set of strings-of-group-numbers representations; R_g partitions the set into equivalence classes where all the strings in each equivalence class are related. Let EC_{R_g} be the set of equivalence classes.
- Two strings-of-centroids S_{c_i} and S_{c_j} are related by a binary relation R_c if they both correspond to the same partition of $\mathcal{X}$. For example, the strings-of-centroids $(1,1);\ (1.5,2);\ (6.5,4)$ and $(1,1);\ (1.5,2);\ (6,4)$ correspond to the same partition of $\mathcal{X}$ given by $\{\{(1,1)\}, \{(1,2),(2,2)\}, \{(6,2),(7,2),(6,6),(7,6)\}$. Again note that R_c is an equivalence relation on the set of strings-of-centroids and partitions it into equivalence classes. Let EC_{R_c} be the set of these equivalence classes.
- There is a bijection between EC_{R_g} and EC_{R_c}. Associated with each partition of $\mathcal{X}$ there is an element of EC_{R_g} and an element of EC_{R_c} which map to each other by the bijection.

2. **Fitness Computation:** It is essential to associate a fitness value with each solution string; this association could be based on one or more criterion functions. This is required to select highly fit strings for further exploration. Let us consider for each type of representation how fitness value is computed.

   (a) Because there is a correspondence between string-of-group-numbers and string-of-centroids we consider them together. Consider the partition given by
   string-of-group-numbers: 1 1 1 3 3 2 2 and equivalently
   string-of-centroids: 1.33,1.66; 6.5,2.0; 6.5,6.0
   it is the optimal partition and the with-in-group-error-sum-of-squares value is
   1.33 + 0.5 + 0.5 = 2.33, where the left hand side indicates the

contribution of each cluster.

Next consider the partition given by
string-of-group-numbers: 1 2 2 3 3 3 3 and equivalently
string-of-centroids: 1.0,1.0; 1.5,2.0; 6.5,4.0.
Here, the squared error value given by the sum over the three clusters is
$0 + 0.5 + 17.0 = 17.5$.

(b) Consider the binary representation for a hard partition given by
1 1 1 0 0 0 0 ; 0 0 0 1 1 0 0; 0 0 0 0 0 1 1
In this case also the squared error is given by 2.33 which corresponds to the optimal partition. It is possible to consider the soft partition given by
1 1 1 0 0 0 0 ; 1 0 0 1 1 0 0; 0 0 0 0 1 1 1
In this case there are some patterns that belong to more than one cluster. In such a situation we can compute a modified version of the squared error. This modification is achieved by adding weighted contributions from different patterns in computing the squared error. The weight for a pattern is 1 if it belongs to a single cluster. If a pattern belongs to more than one cluster, then a suitable weight is assigned to a pattern for each cluster based on its distance to centroid of each cluster to which the pattern belongs.

For example consider the soft partition given by
1 1 1 0 0 0 0 ; 1 0 0 1 1 0 0; 0 0 0 0 1 1 1
where $A \in C_1$ *and* C_2 and $E \in C_2$ *and* C_3. By noting that the cluster centers are: (1.33,1.66); (4.66,1.66); (6.66,4.66) we can compute the distances between A $(1,1)^t$ and the centroids of C_1 and C_2; the squared distances are $\frac{5}{9}$ and $\frac{125}{9}$ which can be used to assign the weights as $\frac{125}{130}$ and $\frac{5}{130}$. In a similar manner one can compute the weights for E $(7,2)^t$ based on the ratio of squared distances to centroids of C_2 and C_3 which will be $\frac{50}{9}$ and $\frac{65}{9}$. So, the weights are $\frac{50}{115}$ and $\frac{65}{115}$. Using these weights the squared error computed is $1.3 + 5.6 + 8.5 = 15.4$. We have suggested a simple scheme here; a modified version of this scheme is used in characterizing the fuzzy criterion function.

(c) Multi-Objective Optimization Problem (MOOP): In several real-world problems the optimization problem might involve more than

one criterion function to be optimized. For example, minimizing squared error without any constraint on the number of clusters K will mean that the minimum squared error is zero when $K = n$; in this case each pattern is a cluster. However, in real-world applications, we want a much smaller K. So, we need to minimize both the squared error and the number of clusters. In addition to the objective function, we may have a set of constraints. The two directions for dealing with MOOP are:

i. Take a weighted combination of the multiple criterion functions and treat the resulting function as a single-objective criterion function.

ii. Treat the multi-objective criterion as a vector of individual objectives. Here we consider a feasible solution dominating another based on how the two solutions are related on each criterion function value; based on the non-dominated set of solutions we have pareto-optimal set and we consider such solutions.

3. **Selection:** Selection is carried out by using the *survival of the fittest strategy*. We have to select a solution so that the squared error is minimized. However, it is convenient to implement selection based on maximizing some criterion function. So, a simple mapping that we implement is to take the reciprocal of the squared error value and maximize it assuming that the squared error is not 0 for any solution string. Let us illustrate using the two-dimensional example considered in computing the squared error values. We use the string-of-centroids representation. We give in Table 1 a string-of-centroids; the squared error; its reciprocal; and the probability of selection proportional to the reciprocal. Note that selection probability shown in the last column is obtained by normalizing the reciprocal of the fitness value. For example, for string 1 the selection probability is $\frac{0.43}{(0.43+0.057+0.046)} = 0.807$. The other probability values are obtained similarly. Assuming for a while that the population has three strings shown in table 1 we generate a uniformly distributed random number r in the range 0 to 1. We select the first string if $r \leq 0.807$; the second string if $0.807 < r \leq 0.914(0.807+0.107)$; a copy of the third string is selected if $0.914 < r \leq 1$. Note that the first string has very high probability of selection. Selection is the popularly used term in GAs; reproduction is the term used in other evolutionary

string number	centroid1	centroid2	centroid3	squared error	fitness= 1/squared error	selection probability
1	(1.33,1.66)	(6.5,2.0)	(6.5,6.0)	2.33	0.43	0.807
2	(1.00,1.00)	(1.5,2.0)	(6.5,4.0)	17.5	0.057	0.107
3	(1.00,1.00)	(1.0,2.0)	(5.6,3.6)	21.88	0.046	0.086

Table 1: Three Different Solution Strings

algorithms.

4. **Crossover:** Crossover is a binary operator; it takes two strings as input and combines them to output two children strings. This may be illustrated using the data in table 2. By considering two parent strings that are not highly fit crossover results a child (child 1) that is highly fit (squared error = 3.5).

string	centroid1	centroid2	centroid3	squared error	fitness= 1/squared error
parent 1	*(1.0,1.0)*	*(6.5,2.0)*	*(1.5,2.0)*	33.5	0.03
parent 2	(1.0,1.0)	(1.0,2.0)	(6.0,6.0)	34	0.03
child 1	*(1.0,1.0)*	*(6.5,2.0)*	(6.0,6.0)	3.5	0.29
child 2	(1.0,1.0)	(1.0,2.0)	*(1.5,2.0)*	134	0.007

Table 2: Crossover on Two Strings

- Here we have used a single-point crossover operator where the genetic material is exchanged across the crossover point.
- The crossover point is randomly selected. In the dataset shown in Table 2 the crossover point is selected between centroid2 and centroid3.
- Based on this crossover point the prefix (centroid1 and centroid2) of parent 1 is combined with the suffix (centroid3) of parent 2 to generate child 1. Similarly prefix of parent 2 and suffix of parent 1 are combined to form child 2.

- Note that in the string-of-centroids representation considered here the crossover point can be selected between two successive centroids.
- Typically crossover is the popularly used term in genetic algorithms whereas *recombination* is used in Evolutionary search.

5. **Mutation:** Mutation is a unary operator; it takes a string as input and outputs a string by mutating some randomly selected values. If the value at position i is v_i and $\delta \in [0, 1]$. After mutation v_i becomes $v_i + 2\delta v_i$ or $v_i - 2\delta v_i$. We show how it works using the input and output strings shown in Table 3.

string	centroid1	centroid2	centroid3	squared error	fitness= 1/squared error
input	(1.5,1.5)	(6.5,3.0)	(6.5,6.0)	4.25	$\frac{4}{17}$
output	(1.5,1.5)	(6.5,2.0)	(6.5,6.0)	2.25	$\frac{4}{9}$

Table 3: A String Before and After Mutation

- By mutating the value 3.0 to 2.0 in centroid2 the reciprocal of the squared error almost doubles.
- Mutation working on a binary string randomly selects a bit position and replaces 0 by 1 and 1 by 0.
- It is possible to show that selection (or reproduction) and mutation are adequate for simulating evolution.
- *Evolutionary Programming* based algorithms use only reproduction and mutation; they do not use recombination.

6. **Steady-State Genetic Algorithm (SSGA):** Typically crossover and mutation are performed with probabilities P_c and P_μ respectively; further larger P_c values are common, but the value of P_μ is typically very small. Increasing the mutation rate P_μ can lead to random search. It is possible to avoid random search and still increase the value of P_μ by copying some highly fit (*elitist*) strings to the next population. This is exploited by SSGA. It can help us in exploring the search space better

by using larger values for P_μ, however it requires a longer time to converge based on generation gap. *Generation gap* between two successive populations $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$ is characterized by the percentage of elitist strings copied from $\mathcal{P}_t$ to $\mathcal{P}_{t+1}$.

It is possible to show that using an elitist strategy the GA converges to the globally optimal solution of the criterion function under some acceptable conditions.

5 Clustering based on Neural Networks

Some of the important properties of neural networks are *competitive learning* and *self-organization*. Competitive learning is concerned with assigning an input pattern to one or more of the competing output neurons. Typically it is implemented using a two-layer neural network where each input node is connected to all the output nodes. Competitive learning is abstracted with the help of minimization of the error function given by:

$$\sum_{i=1}^{n}\sum_{j=1}^{K}\mu_{ij} \mid\mid X_i - R_j \mid\mid^2$$

where μ_{ij} is the membership of X_i in C_j and R_j is the prototype from cluster j that is closer to X_i. Self organization is achieved by using lateral inhibition. The most influential network is the *self-organizing map (SOM)* where the output layer is typically two-dimensional. Some of the important features of SOM are:

- It can be viewed as a feature extractor; it can be viewed as representing a high-dimensional input vector as a member of a cluster present in the two-dimensional output layer.
- It is a topological map where transitions in the input patterns are captured in the output layer with topological properties preserved. For example, if we look at transitions from character 1 to character 7 as shown in Figure 1 it is captured in the output layer of SOM so that these transitions are captured topologically.
- It employs a strategy called *winner-take-most (WTM)* which is a soft version of *winner-take-all* strategy. The soft version is implemented by

assigning a pattern X_i to more than one node in the output layer by using appropriate values of μ_{ij}. A neighborhood function is used to achieve it.

- Training a SOM works as follows:

  1. If the patterns are d-dimensional, then the input layer has d nodes. Each of the input nodes is connected initially using some random weights to all the nodes in the output layer. The number of nodes in the output layer is chosen based on the resolution at which the clustering is required. So, each output node is viewed as a d-dimensional vector based on the weights with which the d input nodes are connected to it.
  2. For each pattern X_i $(i = 1, 2, \cdots, n)$, the nearest output node is selected based on the euclidean distance between the pattern and the node which are both d-dimensional vectors. The winner node and its neighbors are updated by

     $$R_l(k+1) = R_l(k) + \eta(k) g_{lm}(k)(X_i - R_l(k))$$

     where the nearest node to X_i is m. $\eta(k)$ is a monotonically decreasing learning rate. In addition to the winning node m, the other nodes (l) in its vicinity are also updated. But the extent to which a neighbor is updated is based on its distance to m and is reflected in g_{lm}. One popularly used function for g_{lm} is the Mexican hat function; another is the Gaussian function given by $g_{lm}(k) = c_1 \, exp^{-\frac{||R_l - R_m||^2}{exp^{-c_2 k}}}$.

     where c_1 and c_2 are constants. Note that if we want to use winner-take-all strategy then we have to update only R_m.
  3. It is possible to show that because of the WTM strategy the learning is not trapped by local minima easily and also based on the choice of values of η and other constants like c_1 and c_2 in the update equation it is possible to show some kind of probabilistic convergence to the global optimum.

6 Statistical Clustering

K-means algorithm is a well-known hard partitional clustering algorithm where we use the winner-take-all strategy. It is possible to have an overlapping version of the K-Means algorithm that generates a covering instead of a partition. In a covering we have the clusters $C_1, C_2, \cdots, C_K$ of the given dataset $\mathcal{X}$ satisfying the following:

$$\cup_{i=1}^{K} C_i \ = \ \mathcal{X} \ and \ C_i \ \neq \ \phi \ \forall i$$
$$\forall X_i \in \mathcal{X} \ \exists C_j \ s.t. \ X_i \in C_j.$$

The difference between a covering and a partition is that $C_i \cap C_j = \phi$ for $i \neq j$ in a partition whereas the intersection need not be empty in a covering. Next we describe the Overlapping K-Means (OKM) algorithm with the help of a two-dimensional data set.

6.1 Overlapping K-Means (OKM) Algorithm

1. Let the dataset $\mathcal{X}$ be $\{(1,1),(1,2),(2,2),(6,1),(6,3),(8,1),(8,3)\}$. Choose $K(=3)$ clusters $C_1, C_2, \ and \ C_3$ with the initial centers randomly chosen from $\mathcal{X}$, say $M_1^{(0)} \ = \ (1,2), \ M_2^{(0)} \ = \ (6,1), \ M_3^{(0)} \ = \ (8,3)$.

2. For each $X_i \in \mathcal{X}$ obtain L_i, the list of centroids of clusters to which X_i is assigned. L_i is obtained as follows:

   (a) Compute the distance between X_i and each of the K centroids and rank them in increasing order of distance. For example the euclidean distances between $(1,1)$ and the centroids are 1, 5, $\sqrt{53}$.

   (b) Assign the nearest centroid to L_i. So, in the example $L_1 =$ ((1,2).

   (c) Keep on including the next centroid in L_i, based on the rank order, if the error in representing X_i using the centroids in L_i is non-increasing. Let $\mid L_i \mid$ be the size of L_i; then the error is defined as

$$error(X_i, L_i) \ = \| \ X_i - \frac{\sum_{j=1}^{|L_i|} M_j}{\sum_{j=1}^{|L_i|} 1} \|$$

In the case of (1,1) the error in using (1,2) is 1. If we consider adding the next centroid, in the rank order, (6,1) then the error is $|| (1,1) - \frac{1}{2}[(1,2) + (6,1)] || = 2.55$ which is larger than 1; so, the error increases and we do not add (6,1) to $L1$. So, $L_1 = ((1,2))$.

(d) Similarly for the other patterns we have the lists as follows:
$(1,2) : L_2 = ((1,2))$; $(2,2) : L_3 = ((1,2))$
$(6,1) : L_4 = ((6,1))$; $(6,3) : L_5 = ((6,1),(8,3))$
$(8,1) : L_6 = ((6,1),(8,3))$; $(8,3) : L_7 = ((8,3))$.

3. Obtain the clusters based on the assignments and L_i's:
$C_1 = \{(1,1),(1,2),(2,2)\}$
$C_2 = \{(6,1),(6,3),(8,3)\}$
$C_3 = \{(6,3),(8,1),(8,3)\}$
.

4. Update the centroids using the weighted average given below:

$$M_k^* = \frac{1}{\sum\limits_{X_i \in C_k} w_i} \sum_{X_i \in C_k} w_i . C_k^i$$

where $w_i = \frac{1}{|L_i|^2}$ and
$C_k^i = \mid L_i \mid X_i - \sum_{M_j \in L_i - \{M_k\}} M_j$

5. Now the updated centroids are
$M_1^{(1)} = (\frac{4}{3}, \frac{5}{3})$, $M_2^{(1)} = (\frac{19}{3}, \frac{4}{3})$, $M_3^{(1)} = (\frac{23}{3}, \frac{8}{3})$.

6. For the other patterns we have the lists as follows:
$(1,1) : L_1 = ((\frac{4}{3}, \frac{5}{3}))$; $(1,2) : L_2 = ((\frac{4}{3}, \frac{5}{3}))$; $(2,2) : L_3 = ((\frac{4}{3}, \frac{5}{3}))$
$(6,1) : L_4 = ((\frac{19}{3}, \frac{4}{3}))$; $(6,3) : L_5 = ((\frac{19}{3}, \frac{4}{3}), (\frac{23}{3}, \frac{8}{3}))$
$(8,1) : L_6 = ((\frac{19}{3}, \frac{4}{3}), \frac{23}{3}, \frac{8}{3}))$; $(8,3) : L_7 = ((\frac{23}{3}, \frac{8}{3}))$.

7. Using the details given in step 4, the updated centroids are:
$M_1^{(1)} = (\frac{4}{3}, \frac{5}{3})$, $M_2^{(1)} = (\frac{19}{3}, \frac{4}{3})$, $M_3^{(1)} = (\frac{23}{3}, \frac{8}{3})$.

8. There is no change in the clusters and their centroids during two successive iterations. So, the algorithm terminates and the final clusters obtained are:
$\{(1,1),(1,2),(2,2)\}$; $\{(6,1),(6,3),(8,1)\}$; $\{(8,3),(6,3),(8,1)\}$. Note that (6,3) and (8,1) belong to two clusters leading to a soft partition.

9. However, initial centroid selection is important here. If the initial centroids chosen are $(1, 2)$; $(6, 1)$; $(8, 3)$ then the clusters obtained using this algorithm are:
$\{(1, 1), (1, 2), (2, 2)\}$; $\{(6, 1), (6, 3)\}$; $\{(8, 3), (8, 1)\}$ which is a hard partition.

6.2 Expectation Maximization based Clustering

K-Means algorithm is the most popular hard clustering algorithm in the history of clustering. However several important applications of current interest require soft clustering; these applications include document clustering, customer churn prediction, and social networks. Expectation maximization algorithm plays an important role in soft clustering. Each of the clusters is viewed as being generated by a hidden or latent variable.

Expectation Maximization (EM) has been used in learning Hidden Markov Models (HMMs) in the form of the well-known Baum-Welch algorithm. It is also used in learning the Probabilistic Latent Semantic Indexing (PLSI) model. In a nutshell it has revolutionized learning based on probabilistic models including Bayes belief nets and Gaussian Mixture Models (GMMs). It can be viewed as a soft clustering paradigm; more specifically it is a probabilistic version of the popular K-Means algorithm.

From a theoretical perspective it could be viewed as data likelihood maximizer. It can effectively deal with incomplete data; such an incompleteness could result because of different reasons:

1. In supervised classification we are given training data $\mathcal{X}$ of n patterns where the i^{th} pattern $X_i \in \mathcal{X}$, $i = 1, \cdots, n$, is a $(d + 1)$-dimensional vector. Typically the first d features are independent and the $(d+1)^{th}$ feature is the class label of the pattern. In such a classification context, it is possible that values of one or more features are missing. Here, EM can be used to learn the parameter values using a single or a mixture of distributions to characterize each class. This is where EM is used in GMMs and HMMs.

2. In clustering, the data given has no class labels; so, each pattern is a d-dimensional vector. Here, EM could be used to obtain the clusters. Note that in this context, the actual clusters are latent or hidden. More specifically we would like each d-dimensional pattern here to be

associated with an additional feature; this $(d+1)^{th}$ feature will have as its value one or more cluster labels resulting in a soft clustering of patterns in $\mathcal{X}$. Note that one can use a mixture distribution here also; for example one can learn a mixture of Gaussians using the EM where each cluster is represented by a Gaussian component.

We concentrate on the application of EM to clustering here. We assume that the data given in the form of $\mathcal{X}$ is incomplete because the cluster labels associated with the patterns are not available. The log likelihood function is

$$l(\theta) \ = \ ln \ p(\mathcal{X} \mid \theta) \tag{3}$$

where θ is the vector of parameters, corresponding to the distribution/s to be learnt. Let $z_1, z_2, \cdots, z_K$ be the K hidden/latent variables corresponding to the K unknown clusters. Let the joint density of $\mathcal{X}$ and z conditioned on θ be $p(\mathcal{X}, z \mid \theta)$. Then we can write the log likelihood as the marginal of the conditioned joint likelihood given by

$$l(\theta) \ = \ ln \ \sum_{z} p(\mathcal{X}, z \mid \theta) \tag{4}$$

This equality is difficult to deal with as it involves logarithm of a sum. A form of the Jensen's inequality is useful in simplifying such an expression. We discuss this inequality next.

6.2.1 Jensen's Inequality

It is important to examine the notion of *convex function* before we discuss Jensen's inequality.
Definition: A *function* $f(t)$ *is convex* if

$$f(\alpha t_0 \ + \ (1-\alpha)t_1) \leq \alpha f(t_0) + (1-\alpha) f(t_1) \ for \ 0 \leq \alpha \leq 1. \tag{5}$$

For example, $f(t) \ = \ t^2$ is a convex function because

$$f(\alpha t_0 + (1-\alpha)t_1) \ = \ (\alpha t_0 + (1-\alpha)t_1)^2 \ = \ \alpha^2 t_0^2 + (1-\alpha)^2 t_1^2 + 2\alpha(1-\alpha)t_0 t_1 \tag{6}$$

$$and \ \alpha f(t_0) + (1-\alpha) f(t_1) \ = \ \alpha t_0^2 + (1-\alpha) t_1^2 \tag{7}$$

(7)-(6) gives us

$$\alpha(1-\alpha)[(t_0 - t_1)^2] \tag{8}$$

which is equal to 0 if either $\alpha = 0$ or $\alpha = 1$; otherwise it is proportional to $(t_0 - t_1)^2$ which is non-negative as it is the square of a real number. So, we can infer that
$left\ hand\ side\ of\ (6)\ is \leq\ left\ hand\ side\ of\ (7)$ which shows that $f(t)$ is convex.

Theorem: If $f(t)$ is a convex function then

$$f(\sum_{i=1}^{K} \alpha_i t_i) \leq \sum_{i=1}^{K} \alpha_i f(t_i)\ where\ \alpha_i \geq 0\ for\ 1 \leq i \leq K\ and \sum_{i=1}^{K} \alpha_i = 1. \quad (9)$$

Proof: We show the result using mathematical induction.

- **Base case**: The property holds for $K = 2$ as

  $$f(\alpha_1 t_1 + \alpha_2 t_2) \leq \alpha_1 f(t_1) + \alpha_2 f(t_2)\ for\ \alpha_1, \alpha_2 \geq 0\ and\ \alpha_1 + \alpha_2 = 1 (10)$$

  This is because f is a convex function.

- **Induction Hypothesis:** Let the property hold for $K = l$. That is

  $$f(\sum_{i=1}^{l} \alpha_i t_i) \leq \sum_{i=1}^{l} \alpha_i f(t_i),\ where\ \alpha_i \geq 0\ for\ 1 \leq i \leq l\ and\ \sum_{i=1}^{l} \alpha_i = 1 (11)$$

- **Induction Step:** Using the Induction Hypothesis (inequality in (11)) we need to show that

  $$f(\sum_{i=1}^{l+1} \alpha_i t_i) \leq \sum_{i=1}^{l+1} \alpha_i f(t_i),\ where\ \alpha_i \geq 0\ for\ 1 \leq i \leq l+1\ and\ \sum_{i=1}^{l+1} \alpha_i = 1 (12)$$

  Consider the left hand side of the inequality (12) which could be simplified as

  $$f(\textstyle\sum_{i=1}^{l+1} \alpha_i t_i) = f(\sum_{i=1}^{l} \alpha_i t_i + \alpha_{l+1} t_{l+1})$$

  We can write it in the required form by multiplying and dividing by $(1 - \alpha_{l+1})$ as

  $$f[(1 - \alpha_{l+1}) \sum_{i=1}^{l} \alpha_i \frac{t_i}{1 - \alpha_{l+1}} + \alpha_{l+1} t_{l+1}] \quad (13)$$

Using the base case we can show that the expression in (13) is
$\leq\ (1-\alpha_{l+1}) f[\sum_{i=1}^{l} \frac{\alpha_i}{(1-\alpha_{l+1})} t_i] + \alpha_{l+1} f(t_{l+1})$

$\leq\ (1-\alpha_{l+1}) \sum_{i=1}^{l} \frac{\alpha_i}{(1-\alpha_{l+1})} f(t_i) + \alpha_{l+1} f(t_{l+1})$

$= \sum_{i=1}^{l+1} \alpha_i f(t_i) =$ *right hand side of* (12) thus proving the result.

We can use Jensen's inequality to simplify the expression of logarithm of the sum seen in (4); it is transformed to a sum of logarithm form which is easier to deal with. In order to achieve it we need to have appropriate αs which are non-negative and add upto 1. Probability mass function (discrete random variable) is ideally suited for this and we know from (4) that z is a discrete random variable. This prompts us to find a suitable form for the mass function on z. Further, we need to recognize that the function under consideration in (4) is natural logarithm (ln) which is a concave function or equivalently $-ln(x)$ is convex. In order to show that ln is concave we can use the following definition.

Definition: A *function f is concave* on an interval if its second derivative $f''(x)$ is negative in the interval.

Note that if $f(x) = ln(x)$ then $f''(x) = -\frac{1}{x^2}$ which is strictly decreasing if $x > 0$. So, $ln(x)$ is concave and $-ln(x)$ is convex.

We can rewrite equation (4) by multiplying and dividing by $p(z \mid \mathcal{X}, \theta_i)$, where θ_i is the estimate of θ at the i^{th} step of the iterative scheme that we need to use, as

$$l(\theta) = ln \sum_z p(z \mid \mathcal{X}, \theta_i) \frac{p(\mathcal{X}, z \mid \theta)}{p(z \mid \mathcal{X}, \theta_i)} \tag{14}$$

By using the Jensen's inequality seen earlier, we can show that the above is

$$\geq \sum_z p(z \mid \mathcal{X}, \theta_i)\ ln \frac{p(\mathcal{X}, z \mid \theta)}{p(z \mid \mathcal{X}, \theta_i)} \tag{15}$$

It is possible to show that θ that maximizes the *right hand side* of (15) is the same as the θ that maximizes (14). Further as θ appears only in the numerator of the argument of ln, the same θ that maximizes (15) also maximizes

$$\sum_z p(z \mid \mathcal{X}, \theta_i)\ ln\ p(\mathcal{X}, z \mid \theta) = Expectation(ln\ p(\mathcal{X}, z \mid \theta)) \tag{16}$$

where the expectation is over z conditioned on $\mathcal{X}$ and the value of θ in the form of θ_i. So, the two important steps in the EM algorithm are:

1. **Expectation or E Step:** Compute $E_{z|\mathcal{X},\theta_i}(ln\ p(\mathcal{X}, z \mid \theta))$.
2. **Maximization or M Step:**

$$\theta_{i+1} \ = \ \underset{\theta}{argmax} \quad E_{z|\mathcal{X},\theta_i}(ln\ p(\mathcal{X}, z \mid \theta))$$

EM algorithm also can reach only a locally optimal solution.

6.2.2 An Example

Let us consider a one-dimensional example with two clusters. Let the data set be

$\mathcal{X} \ = \ \{2.1, 5.1, 1.9, 4.9\}$.

It is easy to see the two intuitively appealing clusters in the form of $\{2.1, 1.9\}$ $\{5.1, 4.9\}$. Now let us examine how EM can be used to cluster the set $\mathcal{X}$. Let the clusters be c_1 and c_2. Let us assume that these data points are drawn from two normals each with the same variance ($\sigma^2 = 1$) of value 1. The parameter vector $\theta \ = \ (\mu_1, \mu_2)^t$ where μ_1, μ_2 are the means of the normals is to be estimated. The log likelihood function is given by

$$ln\ p(\mathcal{X}, z \mid \theta) \ = \ ln\ \Pi_{i=1}^{4} p[X_i, P_i \mid \theta] \ = \ \sum_{i=1}^{4} ln\ p[X_i, P_i \mid \theta] \tag{17}$$

where $P_i \ = \ (P_{i1}, P_{i2})$; P_{ij} = Probability that X_i belongs to cluster c_j.

$$E[ln\ p(\mathcal{X}, z \mid \theta)] = \ \sum_{i=1}^{4} [ln\ (\frac{1}{\sqrt{2\pi\sigma^2}} - \frac{1}{2\sigma^2} \sum_{j=1}^{2} E(P_{ij})(x_i - \mu_j)^2] \tag{18}$$

Here, $E(P_{ij}) \ = \ P_{ij}$ as i and j are fixed. Looking at the right hand side of the above equation, the first term $ln\ (\frac{1}{\sqrt{2\pi\sigma^2}})$ in the summation does not play any role as we are interested in μs only and σ is known. By equating the gradient, with respect to θ, of the resulting expression we get

$\sum_{i=1}^{4} P_{ij}(X_i - \mu_j) = 0$ which implies

$$\mu_j = \frac{\sum_{i=1}^{4} P_{ij} X_i}{\sum_{i=1}^{4} P_{ij}}. \tag{19}$$

So, in order to estimate the parameter vector θ or equivalently, the values of $\mu_1,\ \mu_2$ we need to get the values of $P_{ij}s$; note that P_{ij} is the probability that X_i belongs to cluster C_j or equivalently X_i is generated by the corresponding (j^{th}) normal density. So,

$$P_{ij} = \frac{exp[-\frac{1}{2\sigma^2}(X_i - \mu_j)^2]}{\sum_{l=1}^{2} exp[-\frac{1}{2\sigma^2}(X_i - \mu_l)^2]} \tag{20}$$

So, in this case, equations (20) and (19) characterize the *Expectation* and *Maximization* steps and they are repeated iteratively till convergence.

Let $\theta_0 = (2, 4)^t$ be the initial selection of the μs. Let us consider computation of P_{11}. By using (20) and $\sigma = 1$, it is given by

$$P_{11} = \frac{exp[-\frac{1}{2}(2.1 - 2)^2]}{exp[-\frac{1}{2}(2.1 - 2)^2] + exp[-\frac{1}{2}(2.1 - 4)^2]} = 0.728.$$

In a similar manner $P_{21},\ P_{31},\ P_{41}$ can be computed. Once we have these values, we can estimate μ_1 using equation (19). In a similar manner one can estimate P_{i2}, for $i = 1, \cdots, 4$ and μ_2. These values are shown in Table 4. So, at the end of the first iteration we get $\mu_1 = 2.76$ and $\mu_2 = 4.53$. So,

j	P_{1j}	P_{2j}	P_{3j}	P_{4j}	μ_j
1	0.728	0.018	0.9	0.015	2.76
2	0.272	0.982	0.1	0.985	4.53

Table 4: Values during the First Iteration

$\theta_1 = (2.76, 4.53)^t$. Using this value of θ, the corresponding values of the parameters are given in Table 5. Table 6 shows the parameters in the third iteration. After some more iterations we expect the value of $\theta = (2,\ 5)$ to be reached. The corresponding densities are shown in Figure 2.

j	P_{1j}	P_{2j}	P_{3j}	P_{4j}	μ_j
1	0.94	0.09	0.96	0.1	2.26
2	0.06	0.91	0.04	0.9	4.6

Table 5: Values during the Second Iteration

j	P_{1j}	P_{2j}	P_{3j}	P_{4j}	μ_j
1	0.946	0.214	0.97	0.03	2.345
2	0.04	0.786	0.03	0.97	4.88

Table 6: Values during the Third Iteration

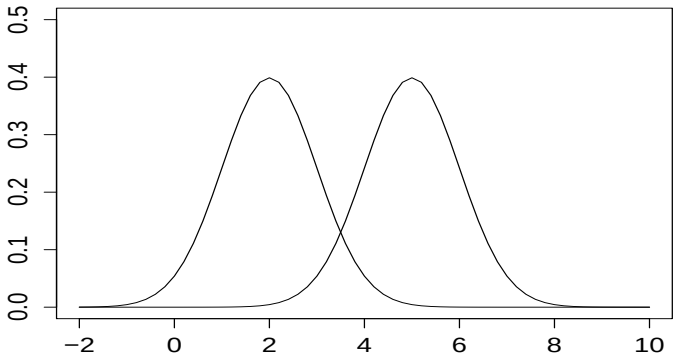


Figure 2: The Densities Corresponding to the Two Clusters

7 Topic Models

One of the important areas of current interest is large-scale document processing. It has a variety of applications including:

- **Document retrieval:** Here the documents in a collection are represented, typically, using the bag-of-words (BOW) model in the case of text. In the BOW model, the order of occurrence of words is not important; their frequency of occurrence is important. Here for a given input

query, a collection of documents is presented in a ranked order as output. Classification, clustering and ranking of documents are important machine learning tasks that are useful in this application.

- **Spam filtering:** Spam electronic mails are unsolicited mails which can be wasting resources of the recipient. So, spam mails need to be identified and isolated appropriately. This activity requires clustering and classification of email messages.

- **Verification (biometrics):** There are several applications concerned with forensics and cyber security where biometrics plays an important role. These include identification based on biometrics like fingerprint, speech, face, and iris. So, classification and clustering are important again in this application.

- **Pin-Code recognition:** Automatic pin-code recognition is required to route surface mail to its destination mechanically. This involves recognition of hand-written numerals.

- **Multimedia and multi-lingual document classification:** Classification and summarization of reports in legal and medical domains.

It is possible to view most of the data stored and processed by a machine as comprising of documents. In this sense, the following are some of the popularly considered document categories:

- *Web pages:* These are easily the most popular type of semi-structured documents. Every search engine crawls and indexes web pages for possible information retrieval; typically search engines return, for a query input by the user, a collection of documents in a ranked manner as output. Here, in addition to the *content* of the webpages, the link structure in terms of *hyperlinks* is also exploited.

- *Academic publications:* This kind of documents are also prominent to the extent that their content and links in terms of Co-citations are used by search engines in document retrieval.

- *Research grant applications:* It is typically a semi-structured text document containing charts, graphs, and some images.

- *Company reports:* It may also be semistructured text document.

- *Newspaper articles:* This could be typically a short text document and it can have images at times.
- *Bank transaction slips:* This is another semi-structured short text document.
- *Manual pages:* This is another short size text document with possible figures.
- *Tweets:* Tweets are short text documents and they may employ several non-standard words.
- *Encyclopedia:* This could be a very lengthy and comprehensive text document with figures and images combined. Wikipedia is a well-known example where search is facilitated.
- *Images (video):* It is an example of a non-textual document.
- *Speech records:* These are also examples of non-textual documents.
- *Fingerprints:* It is also a non-textual type of document used as a biometric.
- *Electronic mails:* It is one of the most popularly used type of documents for communication between two or more individuals; it is typically a short text.
- *Health records:* It could be a multi-media document. The prescription and diagnosis in handwritten textual form, ultrasound and x-rays in the form of images form a part of the document.
- *Legal records:* It is another textual document and words here may have to be precise.
- *Software code:* Analysis of software code is gaining a lot of importance to identify bugs and inefficient parts of the code.
- *Bug reports:* These are reports used to indicate various bugs encountered in the software package and how some of them are fixed periodically.

There could be several other kinds of documents. In order to analyze document collections, a variety of topic based schemes have been proposed. It is possible to view these methods to be performing factorization of the document term matrix. So, we consider these matrix factorization based methods next.

7.1 Matrix Factorization Based Methods

Let X be the document term matrix of size $n \times l$ which means that there are n documents and l terms. Typically l could be very large in most of the practical settings. A possible generic factorization is:

$$X_{n \times l} \;=\; B_{n \times K}\; D_{K \times K}\; C_{K \times l} \tag{21}$$

The basic idea behind such a factorization is that it permits us to work with documents represented in a lower-dimensional (K-dimensional) space. The value of K corresponds to the number of non-zero eigenvalues of XX^t or the rank of the matrix X. In other words we can work with the B matrix instead of the X matrix. Further, it is possible to view this as working with K *topics/clusters* instead of l terms or words.

7.1.1 Why Should this Work?

It is well-known that conventional distance based methods fail to characterize neighbors in high-dimensional spaces. This is because the ratio of $d(x, NN(x))$ and $d(x, FN(x))$ tends to 1 as the dimensionality gets larger and larger; here $d(x, NN(x))$ is the distance between a point x and its nearest neighbor, $NN(x)$, and $d(x, FN(x))$ is the distance between x and its farthest neighbor, $FN(x)$. Two different ways of solving this problem are:

1. **Explicit Dimensionality Reduction**: Here the data in the high-dimensional space is projected into a low-dimensional space by using either a linear or a non-linear projection tool. For example Random Projections are useful in unsupervised feature extraction; similarly mutual information has been successfully used in feature selection. Once the dimensionality is reduced it is possible to successfully use classifiers based on matching. In such a case the data matrix X becomes a n x K matrix instead of n x l where K is the number of features extracted from the given l features.

2. **Implicit Dimensionality Reduction**: Instead of explicit selection it is possible to use matrix factorization approaches to implicitly get the features. For example if we assume that D is an identity matrix of size K x K, then the resulting equation is $X = B\ C$; note that even though X is a collection of n data points (rows) in a possibly high-dimensional space of l terms (columns), the resulting B matrix represents the n patterns in a reduced dimensional (K) space. This amounts to dimensionality reduction; specifically each of the K columns of B may be viewed as a linear combination of the l columns in A. Further it is important that entries of C are well-behaved; popularly the objective function of the optimization problem deals with some error between A and its approximation in the form of BC. In addition C is constrained to be sparse, for example. We deal with different factorization approaches in this chapter.

7.2 Divide-and-Conquer Approach

The simplest scheme to deal with high-dimensional spaces is to partition the set of features $\mathcal{F}$ into some P blocks $F_1,\ F_2,\ \cdots, F_P$ such that $\mathcal{F} = \bigcup_{i=1}^{P} F_i$; cluster subpatterns in each of these P subspaces and use the cluster representatives to realize a compact set of prototypes. We can use them in large data classification. We illustrate it with a simple example.

Let us consider the data set shown in Table 7; there are two classes and each pattern is four dimensional as indicated by the features $f_1,\ f_2,\ f_3$ and f_4. Now given a test pattern $(2, 2, 2, 2)$, it is classified as belonging to class 1 using the nearest neighbor classifier as its nearest neighbor is pattern number 4, given by $(1, 2, 2, 2)$, which is in class 1. Let this set of features be partitioned into $F_1 = \{f_1, f_2\}$ and $F_2 = \{f_3, f_4\}$. Now we cluster data in class separately and while clustering we consider each subset of features separately. We explain the specific steps as:

- Consider patterns 1 to 4 which are from class 1. Cluster these patterns based on F_1 and F_2 separately.
  - The subpatterns based on F_1 are $(1, 1)$, $(1, 2)$, $(1, 1)$ and $(1, 2)$. If we seek two clusters using any conventional algorithm, we get the two clusters: $\{(1, 1), (1, 1)\}$, and $\{(1, 2), (1, 2)\}$. The cluster centroids are $(1, 1)$ and $(1, 2)$ respectively.

Pattern No.	f_1	f_2	f_3	f_4	Class
1	1	1	1	1	1
2	1	2	1	1	1
3	1	1	2	2	1
4	1	2	2	2	1
5	6	6	6	6	2
6	6	7	6	6	2
7	6	6	7	7	2
8	6	7	7	7	2

Table 7: Patterns corresponding to two classes

  - The subpatterns based on F_2 are $(1,1),\ (1,1),\ (2,2),\ (2,2)$ which form two clusters with $(1,1)$ and $(2,2)$ as representatives.

- Similarly consider patterns 5 to 8 that belong to class 2. Again by clustering the subpatterns based on F_1 and F_2 we have
  - Cluster representatives based on F_1 are $(6,6)$ and $(6,7)$.
  - Cluster representatives based on F_2 are $(6,6)$ and $(7,7)$.

We collect these cluster representatives corresponding to different classes based on different feature subsets. In this example, the relevant details are shown in Table 8. It is possible to use data in table 8 to classify a test pattern.

Representative Number	Based on F_1	Based F_2	Class
1	1 1	1 1	1
2	1 2	2 2	1
1	6 6	6 6	2
2	6 7	7 7	2

Table 8: Cluster Representatives of the two classes

For example, consider the pattern $(2,2,2,2)$; we consider its subpatterns corresponding to F_1 and F_2 which are $(2,2)$ and $(2,2)$. Now we obtain the nearest neighbors of these subpatterns in each class. They are given by

- The nearest subpatterns of $(2, 2)$ (based on F_1) are $(1, 2)$ from class 1 and $(6, 6)$ from class 2; of these class 1 subpattern $(1, 2)$ is nearer.
- The nearest neighbors of $(2, 2)$ (based on F_2) are $(2, 2)$ from class 1 and $(6, 6)$ from class 2; of these two $(2, 2)$ is nearer.

Observing that the nearest subpatterns are from class 1 based on both F_1 and F_2, we assign $(2, 2, 2, 2)$ to class 1. If we concatenate the nearest subpatterns $(1, 2)$ and $(2, 2)$ then we get $(1, 2, 2, 2)$ as the nearest neighbor pattern. We can abstract the whole scheme using the following steps.

1. Let there be C classes with the i^{th} class having n_i patterns. Partition patterns of the i^{th} class into P_i subsets of features. In the above example $C = 2$ and $P_1 = P_2 = 2$ as in both the classes we have used the same F_1 and F_2.
2. For the test pattern obtain the nearest neighbors from each of the C classes.
    - For class i consider the P_i subsets of features. Find the nearest subpattern of each test subpattern in each of the corresponding subsets from the training data.
    - Concatenate the P_i nearest subpatterns of the test subpatterns. Compute the distance between the test pattern and this concatenated pattern.
3. Assign the test pattern to class j if the nearest pattern from the j^{th} class is nearer than the other classes.

It is possible to consider several variants of this simple divide-and-conquer approach. Some of them are:

- Instead of using a single nearest neighbor one can use K nearest neighbors from each class to realize a robust classifier.
- Instead of concatenating the P_i nearest subpatterns one can consider K nearest subpatterns based on each of the P_i feature subsets. Then generate neighbors by concatenating P_i subpatterns selecting one from each of the K subpatterns selected based on each of the P_i feature sets. This leads to selection of P_i^K concatenated patterns. Select K of these

patterns based on nearness to the test pattern. if there are C classes then we get CK neighbors, K from each class. Select K nearest of these and use majority voting to decide the class label.

- Computation of distance between the test pattern and a neighbor can be obtained by summing partial distances across the P_i subpatterns in some cases. This is possible when squared euclidean distance or city-block distance is used to compute partial distances.

7.3 Latent Semantic Analysis

In this factorization D is a diagonal matrix. *Latent Semantic Analysis (LSA)* employs this factorization directly. In such a case the above factorization exploits the Singular Value Decomposition (SVD) where the diagonal entries in D are singular values and they correspond to the eigenvalues of the matrices XX^t and X^tX where X^t is the transpose of X. This may be explained as follows:

Consider XX^t; it is a square matrix of size $n \times n$. If β is an eigenvector and λ the corresponding eigenvalue then we have

$$XX^t\beta \;=\; \lambda\beta$$

by premultiplying both sides by X^t we have

$$X^t(XX^t\beta) \;=\; X^t(\lambda\beta)$$

By using the associativity of matrix multiplication and noting that λ is a scalar, we have

$$(X^tX)X^t\beta \;=\; \lambda(X^t\beta)$$

Note that X^tX is a square matrix of size $l \times l$; β is a $n \times 1$ vector and $X^t\beta$ is a $l \times 1$ vector. By assuming that $X^t\beta \;=\; \gamma$ we can write the above equation as

$$X^tX\gamma \;=\; \lambda\gamma$$

which means that γ is the eigenvector of X^tX with the associated eigenvalue being λ. This means that λ is an eigenvalue of both XX^t and X^tX but with the corresponding eigenvectors being β and γ $(= X^t\beta)$ respectively.

Given a matrix X of size $m \times n$ the singular value decomposition is given by

$$X \;=\; B\; D\; C$$

where size of B is $m \times K$; D is a $K \times K$ diagonal matrix; and C is of size $K \times n$. Here B consists of K eigenvectors of the matrix XX^t; these are eigenvectors corresponding to the K nonzero eigenvalues of the matrix XX^t. Similarly C is composed of K eigenvectors of X^tX as the K rows of C. We now illustrate these ideas using an example.

7.3.1 Illustration of SVD

Consider a 3×2 size matrix X given by

$$X \;=\; \begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Note that X^tX is given by

$$X^tX \;=\; \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

The characteristic equation of X^tX is $\lambda^2 \; - 4\lambda \; + 3 \;=\; 0$ and so the roots of the equation or eigenvalues of X^tX are 3 and 1.

The corresponding eigenvectors are

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

The corresponding normalized vectors are:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$$

and

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

In a similar manner it is possible to observe that the matrix XX^t is

$$\begin{pmatrix} 2 & 1 & -1 \\ 1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

The corresponding eigenvalues are 3, 1, and 0 and the eigenvectors corresponding to the nonzero eigenvalues after normalization are

$$\begin{pmatrix} \frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} \end{pmatrix}$$

and

$$\begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

The entries of the diagonal matrix Σ are the square roots of the eigenvalues (singular values) 3 and 1. So, the various matrices are

$$B = \begin{pmatrix} \frac{2}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

$$D = \begin{pmatrix} \sqrt{3} & 0 \\ 0 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

So, finally we have the SVD of X given by

$$\begin{pmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{6}} & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \sqrt{3} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

7.4 SVD and PCA

Principal component analysis (PCA) is a well-known tool for dimensionality reduction. Here, linear combinations of the original features are obtained which are uncorrelated. Mathematically it amounts to computing the eigenvectors of the covariance matrix. The covariance matrix is symmetric and the entries are real numbers; so, the eigenvectors are orthogonal. The eigenvector corresponding to the largest eigenvalue is the first principal component; the one corresponding to the next largest eigenvalue is the second principal component, and so on. We illustrate the computation of PCA using a two-dimensional example.

7.4.1 Example to illustrate PCA

Consider the four two-dimensional patterns given by

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \begin{pmatrix} 2 \\ 1 \end{pmatrix} \begin{pmatrix} 6 \\ 7 \end{pmatrix} \begin{pmatrix} 7 \\ 6 \end{pmatrix}$$

The sample mean of the 4 data points is $\begin{pmatrix} 4 \\ 4 \end{pmatrix}$. The zero mean normalized set of points is

$$\begin{pmatrix} -3 \\ -2 \end{pmatrix} \begin{pmatrix} -2 \\ -3 \end{pmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

The sample covariance matrix of the data is given by

$$\frac{1}{4}[(-3,-2)^t(-3,-2) + (-2,-3)^t(-2,-3) + (2,3)^t(2,3) + (3,1)^t(3,1)]$$
$$= \frac{1}{4}\begin{pmatrix} 26 & 24 \\ 24 & 26 \end{pmatrix} = \begin{pmatrix} 6.5 & 6 \\ 6 & 6.5 \end{pmatrix}$$

The eigenvalues of the matrix are 12.5 and 0.5 and the respective eigenvectors are $\begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. After normalization we get the unit-norm orthogonal eigenvectors given by $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$. The four data points and the corresponding principal components are shown in Figure 3.

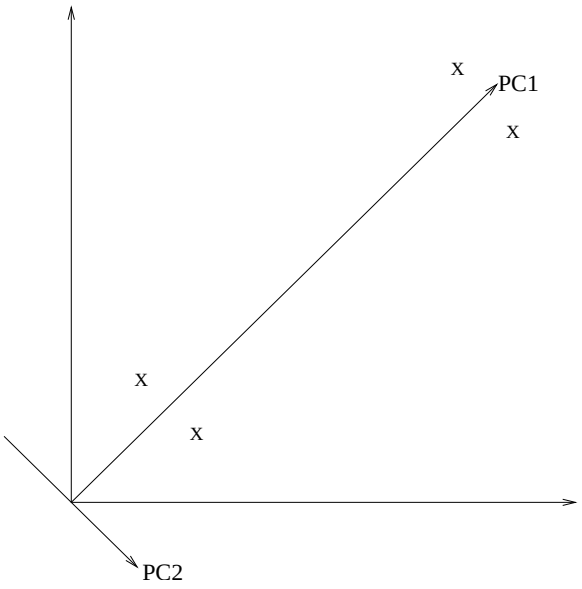


Figure 3: The two principal components

7.4.2 Computing PCs using SVD

Consider the four data points after zero-mean normalization. It is given by the matrix X of size 4×2:

$$X = \begin{pmatrix} -3 & -2 \\ -2 & -3 \\ 2 & 3 \\ 3 & 2 \end{pmatrix}$$

The SVD of X is given by

$$\begin{pmatrix} -3 & -2 \\ -2 & -3 \\ 2 & 3 \\ 3 & 2 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \sqrt{50} & 0 \\ 0 & \sqrt{2} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Note that the rows of the matrix C correspond to the principal components of the data. Further, the covariance matrix is proportional to X^tX given by

$$X^tX = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 50 & 0 \\ 0 & 2 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

In most of the document analysis applications it is not uncommon to view a document collection as a document-term matrix, X as specified earlier. Typically such a matrix is large in size; in a majority of the practical applications the number of documents is large and the number of terms in each document is relatively small. However such data is high-dimensional and so the matrix can be very sparse. This is because even though the number of terms in a document is small, the total number of distinct terms in the collection could be very large; out of which a small fraction of terms appear in each of the documents which leads to sparsity. This means that dimensionality reduction is essential for applying various classifiers effectively.

Observe that the eigenvectors of the covariance matrix are the principal components. Each of these is a linear combination of the terms in the given collection. So, instead of considering all possible eigenvectors, only a small number of the principal components are considered to achieve dimensionality reduction. Typically the number of terms could be varying between 10,000 to 1 million whereas the number of principal components considered could be between 10 to 100. One justification is that people use a small number of topics in a given application context; they do not use all the terms in the given collection.

Latent semantic analysis involves obtaining topics that are *latent* and possibly *semantic*. It is based on obtaining latent variables in the form of linear combinations of the original terms. Note that the terms are observed in the given documents; however the topics are *latent* which means topics are not observed. Principal components are such linear combinations. The eigenvalues of the covariance matrix represent variances in the directions of the eigenvectors. The first principal component is in the direction of maximum variance. In a general setting dimensionality reduction is achieved by considering only the top K principal components where K is smaller than the rank, r, of the matrix X. Specifically dimensionality reduction is useful when the data points in X are high-dimensional.

It is possible to use the SVD to obtain a low-rank representation X_K which is an approximation of X. This is obtained by considering the largest K singular values and forcing the remaining $r - K$ singular values in D to be zero. So, X_K is given by

$$X_K \ = \ BD_KC$$

where D_K is a $r \times r$ diagonal matrix with the top K eigenvalues in D and the remaining $r-K$ diagonal values to be zero. Note that X_K is an approximation

of X. It is an optimal approximation in the sense that among all possible K rank matrices X_K has the minimal Frobenius norm with X. It is possible to show that this Frobenius norm is

$$|| X - X_K ||_F = \lambda_{K+1}$$

where λ_{K+1} is the largest singular value ignored among the least $r - K$ singular values.

There are claims that the resulting reduced dimensional representation is *semantic* and can handle both synonymy and polysemy in information retrieval and text mining. Here, by synonymy we mean two words having the same meaning are synonymous. Similarly we have polysemy when the same word has multiple meanings. This will have impact on the similarity between two documents. Because of synonymy the similarity value computed using dot product type of functions will be less than the intended. For example if *car* and *automobile* are used interchangeably (synonymously) in a document and only car is used in the query document then the similarity measure will fail to take into account the occurrences of automobile. In a similar manner because of polysemy it is possible that similarity between a pair of documents is larger than what it should be. For example if *tiger* is used in a document both to mean an animal and say airlines and the query document has tiger in only one sense then the dot product could be larger than the intended value.

An important feature of the SVD is that it is a deterministic factorization scheme and the factorization is unique. Here each row of the matrix C is a topic and it is an assignment of weights to each of the terms. The entry C_{ij} is the weight or importance of the j^{th} term $(j = 1, \cdots, l)$ to the i^{th} topic. The entry D_{ii} in the diagonal matrix indicates some kind of weight assigned to the entire i^{th} topic.

7.5 Probabilistic Latent Semantic Analysis (PLSA)

Latent semantic analysis employs a deterministic topic model that is reasonably popular in analyzing high-dimensional text document collections. However it does not offer a framework to synthesize or generate documents. Probabilistic latent semantic analysis (PLSA) is a generative model; it is also a topic model where each topic is a mapping from set of terms to [0,1]. Specifically a topic assigns a probability to each term.

It is possible to view the probability, $P(t)$, of a term t as

$$P(t) \;=\; \sum_{i=1}^{K} P(C_i)P(t|C_i) \; where \; \sum_{i=1}^{K} P(C_i) \;=\; 1$$

here C_i is a soft cluster or topic. It is possible to use the Bayes rule to write the joint probability $P(d, t_j)$ as

$$P(d, t_j) \;=\; P(d)P(t_j|d) \; where \; P(t_j|d) \;=\; \sum_{i=1}^{K} P(t_j|C_i)P(C_i|d)$$

It is possible to view the PLSA as a matrix factorization scheme as follows:

$$P(d, t_j) \;=\; \sum_{i=1}^{K} P(d|C_k)P(C_k)P(t_j|C_k)$$

This can be viewed as a matrix factorization of the form $X \;=\; BD_KE$, where the diagonal matrix D_K is made up of probabilities of the K topics. Specifically the i^{th} element on the diagonal is given by $P(C_i)$. The element B_{ij} is given by $P(d_i|C_k)$ where d_i is the i^{th} document and C_k is the k^{th} topic. Similarly, E_{jk} corresponds to $P(t_j|C_k)$.

Learning the model involves, given the document topic matrix X to get the B, D_K and E matrices or equivalently getting the values of $P(d_i|C_k)$, $P(C_i)$, and $P(t_j|C_k)$. Alternatively, one may consider the parameters to be $P(d_i)$, $P(t_j|C_k)$ and $P(C_k|d_i)$. These are obtained using the expectation-maximization (EM) algorithm. The EM iteratively goes through the expectation and maximization steps. In the expectation step the expected value of the log-likelihood function is obtained and in the maximization step, parameter values are obtained based on the maximization of the expected value. Let us assume the following:

- Let the input X be a document collection of n documents and each is a vector in an l dimensional space; this means the vocabulary size is l.
- Let $n(d_i, t_j)$ be the number of times term t_j occurred in document d_i and $P(d_i, t_j)$ be the probability of term t_j in document d_i.
- Let us assume the *bag of terms model* which means each term occurs independent of the others and it may occur more than once in a document.

- Let θ be the parameter vector with components $P(d_i)$, $P(t_j|C_k)$ and $P(C_k|d_i)$; learning involves estimating the values of these parameters from X.

The likelihood function, $L(\theta)$, is given by

$$L(\theta) \;=\; \prod_{i=1}^{n}\prod_{j=1}^{l} P(d_i,t_j)^{n(d_i,t_j)}$$

The log-likelihood function, $l(\theta)$ is given by

$$l(\theta) \;=\; \sum_{i=1}^{n}\sum_{j=1}^{l} n(d_i,t_j) logP(d_i,t_j)$$

$$Hence,\; l(\theta) \;=\; \sum_{i=1}^{n}\sum_{j=1}^{l} n(d_i,t_j) log[P(d_i)P(t_j|C_k)P(C_k|d_i)]$$

So, expected value of $l(\theta)$ with respect to the latent cluster C_k conditioned on d_i and t_j is given by $E_{C_k|d_i,t_j}(l(\theta)) \;=$

$$\sum_{i=1}^{n}\sum_{j=1}^{l} n(d_i,t_j) \sum_{k=1}^{K} P(C_k|t_j,d_i)[logP(d_i) + logP(t_j|C_k) + logP(C_k|d_i)]$$

Here, $P(C_k|t_j,d_i) \;=\; \frac{P(t_j|C_k)P(C_k|d_i)}{\sum_{k=1}^{K} P(t_j|C_k)P(C_k|d_i)}$.

In addition to the expected value of $l(\theta)$, we need to consider the constraints on the probabilities which are

$$\sum_{i=1}^{n} P(d_i) \;=\; 1;\; \sum_{j=1}^{l} P(t_j|C_k) \;=\; 1;\; \sum_{j=1}^{l} P(C_k|d_i) \;=\; 1.$$

The corresponding Lagrangian is given by

$$E_{C_k|d_i,t_j}(l(\theta)) \;+\; \alpha_1(\sum_{i=1}^{n} P(d_i) - 1) \;+\; \alpha_2 \sum_{k=1}^{K}(\sum_{j=1}^{l} P(t_j|C_k) - 1) + \alpha_3 \sum_{i=1}^{n}(\sum_{k=1}^{K} P(C_k|d_i) - 1)$$

where α_1, α_2, and α_3 are the Lagrange variables.

By taking the partial derivatives of this Lagrange function with respect to the three parameters $P(d_i)$, $P(t_J|C_k)$, and $P(C_k|d_i)$ and equating them to zero we get the following estimates:

- $P(d_i) = \frac{\sum_{j=1}^{l}\sum_{k=1}^{K} n(d_i,t_j)P(C_k|t_j,d_i)}{\sum_{i=1}^{n}\sum_{j=1}^{l}\sum_{k=1}^{K} n(d_i,t_j)P(C_k|t_j,d_i)}$

- $P(t_j|C_k) = \frac{\sum_{i=1}^{n} n(d_i,t_j)P(C_k|t_j,d_i)}{\sum_{j=1}^{l}\sum_{i=1}^{n} n(d_i,t_j)P(C_k|t_j,d_i)}$

- $P(C_k|d_i) = \frac{\sum_{j=1}^{l} n(d_i,t_j)P(C_k|t_j,d_i)}{\sum_{j=1}^{l}\sum_{k=1}^{K} n(d_i,t_j)P(C_k|t_j,d_i)}$

7.6 Nonnegative Matrix Factorization

Nonnegative matrix factorization is another popular soft clustering tool. Here, in the factorization shown in (21), D is the Identity matrix I of size KxK and so $X = BC$. Further, given that X has non-negative entries, we insist that B and C also have only non-negative entries. Hence this factorization is called Nonnegative matrix factorization (NMF). Typically we seek an approximate factorization of X into the product BC. There are several possibilities for realizing the approximate factors B and C; some of them are:

- Obtain B and C such that $|| X-BC ||^2$ is minimized. This corresponds to the minimization of squared euclidean distance between the matrices X and BC; such a kind of expression stands for the squared *Frobenius Norm* or element-wise distance given by
  $|| X - BC ||^2 = \sum_{ij} \left(X_{ij} - \sum_{k=1}^{K} B_{ik}C_{kj} \right)^2$.
  It is easy to observe that the minimum value of such a distance is zero when $X = BC$.

- Minimize a generalized *Kullback-Leibler divergence* between X and BC given by

$$\sum_{ij} \left[X_{ij} log \frac{X_{ij}}{\sum_{k=1}^{K}(B_{ik}C_{kj})} - X_{ij} + \sum_{k=1}^{k}(B_{ik}C_{kj}) \right]$$

When entries of X and BC are normalized or probabilities such that $\sum_{ij} X_{ij} = \sum_{ij} \sum_{k} (B_{ik} C_{kj}) = 1$; It could be seen that the minimum value of this divergence is zero.

So, the problem of the minimization of distance is

$Minimize \ || X - BC ||^2$ with respect to B and C
such that B and C have nonnegative entries. Equivalently, $B, C \geq 0$.

There is no guarantee that we can get the globally optimal solution. However, it is possible to get the local optimum using the following update rules.

$$B_{ij} \leftarrow B_{ij} \frac{(XC^t)_{ij}}{(BCC^t)_{ij}}$$
$$C_{ij} \leftarrow C_{ij} \frac{(B^t X)_{ij}}{(B^t BC)_{ij}}$$

It is possible to show some kind of equivalence between NMF and the K-Means algorithm; also between NMF and $PLSA$.

7.7 Latent Dirichlet Allocation (LDA)

Even though PLSA offers an excellent probabilistic model in characterizing the latent topics in a given collection of documents, it is not a fully generative model. It cannot explain documents which are not part of the given collection. Some of the important features of the PLSA are:

- A document d_i and a term t_j are assumed to be independent conditioned on a cluster C_k. Because of this we could write

$$P(d_i, t_j) = P(d_i) \sum_{k=1}^{K} P(t_j, C_k | d_i)$$

By using Bayes rule we can write $P(t_j, C_k | d_i)$ as $P(t_j, d_i | C_k) \frac{P(C_k)}{P(d_i)}$. This will mean that

$$P(d_i, t_j) = P(d_i) \sum_{k=1}^{K} P(t_j, d_i | C_k) \frac{P(C_k)}{P(d_i)}$$
$$= P(d_i) \sum_{k=1}^{K} P(t_j | C_k) P(d_i | C_k) \frac{P(C_k)}{P(d_i)}$$

The independence between d_i and t_j is used to simplify the previous expression which leads to

$$P(d_i, t_j) \;=\; P(d_i) \sum_{k=1}^{K} P(t_j|C_k) P(C_k|d_i)$$

- A document may contain multiple topics or equivalently belong to multiple clusters; $P(C_k|d_i)$ is learnt only on those documents on which it is trained. It is not possible to make probabilistic assignment to an unseen document.
- $P(t_j|C_k)$ and $P(C_k|d_i)$ are important parameters of the model. The number of parameters is $Kl + Kn$; so there will be a linear growth with corpus size and also can overfit.
- Each cluster C_k is characterized by l probabilities $P(t_j|C_k)$ for $j = 1, 2, \cdots, l$. So, a cluster C_k associates different probabilities to different terms. Specifically, a topic or a description of the cluster is explained by the l probabilities.

A more appropriate probabilistic topic model is the Latent Dirichlet Allocation (LDA); it exploits the conjugate prior property between Dirichlet and multinomial distributions. LDA is a truly generative probabilistic model that can assign probabilities to documents not present in the training set unlike the PLSA. Some of the important features of the LDA are

- Each document is a random mixture over latent topics or clusters; each cluster is characterized by a distribution over terms.
- It is assumed that there are a fixed number of topics or clusters that are latent and using them one can generate documents.
- Each topic is a multinomial over the l terms.
- It is assumed that the prior density is Dirichlet which is characterized by a K-dimensional vector θ such that the i^{th} component of θ, given by θ_i is non-negative for all j and $\sum_{k=1}^{K} \theta_k = 1$. The functional form of the density is given by

$$p(\theta|\alpha) \;=\; C(\alpha) \prod_{k=1}^{K} \theta_k^{\alpha_k - 1}$$

where

$$B(\alpha) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)}$$

where Γ stands for the Gamma function and α is the input parameter vector.

- Here Dirichlet is chosen because it is the conjugate prior of the multinomial distribution. Multinomial is useful here because topics/clusters are drawn using the multinomial distribution. Specifically description of the cluster C_k is viewed as a multinomial based on the parameter vector θ.

- Documents are given; terms in the document collection form the observed random variables. Clusters or topics are hidden or latent. The idea of learning or inference in this model is to estimate the cluster structure latent in the collection; typically these clusters are soft.

- The hidden variables characterizing the model are α, a K-dimensional vector and β, a matrix of size K x l when there are K soft clusters/topics and l distinct terms in the entire document corpus.

- In order to generate a document having N terms, it is assumed that each term is generated based on a topic. So, in order to generate a term, a topic is sampled and using the topic and β matrix a term is generated. This procedure is repeated N times to generate the document.

- The generative process may be characterized by a joint probability distribution over both the observed and hidden random variables. The posterior distribution of the latent cluster structure conditioned on the observed distribution of the terms in the document collection is derived out of this.

- The entire document generation process is formally characterized as follows:

  - The vector ϕ is sampled from a Dirichlet distribution $p(\phi|\beta)$ where ϕ is a vector of size K where the k^{th} component corresponds to the k^{th} cluster in the document collection.

- The vector θ is sampled from a Dirichlet distribution $p(\theta|\alpha)$ where θ is the vector of K cluster distributions at the document level.
- For each term in the document, sample a topic/cluster using a multinomial distribution $p(C_k|\theta)$; use a multinomial $p(t_j|\phi_k)$ to sample a term t_j.
- So, the probability of the document with N terms drawn independently is given by

$$p(d, \theta, \mathcal{C}, \phi|\alpha, \beta) \;=\; p(d|\phi)p(\phi|\beta)p(\mathcal{C}|\theta)p(\theta|\alpha)$$

The document generation scheme is explained using this equation. It explains how a collection of documents is generated by using the parameters α and β and assuming independence between documents. However the learning or inference process works by using the documents as input and learning the parameter vector θ, ϕ and the set of clusters $\mathcal{C}$. The corresponding posterior distribution obtained using Bayes rule is given by

$$p(\theta, \phi, \mathcal{C}|d, \alpha, \beta) \;=\; \frac{p(\theta, \phi, \mathcal{C}, d|\alpha, \beta)}{p(d|\alpha, \beta)}$$

Unfortunately exact computation of this quantity is not possible; the basic difficulty is associated with the denominator which has a coupling between θ and β. There are several approximations suggested in the literature; we consider a popular scheme based on Gibbs sampling.

7.7.1 Gibbs Sampling based LDA

It is based on the observation that the complete cluster structure can be derived if one knows the assignment of a cluster/topic given a word. Specifically it needs to compute the probability of a cluster C_i being relevant for a term t_i given all other words being relevant to all other clusters. Formally, this probability is

$$p(C_i|C_{-i}, \alpha, \beta, d)$$

where C_{-i} means all cluster associations other than C_i. We can show using Bayes rule

$$p(C_i|C_{-i}, \alpha, \beta, d) \;=\; \frac{p(C_i, C_{-i}, d|\alpha, \beta)}{p(C_{-i}, d|\alpha, \beta)}$$

where the denominator is independent of C_i; so, it can be ignored but for a scaling factor and the expression may be written as

$$p(C_i|C_{-i}, \alpha, \beta, d) \propto p(C_i, C_{-i}, d|\alpha, \beta) = p(\mathcal{C}, d|\alpha, \beta).$$

Note that C_i and C_{-i} together correspond to $\mathcal{C}$ which is used in simplifying the expression to get $p(\mathcal{C}, d|\alpha, \beta)$.

We can get the required marginal as

$$p(d, \mathcal{C}|\alpha.\beta) = \int\int p(d, \mathcal{C}, \theta, \phi|\alpha, \beta) d\theta d\phi.$$

The joint distribution can be simplified as

$$p(d, \mathcal{C}|\alpha, \beta) = \int\int p(\phi|\beta)p(\theta|\alpha)p(\mathcal{C}|\theta)p(d|\phi_{\mathcal{C}})d\theta d\phi$$

We could separate terms having dependent variables to write the above equation as a product of two integrals.

$$p(d, \mathcal{C}|\alpha, \beta) = \int p(\theta|\alpha)p(\mathcal{C}|\theta)d\theta \int p(d|\phi_{\mathcal{C}})p(\phi|\beta)d\phi$$

Both the integrands have Dirichlet priors in the form of $p(\theta|\alpha)$ and $p(\phi|\beta)$; also there are multinomial terms, one in each integrand. The conjugacy of Dirichlet to the multinomial helps us here to simplify the product to a Dirichlet with appropriate parameter setting. So, it is possible to get the result as

$$p(\mathcal{D}, \mathcal{C}|\alpha, \beta) = \prod_{i=1}^{n} \frac{B(\alpha)}{B(n_{d_i,\cdot} + \alpha)} \prod_{k=1}^{K} \frac{B(\beta)}{B(n_{\cdot,k} + \beta)}$$

where $n_{d_i,k}$ is the number of terms in document d_i that are associated with cluster C_k. There are n documents in the set D. A '.' indicates summing over the corresponding index; for example, $n_{d_i,\cdot}$ is the total number of terms in d_i and $n_{\cdot,k}$ is the number of terms in the entire collection that are associated with C_k. It is possible to use the above equation to get the estimate for $p(C_k|\mathcal{C}_{-k}, D, \alpha, \beta)$ as

$$\begin{aligned} p(C_m|\mathcal{C}_{-m}, D, \alpha, \beta) &= \frac{p(\mathcal{D}, \mathcal{C}|\alpha, \beta)}{p(\mathcal{D}, \mathcal{C}_{-m}|\alpha, \beta)} \\ &\propto (n_{d,k}^{(-m)} + \alpha_k) \frac{n_{t,k}^{-(m)} + \beta_t}{\sum\limits_{t'} n_{t',k}^{(-m)} + \beta_{t'}} \end{aligned}$$

Here the superscript $(-m)$ corresponds to not using the m^{th} token in counting; note that $n_{d,k}$ and $n_{t,k}$ are the counts where k is the topic, d is the document, and t and t' are terms. So, Gibbs sampling based LDA essentially maintains various counters to store these count values. Basically, it randomly initializes K clusters/topics and iterates in updating the probabilities specified by the above equation which employs various counters and also in every iteration the counters are suitably updated.

7.8 Concept and Topic

Another related notion that is used in clustering is *concept*. A framework called conceptual clustering is a prominent soft clustering paradigm that employs classical logic and its variants to describe clusters. For example, a cricket ball could be described using $(colour = red \vee white) \wedge (make = leather) \wedge (dia = medium) \wedge (shape = sphere)$ which is a conjunction of internal disjunctions (or predicates). Each of the conjuncts is viewed as a concept that describes a set of objects; for example, $(colour = red)$ describes a set of red colored objects. This framework is called *conjunctive conceptual clustering*. This is also related to frequent itemset based clustering and classification.

For example consider patterns of 7 and 1 shown in Table 9 wherein the left part is character 7 and in the right part character 1 is depicted; each is in the form of a 3×3 binary matrix (image). In a realistic scenario the size of the matrix could be larger and the number of 1 pixels will also be larger; possibly there could be noise affecting the pixel values mutating a zero by a one and a one by a zero. If there are n such character patterns

1 1 1	1 0 0
0 0 1	1 0 0
0 0 1	1 0 0
7	1

Table 9: Printed characters of 7 and 1

with half of them from class 7 and the other half from class 1, then it is

possible to describe the classes as follows. Consider the 9 pixels in the 3x3 matrix and label them as $i_1,\ i_2,\ \cdots, i_9$ in a row-major fashion. Then the frequent itemset corresponding to 7 is $i_1 \wedge i_2 \wedge i_3 \wedge i_6 \wedge i_9$ because this pattern is present in half the number of characters given and the frequent itemset in class 1 is $i_1 \wedge i_4 \wedge i_7$. So, it is possible to have the following classification rules where the antecedent of the rule is the frequent itemset and the consequent is the Class Label:

- Character 1: $i_1 \wedge i_4 \wedge i_7 \ \rightarrow Class1$
- Character 7: $i_1 \wedge i_2 \wedge i_3 \wedge i_6 \wedge i_9 \ \rightarrow Class7$

Equivalently one can describe these conjunctive concepts as shown in Table 10. Note the similarity between concept which describes a cluster or a

Concept	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9
Class 1	1	0	0	1	0	0	1	0	0
Class 7	1	1	1	0	0	1	0	0	1

Table 10: Description of the Concepts

class by selecting a subset of items or features and topic which assigns a probability to each item or feature; both describe clusters. In the example shown in Table 10 both the class descriptions (concepts) select item i_1 indicating softness.

8 Research Ideas

1. Derive an expression for the number of soft clusterings of n patterns into K soft clusters.
2. Discuss Rough Fuzzy Clustering in the case of leader and what can happen to other algorithms.

   Relevant References:

(a) S. Asharaf, and M. Narasimha Murty, An Adaptive Rough Fuzzy Single Pass Algorithm for Clustering Large Data Sets, Pattern Recognition, 36: 3015-3018, 2003.

(b) V. Suresh Babu and P. Viswanath. Rough-fuzzy weighted k-nearest leader classifier for large data sets, Pattern Recognition, Vol. 42, pp. 17191731, 2009.

(c) P. Maji, and S. Paul, Rough-fuzzy clustering for grouping functionally similar genes from microarray data, IEEE/ACM Trans Comput Biol Bioinform., 10:286-299, 2013.

3. A difficulty with the use of GAs is that they are not scalable. The problem gets complicated further when one considers multi-objective optimization. How to design scalable GAs?

   Relevant References:

   (a) A. Kink, D. Coit, and A. Smith, Multi-Objective Optimization using Genetic Algorithms: A Tutorial, Reliability Engineering and System Safety, 91(9): 992-1007, 2006.

   (b) K. Amours, Multi-Objective Optimization using Genetic Algorithms, Master's thesis, Jinking University, 2012.

   (c) D. Yumin, X. Shufen, J. Fanghua, and L. Jinhai, Research and Application on a Novel Clustering Algorithm of Quantum Optimization in Server Load Balancing, Mathematical Problems in Engineering, Vol. 2014, 2014.

4. Discuss the possibilities of abstracting soft clustering using the string-of-centroids. Can we extend the string-of-group-representation to realize a soft partition? How?

   Relevant References:

   (a) Ujjwal Maulik, and Sanghamitra Bandyopadhyay: Genetic algorithm-based clustering technique, Pattern Recognition, 33: 1455-1465, 2000.

   (b) M. N. Murty, Clustering Large Data Sets, in Soft Computing Approach to Pattern Recognition and Image Processing, pp. 41-63,

edited by A. Ghosh and S. K. Pal, World-Scientific, New Jersey, 2002.

(c) Lin Zhu, Longbing Cao, and Jie Yang, Multiobjective evolutionary algorithm-based soft subspace clustering, Proc. of IEEE Congress on Evolutionary Computation, 2012.

5. Expectation Maximization (EM) is a probabilistic version of the K-Means algorithm. Why did EM became so popular in Machine Learning?

Relevant References:

(a) Thomas Hoffman, Latent Semantic Models for Collaborative Filtering, ACM Trans. on Inf. Syst. 22(1):89-115, 2004.

(b) D. Sontag, and D. M. Roy, Complexity of Inference in Latent Dirichlet Allocation, Proc. of NIPS, 2011.

(c) S-K Ng, Recent developments in expectation-maximization methods for analyzing complex data, Wiley Interdisciplinary Reviews: Computational Statistics, Vol. 5, pp. 415-431, 2013.

6. Matrix factorization is useful in clustering. It is possible to show equivalence between PLSA and NMF; similarly between K-Means and NMF. Is it possible to unify clustering algorithms through matrix factorization?

Relevant References:

(a) Arghya Roy Chaudhuri, and M. Narasimha Murty, On the relation between K-means and PLSA, In Proc. of ICPR, 2012.

(b) Chris Ding, Tao Li, and Wei Peng, On the equivalence between Non-negative Matrix Factorization and Probabilistic Latent Semantic Indexing, Computational Statistics and Data Analysis, 52:39133927, 2008.

(c) Jingu Kim, and Haesun Park, Sparse Nonnegative Matrix Factorization for Clustering, Tech. Report, Georgia Tech., GT-CSE-08-01.pdf, 2008.

7. Is it possible to view clustering based on frequent itemsets as a matrix factorization problem?

   Relevant References:

   (a) Benjamin C. M. Fung, Ke Wang, and Martin Ester, Hierarchical Document Clustering using Frequent Itemsets, In Proc. of SDM, 2003.

   (b) Kiran G.V.R., Ravi Shankar, and Vikram Pudi, Frequent Itemset Based Hierarchical Document Clustering Using Wikipedia as External Knowledge, in Proc. of Knowledge-Based and Intelligent Information and Engineering Systems, Lecture Notes in Computer Science Volume 6277, 2010.

   (c) J. Leskovec, A. Rajaraman, and J. D. Ullman, Mining Massive Datasets,
   infolab.stanford.edu/ ullman/mmds/book.pdf, 2014.

8. Latent Dirichlet Allocation employs Dirichlet and multinomial conjugate pair of distributions. Is it possible to use other distributions?

   Relevant References:

   (a) David M. Blei, Probabilistic topic models, Commun. ACM, 55: 77-84, 2012.

   (b) David Newman, Edwin V. Bonilla, and Wray L. Buntine, Improving Topic Coherence with Regularized Topic Models, In Proc. of NIPS, 2011.

   (c) Hanna M. Wallach, David M. Mimno, Andrew McCallum: Rethinking LDA: Why Priors Matter, In Proc. of NIPS, 2009.