Initialization of Iterative Refinement Clustering Algorithms

Usama Fayyad, Cory Reina, and P.S. Bradley

Microsoft Research
Redmond, WA 98052, USA
{Fayyad,CoryR}@microsoft.com, Paulb@cs.wisconsin.edu

Abstract

Iterative refinement clustering algorithms (e.g. K-Means, EM) converge to one of numerous local minima. It is known that they are especially sensitive to initial conditions. We present a procedure for computing a refined starting condition from a given initial one that is based on an efficient technique for estimating the modes of a distribution. The refined initial starting condition leads to convergence to "better" local minima. The procedure is applicable to a wide class of clustering algorithms for both discrete and continuous data. We demonstrate the application of this method to the Expectation Maximization (EM) clustering algorithm and show that refined initial points indeed lead to improved solutions. Refinement run time is considerably lower than the time required to cluster the full database. The method is scalable and can be coupled with a scalable clustering algorithm to address the large-scale clustering in data mining.

1 Background

Clustering has been formulated in various ways in the machine learning [F87], pattern recognition [DH73,F90], optimization [BMS97,SI84], and statistics literature [KR89,BR93,B95,S92,S86]. The fundamental clustering problem is that of grouping together data items which are similar to each other. The most general approach to clustering is to view it as a density estimation problem [S86, S92,BR93]. We assume that in addition to the observed variables for each data item, there is a *hidden*, unobserved variable indicating the "cluster membership" of the given data item. Hence the data is assumed to arrive from a *mixture model* and the mixing labels (cluster identifiers) are hidden. In general, a mixture model M having K clusters C_i, $i=1,\ldots,K$, assigns a probability to a data point x as follows:

$\Pr(x \mid M) = \sum_{i=1}^{K} W_i \cdot \Pr(x \mid C_i, M)$ where W_i are called the mixture weights. Many methods assume that the number of clusters K is known or given as input.

The clustering optimization problem is that of finding parameters associated with the mixture model M (W_i and parameters of components C_i) to maximize the likelihood of the data given the model. The probability distribution specified by each cluster can take any form. The EM (Expectation Maximization) algorithm [DLR77, CS96] is a well-known technique for estimating the parameters in the general case. It finds locally optimal solutions maximizing the likelihood of the data. Maximum likelihood mixture model parameters are computed iteratively by EM:

1. Initialize the mixture model parameters, producing a current model,
2. Compute posterior probabilities of data items, assuming the current model (E-Step)
3. Re-estimate model parameters based on posterior probabilites from 2, producing new model, (M-Step)
4. If current and new model are sufficiently close, terminate, else go to 2.

We focus on the initialization step 1. Given the initial condition of step 1, the algorithms define a deterministic mapping from initial point to solution. EM converges finitely to a point (set of parameter values) that is locally maximal for the likelihood of the data given the model. The deterministic mapping means the locally optimal solution is sensitive to the initial point choice.

We shall assume a model that represents a mixture of Gaussains. Associated with each data point $\boldsymbol{x}$ is the "weight" or posterior probability that $\boldsymbol{x}$ was generated by mixture component l, $p(\boldsymbol{x}|l)$. We focus on mixture models in which individual component densities are multi-variate Gaussians:

$$p(\mathbf{x} \mid l) = \frac{1}{(2\pi)^{d/2}\left|\Sigma^l\right|^{1/2}} \exp\left\{-\frac{1}{2}\left(\mathbf{x}-\mu^l\right)^T\left(\Sigma^l\right)^{-1}\left(\mathbf{x}-\mu^l\right)\right\}$$ where μ^l is d-dimensional mean and Σ^l is dxd covariance matrix.

There is little prior work on initialization methods for clustering. According to [DH73] (p. 228):

> "One question that plagues all hill-climbing procedures is the choice of the starting point. Unfortunately, there is no simple, universally good solution to this problem."

"Repetition with different random selections" [DH73] appears to be the *defacto* method. Most presentations do not address the issue of initialization or assume either user-provided or randomly chosen starting points [DH73,R92, KR89]. A recursive method for initializing the means by running K clustering problems is mentioned in [DH73] for K-Means. A variant consists of taking the mean of the entire data and then randomly perturbing it K times [TMCH97]. This method does not appear to be better than random initialization in the case of EM over discrete data [MH98]. In [BMS97], the values of initial means along any one of the d coordinate axes is determined by selecting the K densest "bins" along that coordinate. Methods to initialize EM include K-Means solutions, hierarchical agglomerative clustering (HAC) [DH73,R92,MH98] and "marginal+noise" [TMCH97]. It was found that for EM over discrete data initialized with either HAC or "marginal+noise" showed no improvement over random initialization [MH98].

2 Refining Initial Conditions

We address the problem of initializing a general clustering algorithm, but limit our presentation of results to EM. Since no good method for initialization exists [MH98], we

Copyright © 1998, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

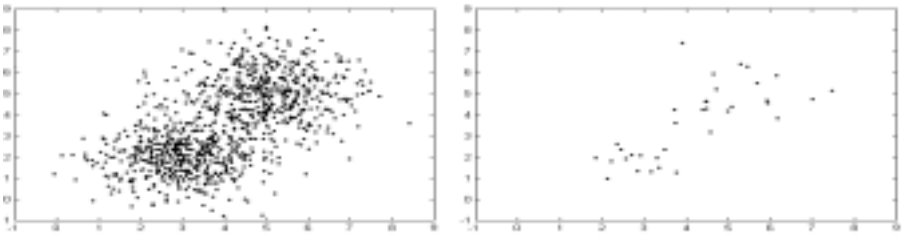


Figure 1. Two Gaussian bumps in 2-d: full sample versus small subsample.

compare against the *defacto* standard method for initialization: randomly choosing an initial starting point. The method can be applied to any starting point provided.

A solution of the clustering problem is a parameterization of each cluster model. This parameterization can be performed by determining the *modes* (maxima) of the joint probability density of the data and placing a cluster centroid at each mode. Hence one clustering approach is to estimate the density and attempt to find the maxima ("bumps") of the

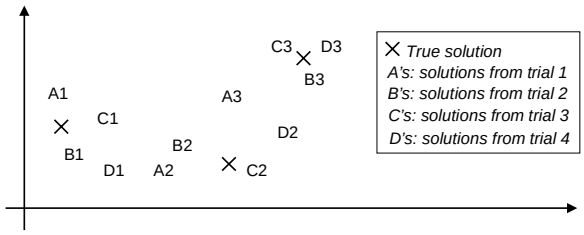


Figure 2. Multiple Solutions from Multiple Samples.

estimated density function. Density estimation in high dimensions is difficult [S92], as is bump hunting [F90]. We propose a method, inspired by this procedure that refines the initial point to a point likely to be closer to the modes. The challenge is to perform refinement efficiently.

The basic heuristic is that severely subsampling the data will naturally bias the sample to representatives "near" the modes. In general, one cannot guard against the possibility of points from the tails appearing in the subsample. We have to overcome the problem that the estimate is fairly unstable due to elements of the tails appearing in the sample. Figure 1 shows data drawn from a mixture of two Gaussians (clusters) in 2-D with means at [3,2] and [5,5]. On the left is the full data set, on the right a small subsample is shown, providing information on the modes of the joint probability density function. Each of the points on the right may be thought of as a "guess" at the possible location of a mode in the underlying distribution. The estimates are fairly varied (unstable), but they certainly exhibit "expected" behavior. Worthy of note here is that good separation between the two clusters is achieved. This observation indicates that the solutions obtained by clustering over a small subsample may provide good refined initial estimates of the true means, or centroids, in the data

Clustering Clusters

To overcome the problem of noisy estimates, we employ secondary clustering. Multiple subsamples, say *J*, are drawn and clustered independently producing *J* estimates of the true cluster locations. To avoid the noise associated with each of the *J* solution estimates, we employ a "smoothing" procedure. However, to "best" perform this smoothing, one needs to solve the problem of grouping the *K*J* points (*J* solutions, each having *K* clusters) into *K* groups in an "optimal" fashion. Figure 2 shows 4 solutions obtained for *K*=3, *J*=4. The "true" cluster means are depicted by "X". The A's show the 3 points obtained from the first subsample, B's second, C's third, and D's fourth. The problem then is determining that D1 is to be grouped with A1 but A2, not with {A1, B1, C1, D1}.

The Refinement Algorithm

The refinement algorithm initially chooses *J* small random sub-samples of the data, S_i, $i=1,\ldots,J$. The sub-samples are clustered via EM with the proviso that empty clusters at termination will have their initial centers re-assigned and the sub-sample will be re-clustered. The sets CM_i , $i=1,\ldots,J$ are these clustering solutions over the sub-samples which form the set *CM*. *CM* is then clustered via K-Means initialized with CM_i producing a solution FM_i. The refined initial point is then chosen as the FM_i having minimal distortion over the set *CM*. Note that this secondary clustering is a K-means clustering and not EM. The reason is that the goal here is to cluster solutions in a hard fashion to solve the corresondence problem. Other procedures could be used for secondary clustering including hierarchical agglomerative clustering. Clustering *CM* is a smoothing over the CM_i to avoid solutions "corrupted" by outliers included in the sub-sample S_i. The refinement algorithm takes as input: *SP* (initial starting point), *Data*, *K*, and *J* (number of small subsamples to be taken from *Data*):

Algorithm Refine(*SP, Data, K, J*)

CM = ϕ

For i=1,…,*J*

Let S_i be a small random subsample of *Data*

Let CM_i = EM_Mod(*SP*, S_i, *K*)

$CM = CM \cup CM_i$

FMS = ϕ

For i=1,…,*J*

Let FM_i = KMeans(CM_i, *CM*, *K*)

Let $FMS = FMS \cup FM_i$

Let $FM = \underset{FM_i}{ArgMax}\{Likelihood(FM_i, CM)\}$

Return (*FM*)

We define the following functions called by the refinement algorithm: KMeans( ), EM_Mod( ) and Likelihood( ). *KMeans* is simply a call to the classic K-Means algorithm taking: an initial starting point, dataset and the number of clusters *K*, returning a set of *K d*-dimensional vectors, the estimates of the centroids of the *K* clusters. *EM_Mod* takes the same arguments as KMeans (above) and performs the same iterative procedure as classic EM except for the following slight modification. When classic EM has converged, the *K* clusters are checked for membership. If any of the *K* clusters have no membership (which often happens when clustering over small subsamples), the corresponding initial estimates of the empty cluster centroids are set to data elements which have least likelihood given the current model, and classic EM is called again from these new initial centriods.

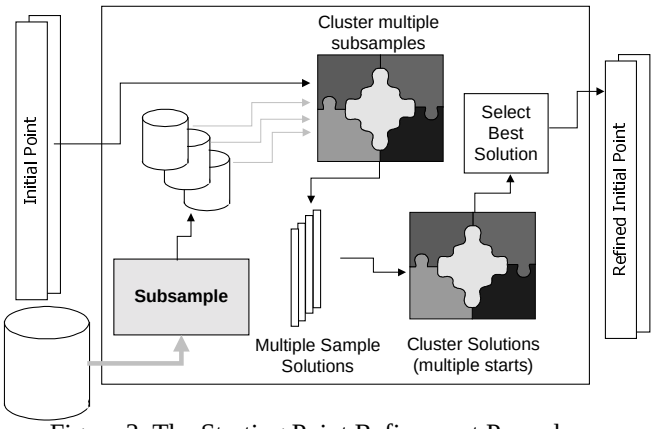


Figure 3. The Starting Point Refinement Procedure

The heuristic re-assignment is motivated by the following: if, at termination of EM, there are empty clusters then reassigning all empty clusters to points with least likelihood, maximizes likelihood the most at this step. *Likelihood* takes set of K estimates of cluster parameters (the means and covariances) and the data set and computes the likelihood of the data set given the model. This scalar measures the degree of fit of a set of clusters to the dataset. The EM algorithm terminates at a solution which is locally optimal for this likelihood function [B95,DLR77,CS96]. The refinement process is illustrated in the diagram of Figure 3.

3 Scalability to Large Databases

The refinement algorithm is primarily intended to work on large databases. When working over small datasets (e.g. most data sets in the Irvine Repository), applying the classic EM algorithm from many different starting points is a feasible option. However, as database size increases (especially in dimensionality), efficient and accurate initialization becomes critical. A clustering session on a data set with many dimensions and tens of thousands or millions of records can take hours to days. In [BFR98], we present a method for scaling clustering to very large databases, specifically targeted at databases not fitting in RAM. We show that accurate clustering can be achieved with improved results over a sampling based approach [BFR98]. Scalable clustering methods obviously benefit from better initialization.

Since our method works on very small samples of the data, the initialization is fast. For example, if we use sample sizes of 1% (or less) of the full dataset size, trials over 10 samples can be run in time complexity that is less than 10% of the time needed for clustering the full database. For very large databases, the initial sample becomes negligible in size.

If, for a data set D, a clustering algorithm requires $Iter(D)$ iterations to cluster it, then time complexity is $|D| * Iter(D)$. A small subsample $S \subseteq D$, where $|S| << |D|$, typically requires significantly fewer iteration to cluster. Empirically, it is reasonable to expect that $Iter(S) < Iter(D)$. Hence, given a specified budget of time that a user allocates to the refinement process, we simply determine the number J of subsamples to use in the refinement process. When $|D|$ is very large, and $|S|$ is a small proportion of $|D|$, refinement time is essentially negligible, even for large J.

Another desirable property of the refinement algorithm is that it easily scales to very large databases. The only memory requirement is to hold a small subsample in RAM. In the secondary clustering stage, only the solutions obtained from the J subsamples need to be held in RAM. Note we assume that it is possible to obtain a *random sample* from a large database. In reality this can be a challenging task. Unless one can guarantee that the records in a database are not ordered by some property, random sampling can be as expensive as scanning the entire database (using some scheme such as reservoir sampling, e.g. [J62]). Note that in a database environment a data view may not exist as a physical table. The result of a query may involve joins, groupings, and sorts. In many cases database operations impose a special ordering on the result set, and "randomness" of the resulting *database view* cannot be assumed in general.

4 Experiments on Synthetic Data

Synthetic data was created for dimension d = 2, 3, 4, 5, 10, 20, 40, 50 and 100. For a given value of d, data was sampled from 10 Gaussians (hence K=10) with elements of their mean vectors (the true means) μ sampled from a uniform distribution on [-5,5]. Elements of the diagonal covariance matrices Σ were sampled from a uniform distribution on [0.7,1.5]. The number of data points sampled was chosen as 20 times the number of model parameters. The K=10 Gaussians were not evenly weighted.

The goal of this experiment is to evaluate how close the means estimated by classic EM are to the true Gaussian means generating the synthetic data. We compare 3 initializations:

1. No Refinement: random starting point chosen uniformly on the range of the data.

2. Refinement (J=10, 1%): a starting point refined from (1) using our method. Size of the random subsamples: 1% of full dataset , the number of subsamples: 10.

3. Refinement (J=10, 5%): same as (2) but subsample of size 5%.

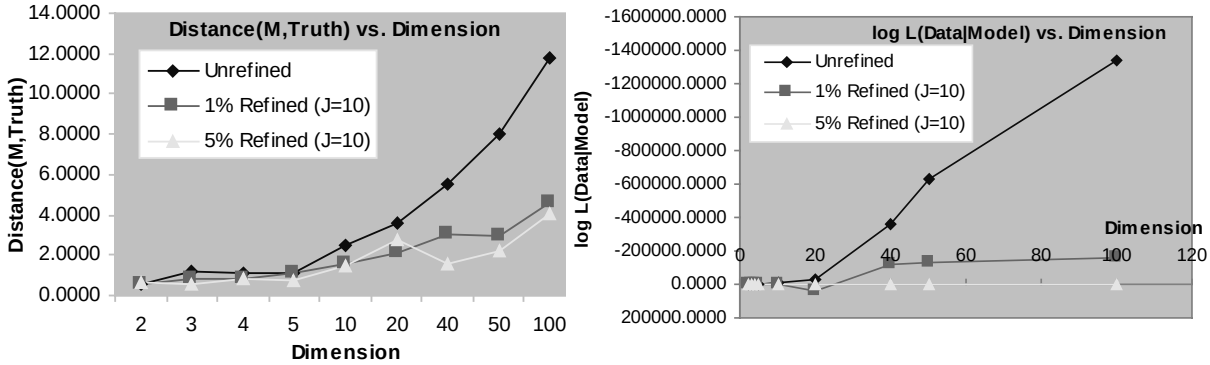


Figure 5. Comparing performance as dimensionality increases

Once classic EM has computed a solution over the full dataset from any of the 3 initial points described above, the estimated means must be matched with the true Gaussian means in some optimal way prior to computing the distance between these estimated means the true Gaussian means. Let $\mu^l, l = 1,\ldots,K$ be the K true Gaussian means and let $\bar{x}^l, l = 1,\ldots,K$ be the K means estimated EM over the full dataset. A "permutation" π is determined so that the following quantity is minimized: $\sum_{l=1}^{K} \left\| \mu^l - \bar{x}^{\pi(l)} \right\|_2$. The "score" for a solution computed by classic EM over the full dataset is simply the above quantity divided by K. This is the average distance between the true Gaussian means and those estimated by EM from a given initial starting point.

Results

Figure 5 summarizes results averaged over 10 random initial points determined uniformly on the range of the data. Note that the EM solution computed from refined initial points is consistently nearer to the true Gaussian means generating the dataset than the EM solution computed from the original initial points. On the left we summarize average distance to the true Gaussian means. On the right we show corresponding log-likelihood of the data given the model. For low dimensions, effect of initialization is not as strong as it is for high dimensions, as expected. Sampling at 5% produces better refinement, but of course costs more. With non-synthetic data, sampling more sometimes results in worse solution as effect of "finding modes by subsampling" is reduced.

5 Results on Public Data

We present computational results on two publicly available "real-world" datasets. We are primarily more interested in *large databases* -- hundreds of dimensions and tens of thousands to millions or records. It is for these data sets that our method exhibits the greatest value. The reason is simple: a clustering session on a large database is a time-consuming affair. Hence a refined starting condition can insure that the time investment pays off.

To illustrate this, we used a large publicly available data set available from Reuters News Service. We also wanted to demonstrate the refinement procedure using data sets from the UCI Machine Learning Repository[1]. For the most part, we found that these data sets are *too easy*: they are low dimensional and have a very small number of records. With a small number of records, it is feasible to perform multiple restarts efficiently. Since the sample size is small to begin with, sub-sampling for initialization is not effective. Hence most of these data sets are not of interest to us. Nevertheless, we report on our general experience with them as well as detailed experience with one of these data sets to illustrate that the method we advocate is useful when applied to smaller data sets. We emphasize, however, that our refinement procedure is best suited for large-scale data. The refinement algorithm operates over small sub-samples of the database and hence run-times needed to determine a "good" initial starting point (which speeds the convergence on the full data set) are orders of magnitude less than the total time needed for clustering in a large-scale situation. We note that it is very likely that the cluster labeling associated with many real-world databases do not correspond to the clusters assigned by EM (whose objective is to maximize likelihood). So evaluation of results on such data is not as easy as synthetic data where truth is known.

5.1 Datasets from UCI ML Repository

We evaluated our method on several Irvine data sets. First we discuss one set, then general comments on others.

Image Segmentation Data Set

This data set consists of 2310 data elements in 19 dimensions. Instances are drawn randomly from a database of 7 outdoor images (brickface, sky, foliage, cement, window, path, grass). Each of the 7 images is represented by 330 instances.

Random initial starting points were computed by sampling uniformly over the range of the data. We compare solutions achieved by the classic EM algorithm starting from: 1) random initial starting points, and 2) initial points refined by our method. A best measure in this case is to report the log-likelihood of the data given the extracted model. The results are as follows:

Refinement Method	Log Likelihood	% increase
None	-123857	45.6%
1% Refined (J=10)	-85577	1%
5% Refined (J=10)	**-85054**	**0**

Other Irvine Datasets

We evaluated the refinement procedure on other data sets such as Fisher's IRIS, Star-Galaxy-Bright, etc. Because these data sets are very low dimensional and their sizes small, the majority of the results were of no interest.

Clustering these data sets from random initial points and from refined initial points led to approximately equal gain in entropy and equal distortion measures in most cases. We did observe, however, that *when a random starting point leads to a "bad" solution, then refinement indeed takes it to a "good" solution.* So in those (admittedly rare) cases, refinement does provide expected improvement. We use the *Reuters* information retrieval data set to demonstrate our method on a real and difficult clustering task.

5.2 Reuters Information Retrieval Data Set

The Reuters text classification database is derived from the original Reuters-21578 data set made publicly available as part of the Reuters Corpus, through available as part of the Reuters Corpus, through Reuters, Inc., Carnegie Group and David Lewis[2]. This data consists of 12,902 documents. Each document is a news article about some topic: e.g. *earnings*, *commodities*, *acquisitions*, *grain*, *copper*, etc… There are 119 categories, which belong to some 25 higher level categories (there is a hierarchy on categories). The Reuters database consists of word counts for each of the 12,902 documents. There are hundreds of thousands of words, but for purposes of our experiments we selected the 302 most frequently occurring words, hence each instance has 302 dimensions indicating the integer number of times the corresponding word occurs in the given document. Each document in the IR-Reuters database has been classified into one or more categories. We use K=25 for clustering

[1] For more details, see the the Irvine ML Data Repository at http://www.ics.uci.edu/~mlearn/MLRepository.html

[2] See: http://www.research.att.com/~lewis/ reuters21578/ README.txt for more details on this data set.

purposes to reflect the 25 top-level categories. The task is then to find the best clustering given K=25.

Reuters Results

For this data set, because clustering the entire database requires a large amount of time, we chose to only evaluate results over 5 randomly chosen starting conditions. Results are shown as follows:

Refinement Method	Δ Log Likelihood
None	-5319570
1% Refined (J=10)	**0**
5% Refined (J=10)	-201116

The chart shows a significant decrease in the log likelihood measure from the best solution. In this case, 1% samples did better than 5%. To normalize results per case simply divide by 12K (size of data). Since each document belongs to a category (there are 119 categories), we can also measure the quality of the achieved by any clustering by measuring the gain in information about the categories that each cluster gives (i.e. pure clusters are informative). The quality of the clusters can be measured by the average category purity in each cluster. In this case the average information gain for the clusters obtained from the refined starting point was 4.13 times higher than the information gain obtained without refining the initial points.

6 Concluding Remarks

A fast and efficient algorithm for refining an initial starting point for a general class of clustering algorithms has been presented. The refinement algorithm operates over small subsamples of a given database, hence requiring a small proportion of the total memory needed to store the full database and making this approach very appealing for large-scale clustering problems. The procedure is motivated by the observation that subsampling can provide guidance regarding the location of the modes of the joint probability density function assumed to have generated the data. By initializing a general clustering algorithm near the modes, not only are the true clusters found more often, but it follows that the clustering algorithm will iterate fewer times prior to convergence. This is very important as the clustering methods discussed here require a full data-scan at each iteration and this may be a costly procedure in a large-scale setting.

We believe that our method's ability to obtain a substantial refinement over randomly chosen starting points is due in large part to our ability to avoid the empty clusters problem that plagues traditional EM. Since during refinement we reset empty clusters to far points and reiterate the EM algorithm, a starting point obtained from our refinement method is less likely to lead the subsequent clustering algorithm to a "bad" solution. Our intuition is confirmed by the empirical results.

The refinement method presented so far has been in the context of the EM. However, we note that the same method is generalizable to other algorithms: an example of this method used to initialize the K-Means algorithm is given in [BF98]. Generalization is possible to discrete data (on which means are not defined). The key insight here is that if some algorithm ClusterA is being used to cluster the data, then ClusterA is also used to cluster the subsamples. The algorithm ClusterA will produce a *model.* The model is essentially described by its parameters. The parameters are in a continuous space. The stage which clusters the clusters (i.e. step 3 of the algorithm Refine in Section 2) remains as is; i.e. we use the K-Means algorithm in this step. The reason for using K-Means is that the goal at this stage is to find the "centroid" of the models, and in this case the *harsh* membership assignment of K-Means is desirable.

References

[BR93] J. Banfield and A. Raftery, "Model-based Gaussian and non-Gaussian Clustering", *Biometrics,* vol. 49: 803-821, pp. 15-34, 1993.

[B95] C. Bishop, 1995. *Neural Networks for Pattern Recognition*. Oxford University Press.

[BMS97] P. S. Bradley, O. L. Mangasarian, and W. N. Street. 1997. "Clustering via Concave Minimization", in *Advances in Neural Information Processing Systems 9*, M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.) pp 368-374, MIT Press, 1997.

[BFR98] P. S. Bradley, U. Fayyad, and C. Reina, "Scaling Clustering Algorithms to Large Databases", *Proc. 4th International Conf. on Knowledge Discovery and Data Mining (KDD-98).* AAAI Press, 1998.

[CS96] P. Cheeseman and J. Stutz, "Bayesian Classification (AutoClass): Theory and Results", in [FPSU96], pp. 153-180. MIT Press, 1996.

[DLR77] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM algorithm". *Journal of the Royal Statistical Society, Series B,* 39(1): 1-38, 1977.

[DH73] R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis. New York: John Wiley and Sons. 1973

[FHS96] U. Fayyad, D. Haussler, and P. Stolorz. "Mining Science Data." *Communications of the ACM* 39(11), 1996.

[BF98] P. S. Bradley and U. Fayyad, "Refining Initial Points for K-Means Clustering", *Proc. 15th International Conf. Machine Learning.* Morgan Kaufmann 1998.

[F87] D. Fisher. "Knowledge Acquisition via Incremental Conceptual Clustering". *Machine Learning*, 2:139-172, 1987.

[F90] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, San Diego, CA: Academic Press, 1990.

[J62] Jones, "A note on sampling from a tape file". *Communications of the ACM*, vol 5, 1962.

[KR89] L. Kaufman and P. Rousseeuw, 1989. Finding Groups in Data, New York: John Wiley and Sons.

[MH98] M. Meila and D. Heckerman, 1998. "An experimental comparison of several clustering methods", *Microsoft Research Technical Report MSR-TR-98-06*, Redmond, WA.

[R92] E. Rasmussen, "Clustering Algorithms", *in Information Retrieval Data Structures and Algorithms*, Frakes and Baeza-Yates (Eds.), pp. 419-442, New Jersey: Prentice Hall, 1992.

[S92] D. W. Scott, *Multivariate Density Estimation*, New York: Wiley. 1992

[SI84] S. Z. Selim and M. A. Ismail, "K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality." *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, No. 1, 1984.

[S86] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, London: Chapman & Hall, 1986.

[TMCH97] B. Thiesson, C. Meek, D. Chickering, and D. Heckerman, 1997. "Learning Mixtures of Bayesian Networks", *Microsoft Research Technical Report TR-97-30*, Redmond, WA.