A PRIMER FOR THE PRACTICE OF COMPUTATIONAL INTELLIGENCE

Yoh-han Pao and Wyatt S. Newman

Preface

These notes were prepared to serve in support of lectures and problem exercises in an introductory course on the practice of Computational Intelligence.

It is the conceit of Computational Intelligence that it is committed to dealing with computational tasks of great complexity, not by overpowering them but by being clever about it.

And so there is a base of insights, algorithms and practices that are indeed clever and powerful, often amazingly so. It is useful to understand these matters in a coherent manner so that perhaps deeper insights might be attained and the practice of the paradigm might be efficient and effective. These notes were written in support of the study of such subject matters with a holistic viewpoint in mind.

Some characteristics of Computational Intelligence are that it deals with large bodies of multivariate data, with discerning the presence of complex functional relationships between such bodies of data, with use of knowledge in search for optimal solutions, and with autonomous design of efficient solutions. In execution, it is distinctive in the emphasis it places on learning and adaptation and on the use of networks of simple elemental processors.

The designation of this style of computing and problem solving as Computational Intelligence is more a consequence of common usage than anything else. There is no widespread commitment to the defining of what constitutes 'intelligence' human or otherwise. There is the question of whether the use of the words 'Computational Intelligence' is just another example of the reprehensible practice of mindlessly renaming prior practices for the sake of registering 'advances'. Indeed the science of this body of computational practice does trace back to foundations in Computer Science, Mathematics, Statistics, and to inspirations from Neuroscience, Psychology and Philosophy, but not just to one or the other singly nor in any simple linear manner. And it would seem that there is virtue to providing a label for this collectively forming field of knowledge and activity and these notes are written in support of that viewpoint.

Although there are many interesting and well-written books on aspects of Computational Science, it was difficult to select any single one as the text for an introductory course.

Some tended to explain Computational Intelligence in terms of ‘classical’ results known to that author’s favorite discipline. Some other encyclopedic works are excellent references but lacked depth at focal points.

However it must be admitted that the choice of texts or notes in teaching is a matter of taste and style and not an objective matter.

These present notes deal with some key topics in moderate depth and try to bind them into a holistic structure of Computational Intelligence and try to bind them into a holistic structure as well as try to trace connections to supporting sciences.

A set of basic algorithms is presented in the first ten chapters.

In Chapter 1, the K-Means algorithm is presented and discussed in terms of clustering data points in vector space, for the purpose of self-organization. Chapter 2 describes two related algorithms, these being SOM and ART. There are several different motivations for self-organization and different algorithms may be more appropriate depending on the nature of the task and perhaps also on the nature of the data.

Chapter 3 is concerned with the task of Function Approximation. In its basic form, it can be cast in the form of learning a scalar function defined in vector space. More generally, the task is that of learning a mapping from one vector space to another vector space. In Computational Intelligence practice, this activity is often referred to as Supervised Learning, generalizing a discrete set of associations. The Multi-layer Feedforward Net is an architecture that is widely used for such purposes in conjunction with the Back-Propagation of Error algorithm for learning or training. The details of those two matters are presented in Chapter 4. Matters of generalization, evaluation, and use are presented in Chapter 5.

The Functional-Link perspective, the Radial Function implementation and the Orthogonal Least Squares (OLS) refinement are discussed in Chapter 6 and details of the OLS algorithm are developed in Chapter 7.

The concepts of Associative Memory and Recall, Recurrent Nets and Dynamic Systems are discussed in Chapter 8 in the context of the Hopfield Net, and with reference to Attractors, Basins of Attraction, and Deterministic Chaos.

The use of the Hopfield net for optimization is described in Chapter 9 and illustrated with the problem of finding the optimal path for the Traveling Salesman.

The task of search for optimal or near-optimal solutions is an important aspect of Computational Intelligence practice. This task is discussed in the context of continuous functions and combinatorial optimization in Chapter 10 in the context of the GESA algorithm. GESA is based on the belief that stochastic search does not have to be ‘mindless’ and that Guided Evolutionary search with Simulated Annealing can be effective in locating optimal solutions.

Each chapter is accompanied by a main exercise for the purpose of consolidating ideas.

Chapters 11 and 12 are devoted to discussions of applications of such Computational Intelligence.

Chapter 1 The K-Means Algorithm

1.1 The Algorithm

Given data in the form of P points in N-dimensional space, it is helpful if those data points would automatically organize themselves into K disjoint clusters, each with a cluster center. Such clustering provides means for getting an idea of how the data points are distributed in the N-dimensional data space. Criterion for belonging to a cluster is that the pattern in question is closer, by some measure, to that cluster center than to any of the other centers.

The value of K is specified external to the algorithm and that issue will be discussed separately because the number of clusters is an important consideration in gathering insight from the data.

The data points in N-dimensional space may also be considered to be vectors each of N-components.

A metric is needed for measuring ‘distance’ or closeness. For the purposes of these notes, Euclidean Distance or Scalar Product suffice as examples of metrics. Other measures might be more appropriate for other cases.

The K-Means Algorithm can be described in terms of data structures and functions which operate on these structures.

Data Structures

Pattern: Each pattern is a vector, a one-dimensional ordered array of numeric values lying between +1 and –1. In addition each pattern is identified by a pattern_label, for purposes of identification and by a cluster_label, identifying the cluster to which the pattern belongs.

Therefore each pattern can be described in terms of N+2 items, consisting of a vector of N-components and two id values, as illustrated in the following:

Pattern_label	N Pattern_elements	Cluster_label
pl(p)	x(p 0)…x(p n)…x(p N-1)	k(p)

For a set of P such patterns, it is helpful to consider the following data structure, the Pattern_Set.

Pattern_Set:

Pattern-label	Pattern_elements	Cluster_label
pl(0)	x(0 0), ------------, x(0 n),-----------, x(0 N-1)	k(0)
pl(1)	x(1 0), ------------, x(1 n),-----------, x(1 N-1)	k(1)
----	--- , -------------, --- ,---------- , ---	---
----	--- , -------------, --- ,-----------, ---	---
pl(p)	x(p 0), ------------ , x(p n),---------- , x(p N-1)	k(p)
----	--- , -------------, --- , ---------- , ---	
pl(P-1)	x(P-1 0), ----------, x(P-1 n),---------, x(P-1 N-1)	k(P-1)

For some circumstances the two dimensional array of pattern elements can be considered to be a matrix and can be managed readily as such.
Corresponding data structures can be declared for **Cluster** and for **Cluster_Set**. The latter is illustrated in the following.

Cluster_label	Cluster_center elements	# of patterns in cluster	Pointer to member pattern_labels
cl(0)	c(0 0) --- c(0 n) --- c(0 N-1)	n(0)	*p(0)
cl(1)	c(1 0) --- c(1 n) --- c(1 N-1)	n(1)	*p(1)
cl(2)	c(2 0) --- c(2 n) --- c(2 N-1)		
-	-	-	
-	-	-	
-	-	-	
k	c(k 0) --- c(k n) --- c(k N-1)	n(k)	*p(k)
-	-	-	
-	-	-	
-	-	-	
(K-1)	c(K-1) --- c(K-1 n)---c(K-1 N-1)	n(K-1)	*p(K-1)

These data structures are illustrated for the purpose of providing pictorial images for visualizing the manner in which the algorithm works. Different fields in these structures may be of different data types; those matters are not addressed in this discussion but might have to be taken into account explicitly in some implementations of the algorithm.

In addition to declaring data structures we can define functions which operate on these data structures. The K-Means Algorithm can then be stated schematically as follows.

Algorithm

1) Preprocess

   Scale values of pattern elements {x(p n)} to lie between +1 and –1.

2) Initialize

   Copy the first K rows of x(p n) values into the c(k n) sites, [that is, let the first K patterns be the initial cluster centers].

3) Process Patterns

   Set change counter Del=0
   Starting with the first pattern, with pl(p)=0, process each and all of the patterns in the P pattern data set.
   For pattern **x**, call function **distance** to compute distances from **x** to the cluster centers **c**(k) and find that center k for which the distance is the least.
   Assign pattern **x** to cluster k, the nearest cluster center.

The Euclidean Distance between a pattern vector **x** and a cluster center **c** is

$$d = \left(\sum_n (x_n - c_n)^2 \right)^{1/2}$$

4) Update Data Structures

In the event that pattern **x** is reassigned from cluster k' to cluster k, or is assigned for the first time to a cluster k, call function **update_centers** to update the positions of clusters k and k'. In the event that the pattern had not been processed previously, there is no need to update any k' cluster center.

In the function **update_centers**, we make use of the following relationships for computing the updated values of the cluster centers, k and k', as appropriate.

For updating cluster center k, the 'new cluster center, use

$$c_{kn}(t+1) = \frac{n_k(t)}{n_k(t)+1} c_{kn}(t) + \frac{1}{n_k(t)+1} x_{pn}$$

And the co-ordinates of the 'old' cluster center is updated with use of

$$c_{k'n}(t+1) = \frac{n_{k'}(t)}{n_{k'}(t)-1} c_{k'n}(t) - \frac{1}{n_{k'}(t)-1} x_{pn}$$

5) Update data entries.

Set Del=Del+1 if a change in cluster membership is made, such as assigning a pattern **p** to cluster k
Set Del=Del+2 if two changes are involved, such as moving a pattern **p** from cluster k' to k.
Set n(k)=n(k)+1
Set n(k')=n(k')-1, if warranted
Update list of member patterns in Cluster_Set
Update Cluster_label in Pattern-Set

6) Check termination condition

    After every pattern assignment, check to see if p<=P-2, if yes, proceed to process the next pattern on Pattern_label columnar list.

    If p=P-1, consult value of Del, the count of the number of changes made in assignments on that pass of the set of patterns.

    Exit algorithm if Del=0. Return that portion of the data structure that is of interest.

7) Validate and Refine

    Process the list of patterns as indicated by 'Pattern_label' one more time, to see if a stable set of assignments has been obtained.

    Post-process, as appropriate, by increasing or decreasing the value of K and repeating the algorithm, and by use of a number of different operations depending on the intended use of the results of the K-Means clustering operation.

1.2 *Discussion of Applications, Issues and Refinements*

Applications of the K-Means algorithm fall into two major categories one being the autonomous formation of a description of the distribution of data points in N-dimensional data space and the other being use of such information for categorization, classification and associative recall from memory. The first type of application is generally referred to as self-organization and is truly a learning action carried out entirely in the 'input' space. That so-called 'unsupervised' learning is very useful even if it is terminated at that stage but more often than not the self-organized structure established in input space is 'decorated' with information obtained from an associated 'output' space. For example, if it can be established that all patterns in cluster j in input space are 'positive', then any new data point found to be cluster j might indeed be classified as 'positive'. This type of 'supervised self-organization' can be used to great effect in many different ways in associative memories and search engines, and can be used to enable the learning of rules and autonomous decision making, purely on the basis of experience.

In the following we take a closer look at unsupervised learning in its basic form, to comment on strengths and weaknesses and to mention refinements meant to counter weaknesses or to augment functionality. We also mention another self-organization scheme that of the Self-Organizing-Map (SOM). It is interesting in that it is a grid-point method, uses the scalar product as metric and might even use K-Means for initialization. The SOM algorithm is described in Chapter 2.

With regards to the basic K-Means algorithm itself, the principal question is whether it should be viewed as a tool for cluster analysis of multivariate data or whether it can be used to yield an a priori probability density distribution function for estimating the probability of the occurrence of a data point at **x**. The latter task is very difficult. In a simple form, one might consider trying to represent the distribution of the data points by erecting a normal distribution function at each cluster center and then adjusting the amplitudes of the Gaussian functions until the actual density distribution of data points is replicated satisfactorily, judged by some criterion. But it is this step that is difficult. Not all densities are radially symmetric about a set of cluster centers.

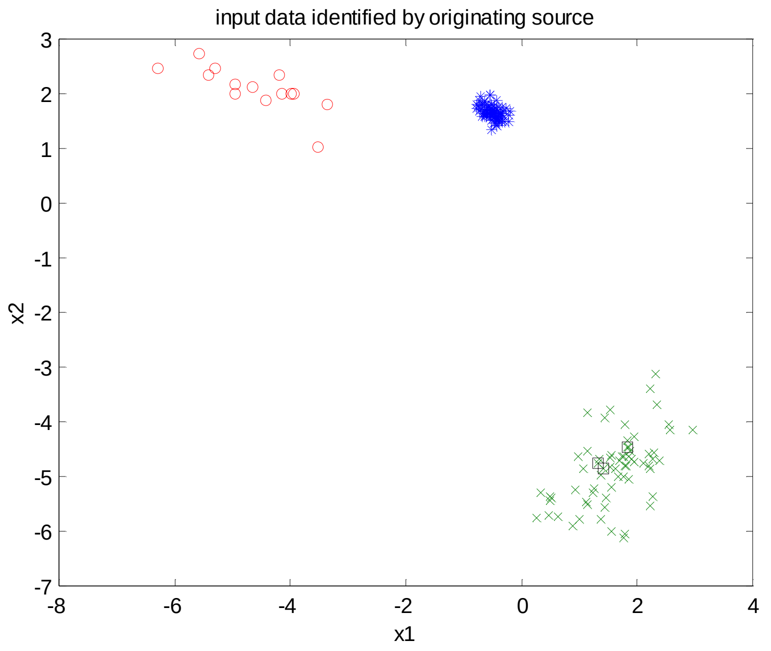

Fig 1.1: Data points originating from 3 separate 2-D Gaussian sources. In this example, the means are relatively widely separated and the variances are relatively low, resulting in obvious clustering of the data.

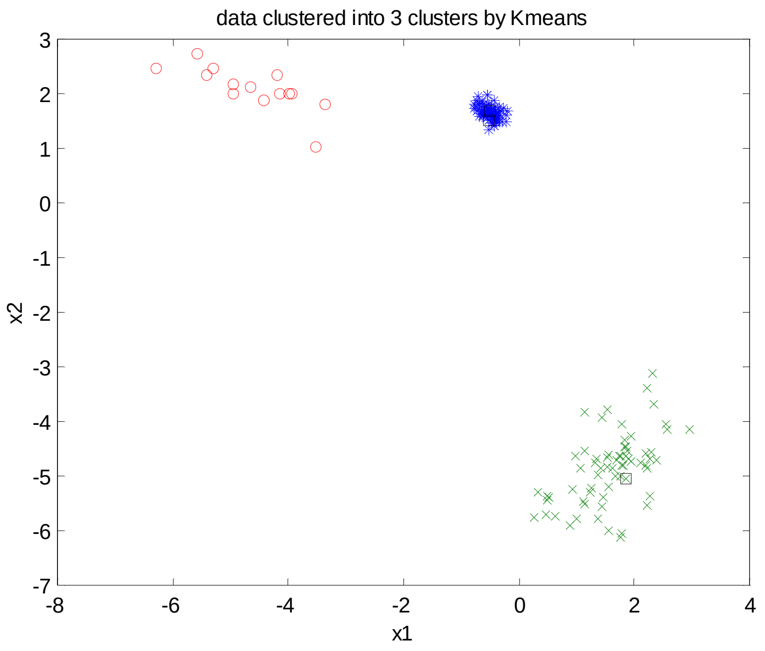

Fig 1.2: An example of Kmeans clustering. Given presumption specification of finding three clusters, the datapoints of Fig 1.1 are clustered in groups corresponding to the originating sources.

The 2-dimensional data points in Figure 1.1 were generated in accordance with normal distribution functions centered about 3 different sites. The underlying statistics of each cluster in 2-D is specified by 5 parameters: mean value of x1, mean value of x2, standard deviation of x1, standard deviation of x2, and the correlation between x1 and x2.

Figure 1.2 shows the result of the Kmeans algorithm operating on the data of Fig 1.1. The number of clusters was (fortuitously) selected to be identical to the actual number of clusters. In general, one can not know in advance how many clusters to assume. In the case of Fig 1.2, the Kmeans algorithm clustered the data into three groups identical to the underlying (hidden) organization of the data sources.

Figure 1.3 shows the result of assuming the existence of four clusters. The result is useful, but somewhat surprising. Intuitively, one would expect a high-variance cluster to be subdivided to create the fourth prescribed cluster. In this example, though, a relatively tightly-packed cluster was subdivided. This result is due to the random choice of initial assumed cluster centers. The first four datapoints were used as the assumed initial cluster centers. In this case, two of the first four datapoints happened to be members of the tightly-packed cluster, resulting in this cluster being subdivided. In spite of this counterintuitive selection of clusters, the result is consistent and useful.

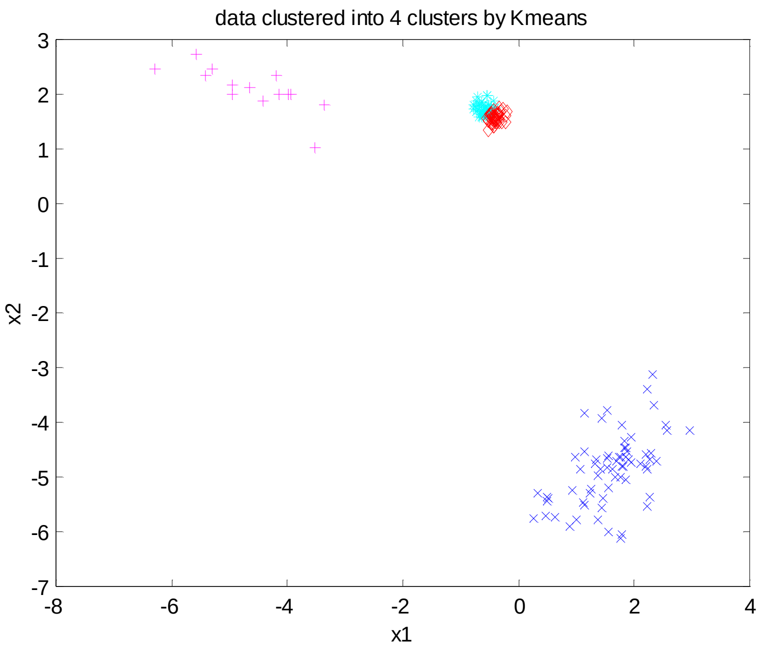

Fig 1.3: Result of a Kmeans clustering of the data of Fig 1.1 assuming 4 clusters. Note that the 4 clusters are still consistent in terms of grouping points near 4 derived centers. However, the centers that emerged were a consequence of the initial seed points. As a result, a relatively tight cluster was separated into two clusters, rather than the more logical choice of subdividing one of the higher-variance clusters.

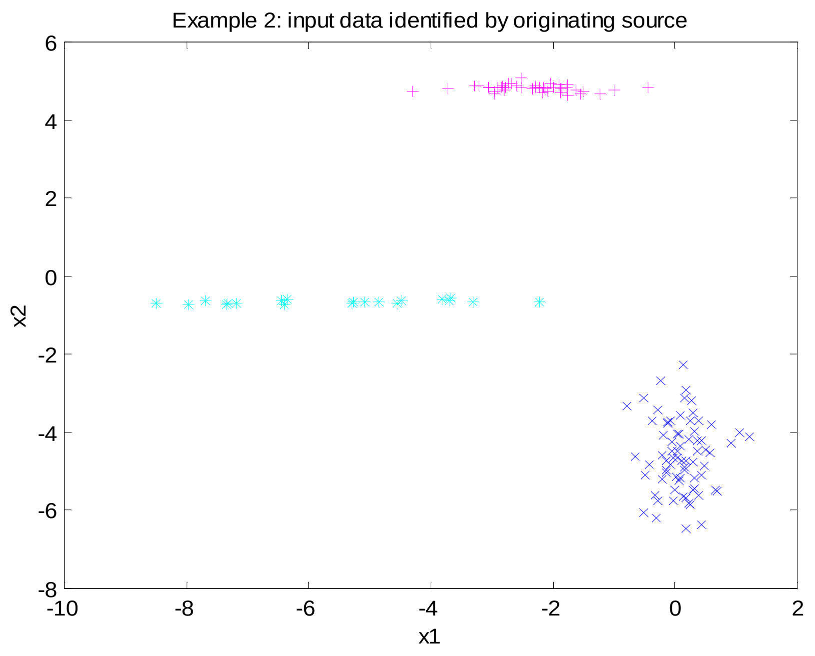

Fig 1.4: Example 2 dataset. Three groups of data originate from three, 2-D Gaussian sources. In this case, two of the clusters have a relatively large variance in the x1 direction. Nonetheless, the natural clustering corresponding to the underlying sources is apparent.

Things are less satisfactory when the data points are distributed in the manner shown in Figure 1.4. In this case, clusters associated with the underlying statistics are visually apparent. However, two of the clusters have a relatively large variance in the x1 direction. As a result, using Euclidian distance within the Kmeans algorithm does not separate the clusters as expected.

The clustering obtained with a K value of 3 is shown in Figure 1.5. In this case, two of the "seed" cluster centers happened to be drawn from the lower-right group of points. As a result, the lower-right cluster was subdivided into two clusters, and the middle and upper clusters were lumped together as a single, high-variance cluster.

Figure 1.6 shows the result of attempting to cluster the second example into four clusters. In this case, all four cluster centers were initialized to points drawn from the lower-right group. The Kmeans algorithm was able to create a new cluster center distinct from the lower-right group, simply by adjusting centroid coordinates and reassigning points to nearest cluster centers. However, the algorithm did not recognize the separation of the top and middle clusters and instead created three low-variance clusters from the lower-right group and a single high-variance cluster from the upper and middle groups.

Figure 1.7 shows a second attempt at organizing the data into four clusters. In this case, the initial cluster centers happened to be drawn from a point within the top cluster, a point within the middle cluster, and two points within the lower cluster. As a result, the Kmeans algorithm successfully separated the top and middle groups from the lower

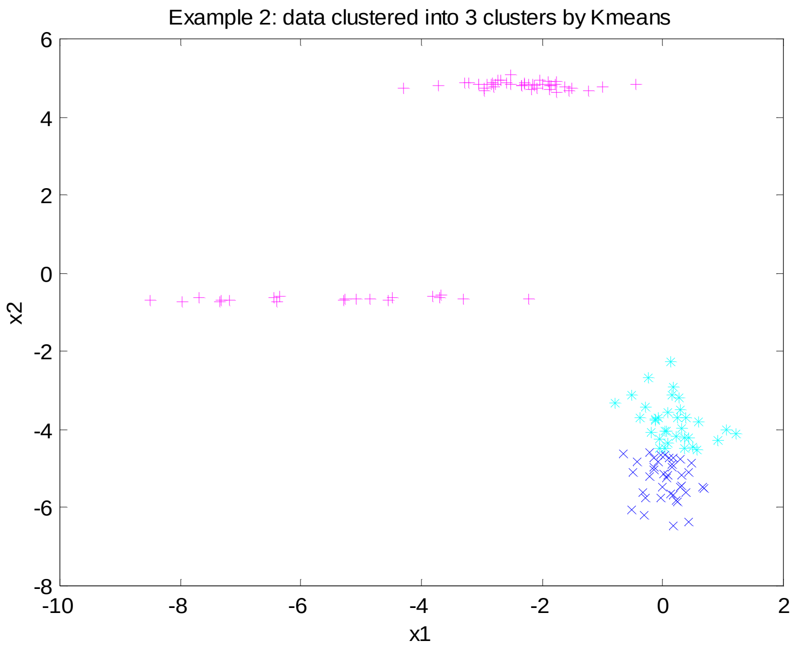

Fig 1.5: Example 2 dataset clustered into 3 clusters by Kmeans. The initial cluster centers were randomly assigned, in this case with two initial centers drawn from the lower-right cluster and one from the top cluster. As a result, the lower cluster was subdivided and the middle cluster was grouped with the top cluster.

group. Although the lower group is subdivided unnecessarily, the clustering is nearly in complete agreement with the originating sources (Fig 1.4). Fig 1.7 includes a lone point from the middle cluster that has been associated by Kmeans with part of the lower cluster.

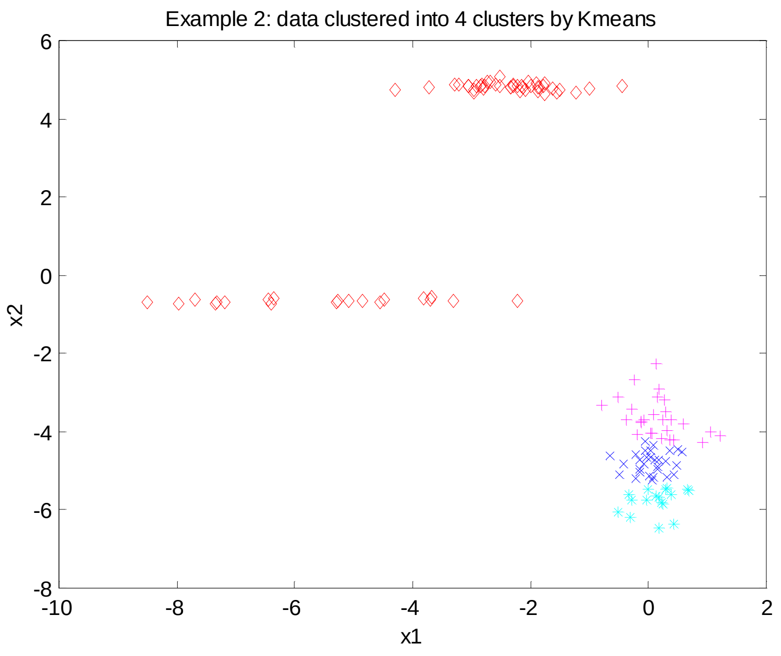

Fig 1.6: Example 2 dataset clustered into 4 clusters by Kmeans. In this case, the initial cluster centers were all drawn from the lower-right group. The Kmeans algorithm was able to derive a new cluster center that separated out the top two clusters from the lower-right cluster. However, it did not recognize how to separate the top two clusters.

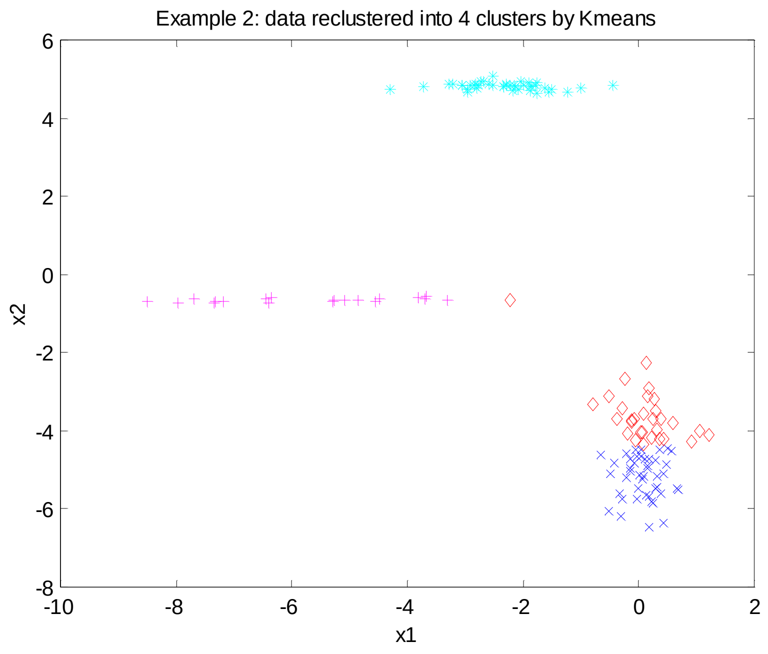

Fig 1.7: Kmeans clustering of example 2 dataset into 4 clusters. In this case, the cluster centers were initialized to a point from the top cluster, a point from the middle cluster and two points from the bottom cluster. The original associations are recovered nearly identically, although the lower cluster is unnecessarily subdivided into two, and a single point from the middle cluster is incorrectly associated with part of the lower cluster.

The situations illustrated here can be visualized and understood readily in 2D space, but matters are more complicated in higher dimensional space. One cannot invoke visualization to declare the number of clusters or to seed the initial cluster centers to encourage a more successful result.

In so far as cluster analysis is concerned, the situation illustrated in Figure 1.4. can be handled with hierarchical clustering. Some simple probing action would reveal that the distribution in the large extended cluster is highly irregular. One would then proceed to build a multi-leveled tree of clusters-- a structure that would correspond more accurately to the actual distribution of data points. In the interpretation of a new data point, classification would proceed from level to level, to more precise cluster identification and to higher resolution in estimating the location of the new data point.

In a hierarchical search, as one proceeds towards the leaf nodes, or leaf clusters, there is the danger of being switched by chance to a wrong branch and not having means for recovery from error. Figure 1.8 shows an example of how the data can mislead the clustering. The original data is comprised of four clusters near the corners of the 2-D region, plus a fifth, single-point cluster near the center of the region. This fifth point is biased slightly towards the upper right corner of the pattern space.

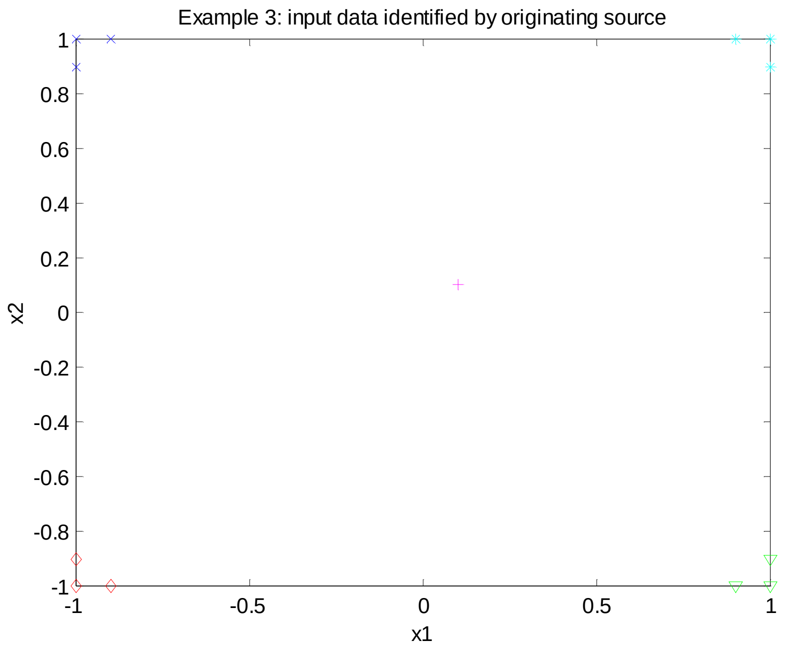

Fig 1.8: Example 3 dataset. Points are clustered near four corners of the region, and an additional point lies near the center of the region, biased slightly towards the upper right corner.

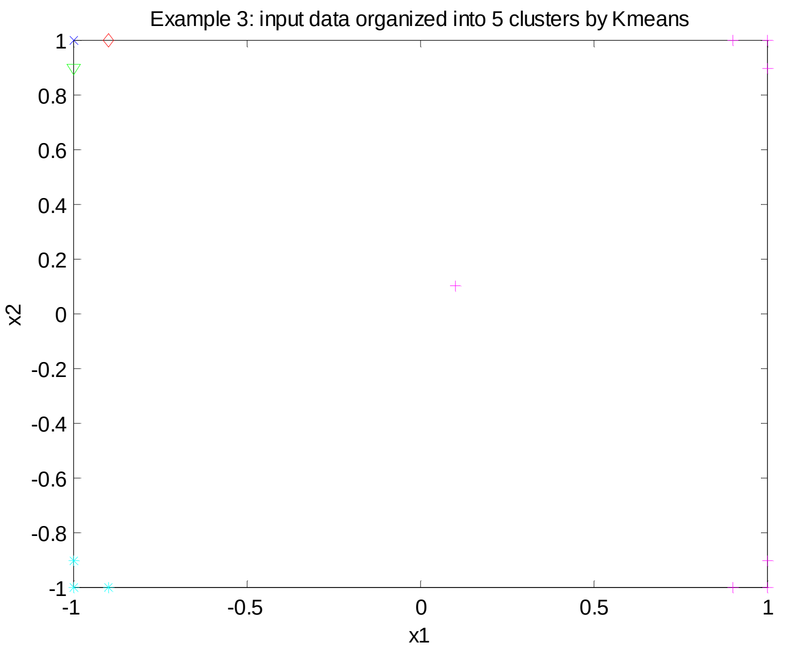

Fig 1.9: Kmeans clustering of example 3. Five clusters assumed. Initial seed points and influence of the central point produce an unlikely organization.

In performing Kmeans clustering, if we (correctly) assume five clusters, the result nonetheless can be surprising. Figure 1.9 shows a computation when the initial cluster centers are drawn from three of the corners. The influence of the central point produces a highly non-intuitive organization.

As shown in Figure 1.9, a pattern at the boundary of a cluster might by chance be relegated to a separate cluster. The isolated pattern near the middle can subsequently solidify this poor choice, resulting in a high-variance cluster including the central point and two corners.

To provide ability to recover from such poor initial organizations, Kohonen has suggested a variation on the theme. In that modified approach, one searches not only the descendents of the current node but also the nodes that *are linked to the current level at the same level.* This means that in building the hierarchical tree, one needs to determine, at every level, which of the other nodes at the same level are close to the current node. It is reported that this refinement renders tree search more stable.

In the early days of stochastic pattern recognition there were hopes that interactive ways of clustering would enable the algorithm to deal with non-radially-symmetric distributions either through use of interactive interventions as in the ISODATA procedure or with use of slightly modified metrics as in the use of the Mahalanobis distance. However, these approaches do not lead to generic practicable procedures and will not be discussed further in these notes.