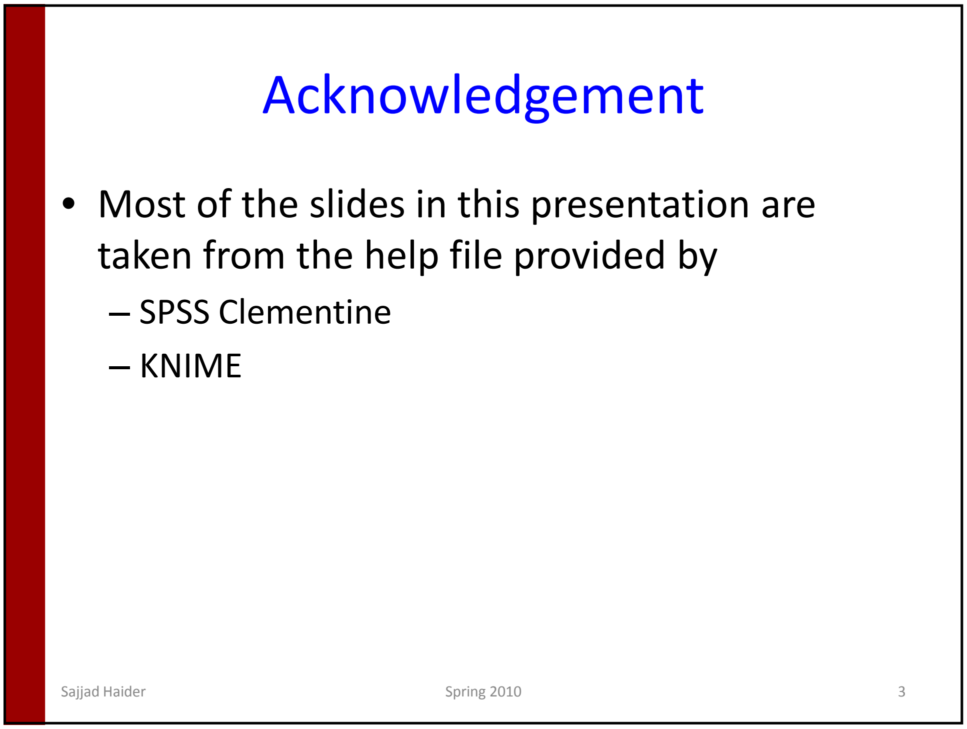
Knowledge Discovery and Data Mining

Unit # 10

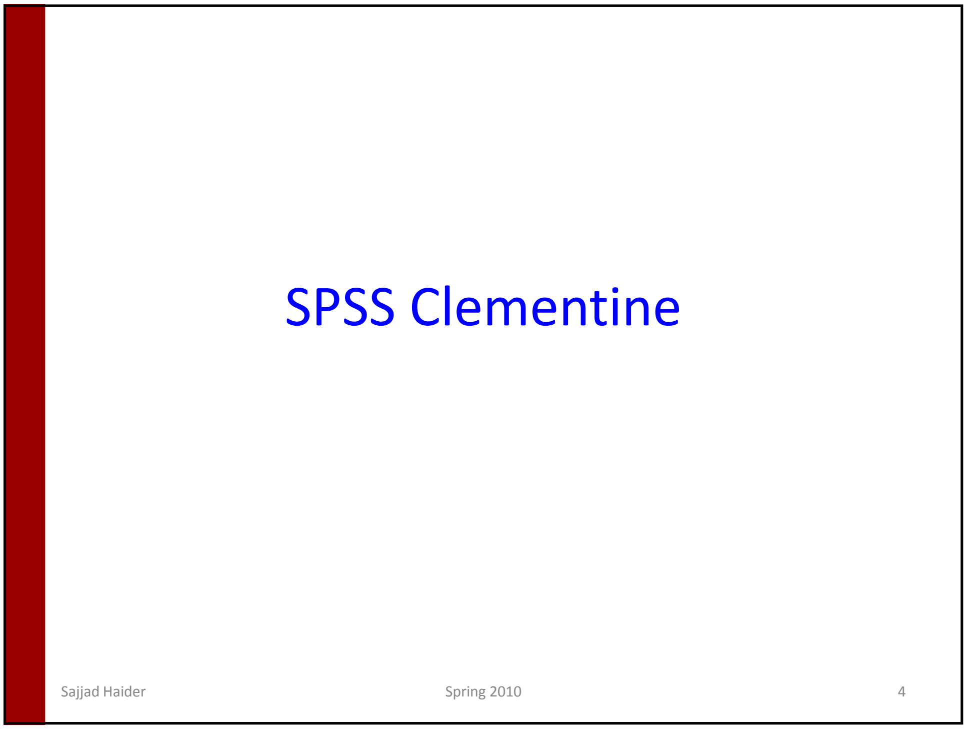
Clustering Algorithms

- SPSS Clementine
  - K-Means
  - Kohonen Maps
  - Two-Step Cluster
  - Anomaly Detection
- KNIME
  - K-Means
  - Agglomerative
  - Fuzzy C-Means

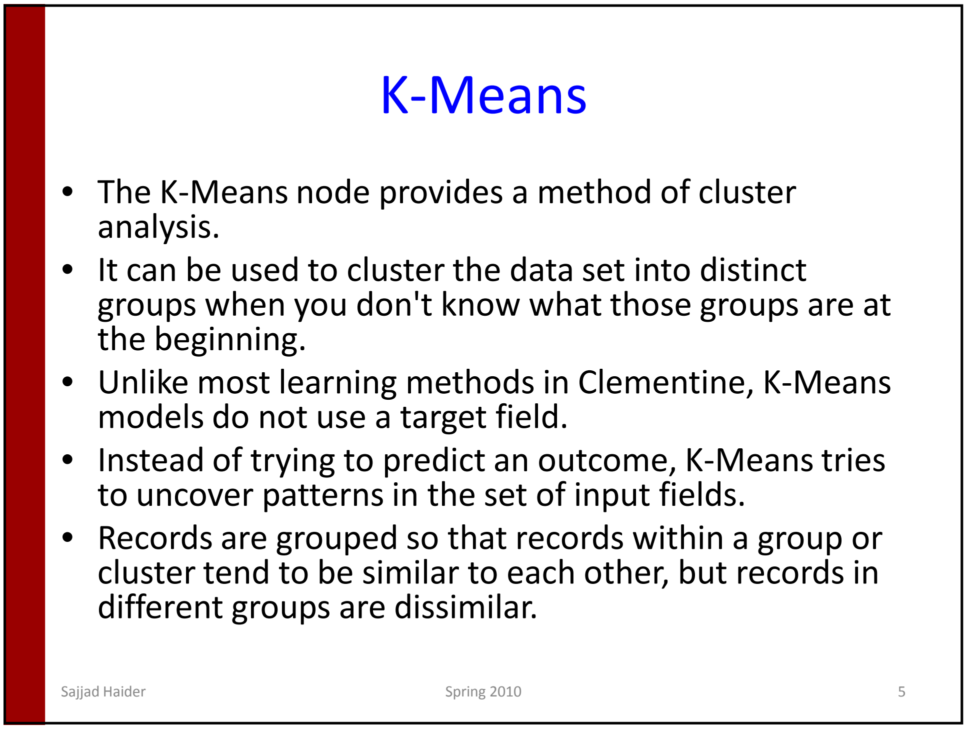


Acknowledgement

- Most of the slides in this presentation are taken from the help file provided by
  - SPSS Clementine
  - KNIME

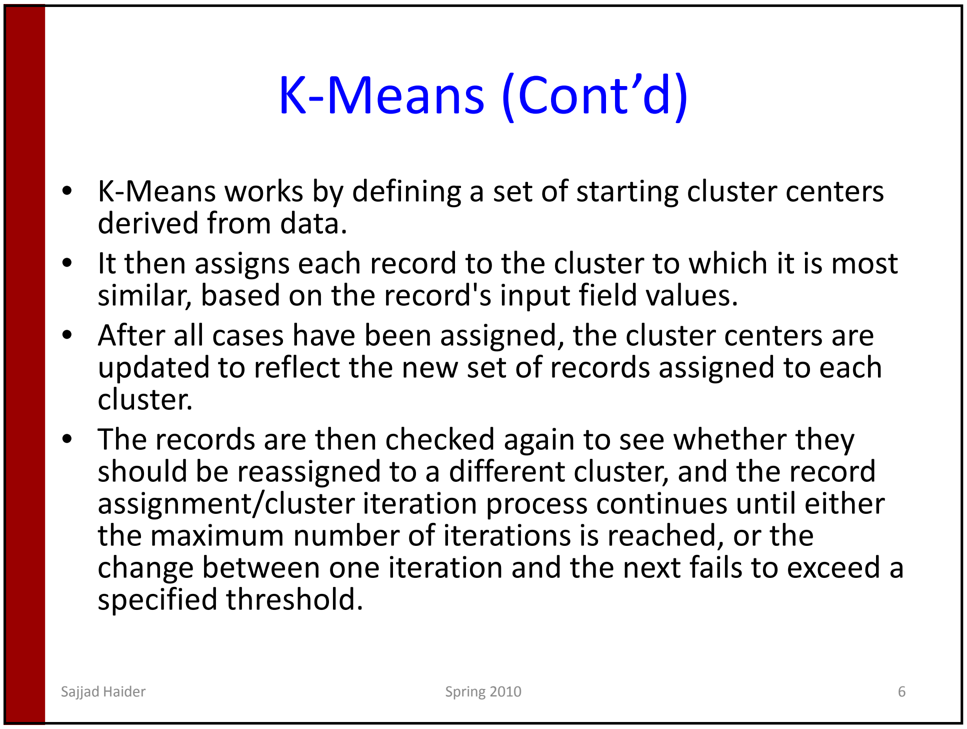


SPSS Clementine



K-Means

- The K-Means node provides a method of cluster analysis.
- It can be used to cluster the data set into distinct groups when you don't know what those groups are at the beginning.
- Unlike most learning methods in Clementine, K-Means models do not use a target field.
- Instead of trying to predict an outcome, K-Means tries to uncover patterns in the set of input fields.
- Records are grouped so that records within a group or cluster tend to be similar to each other, but records in different groups are dissimilar.



K-Means (Cont’d)

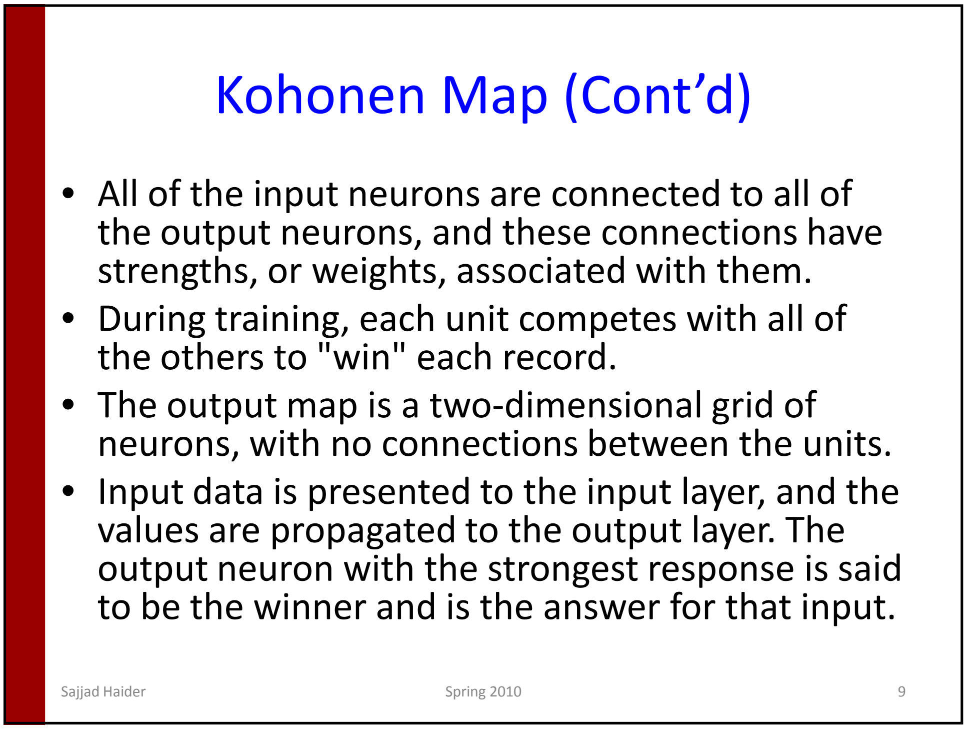
- K-Means works by defining a set of starting cluster centers derived from data.
- It then assigns each record to the cluster to which it is most similar, based on the record's input field values.
- After all cases have been assigned, the cluster centers are updated to reflect the new set of records assigned to each cluster.
- The records are then checked again to see whether they should be reassigned to a different cluster, and the record assignment/cluster iteration process continues until either the maximum number of iterations is reached, or the change between one iteration and the next fails to exceed a specified threshold.

K-Means (Cont'd)

- Note: The resulting model depends to a certain extent on the order of the training data. Reordering the data and rebuilding the model may lead to a different final cluster model.
- Requirements. To train a K-Means model, you need one or more In fields. Fields with direction Out, Both, or None are ignored.
- Strengths. You do not need to have data on group membership to build a K-Means model. The K-Means model is often the fastest method of clustering for large data sets.

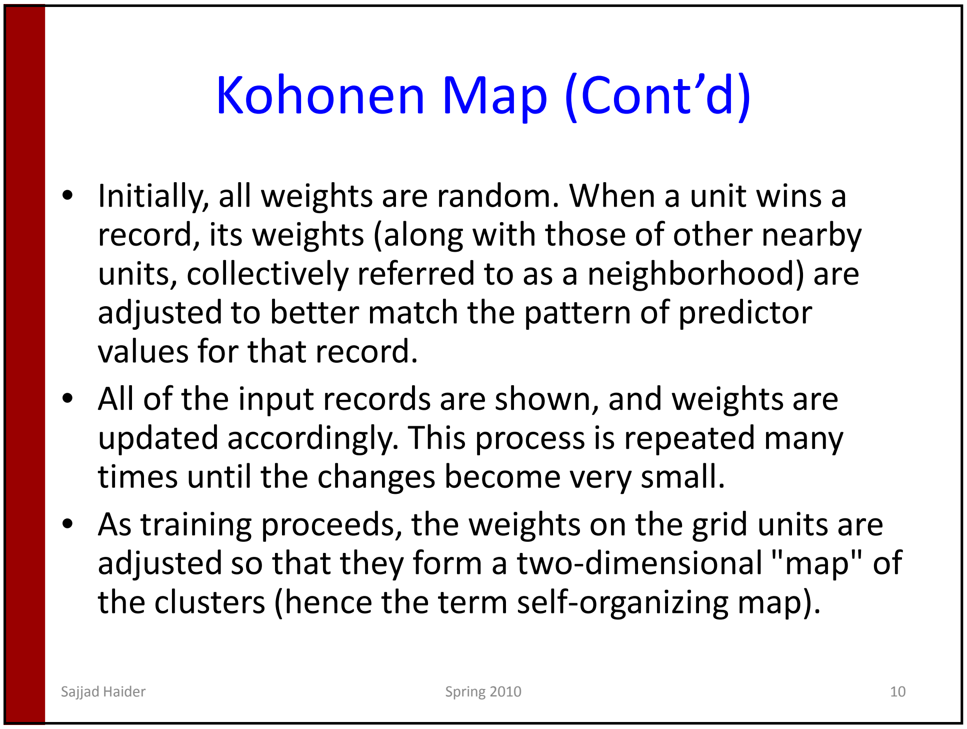
Kohonen Map

- Kohonen networks are a type of neural network that perform clustering, also known as a knet or a self-organizing map.
- This type of network can be used to cluster the data set into distinct groups when you don't know what those groups are at the beginning.
- Records are grouped so that records within a group or cluster tend to be similar to each other, and records in different groups are dissimilar.
- The basic units are neurons, and they are organized into two layers: the input layer and the output layer (also called the output map).



Kohonen Map (Cont'd)

- All of the input neurons are connected to all of the output neurons, and these connections have strengths, or weights, associated with them.
- During training, each unit competes with all of the others to "win" each record.
- The output map is a two-dimensional grid of neurons, with no connections between the units.
- Input data is presented to the input layer, and the values are propagated to the output layer. The output neuron with the strongest response is said to be the winner and is the answer for that input.



Kohonen Map (Cont'd)

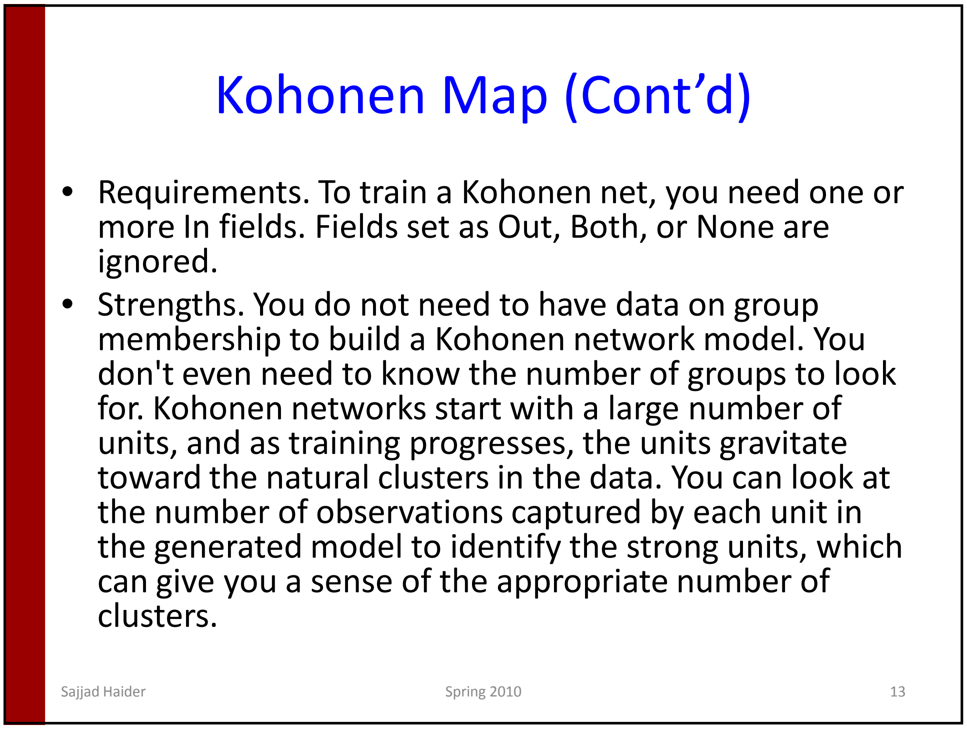
- Initially, all weights are random. When a unit wins a record, its weights (along with those of other nearby units, collectively referred to as a neighborhood) are adjusted to better match the pattern of predictor values for that record.
- All of the input records are shown, and weights are updated accordingly. This process is repeated many times until the changes become very small.
- As training proceeds, the weights on the grid units are adjusted so that they form a two-dimensional "map" of the clusters (hence the term self-organizing map).

Kohonen Map (Cont’d)

- When the network is fully trained, records that are similar should appear close together on the output map, whereas records that are vastly different will appear far apart.
- Unlike most learning methods in Clementine, Kohonen networks do not use a target field.
- Instead of trying to predict an outcome, Kohonen nets try to uncover patterns in the set of input fields.

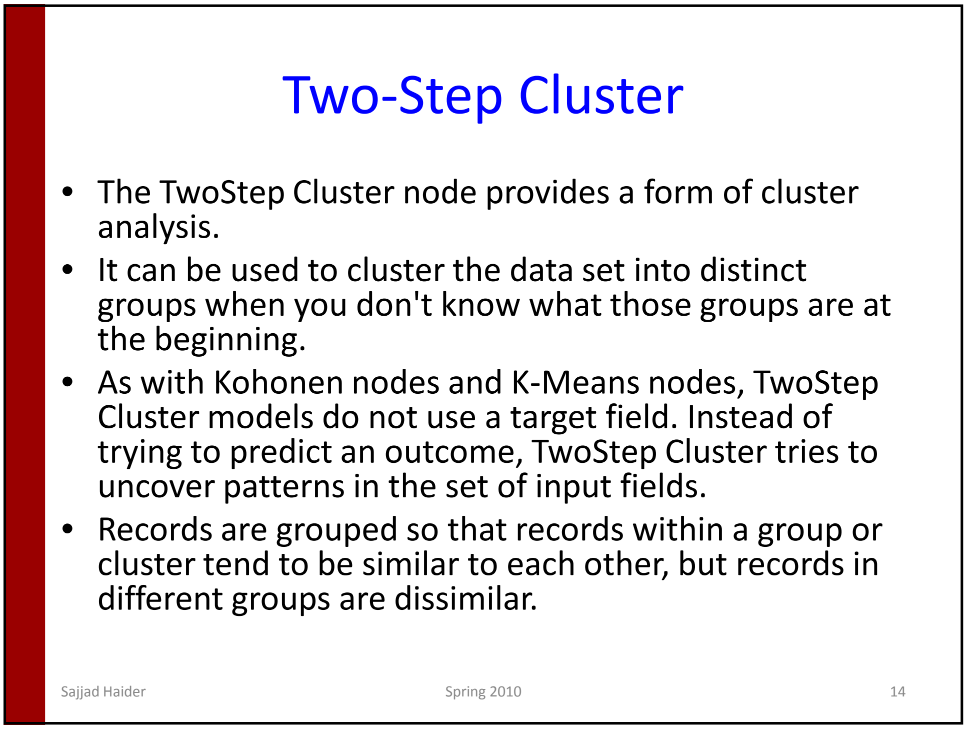
Kohonen Map (Cont’d)

- Usually, a Kohonen net will end up with a few units that summarize many observations (strong units), and several units that don't really correspond to any of the observations (weak units). The strong units (and sometimes other units adjacent to them in the grid) represent probable cluster centers.
- Another use of Kohonen networks is in dimension reduction. The spatial characteristic of the two-dimensional grid provides a mapping from the k original predictors to two derived features that preserve the similarity relationships in the original predictors. In some cases, this can give you the same kind of benefit as factor analysis or PCA.



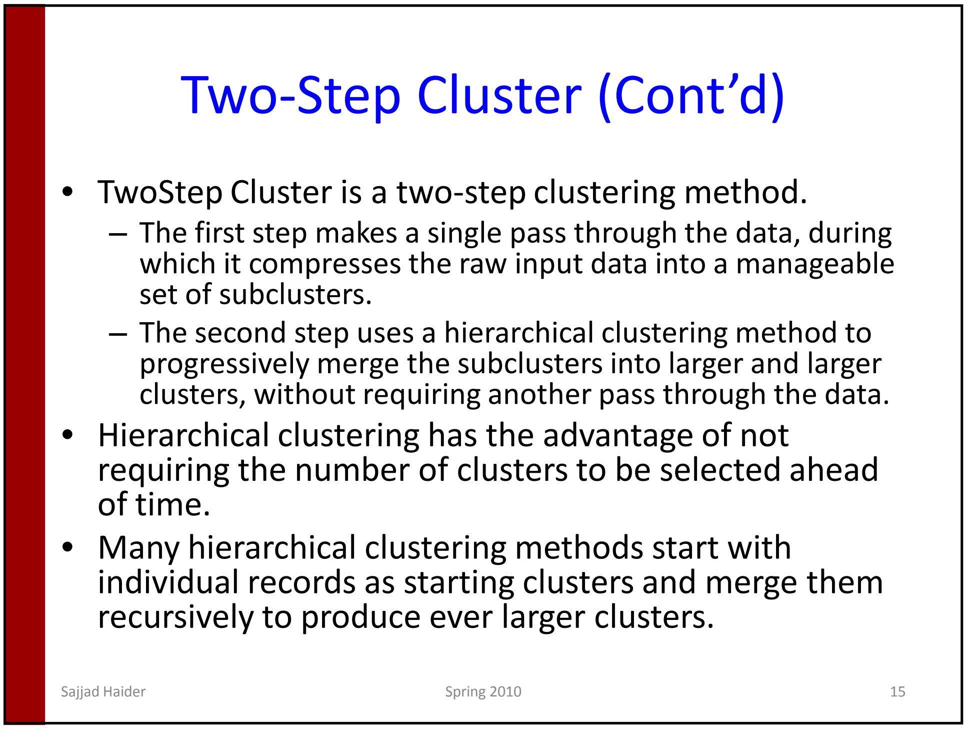
Kohonen Map (Cont’d)

- Requirements. To train a Kohonen net, you need one or more In fields. Fields set as Out, Both, or None are ignored.
- Strengths. You do not need to have data on group membership to build a Kohonen network model. You don't even need to know the number of groups to look for. Kohonen networks start with a large number of units, and as training progresses, the units gravitate toward the natural clusters in the data. You can look at the number of observations captured by each unit in the generated model to identify the strong units, which can give you a sense of the appropriate number of clusters.



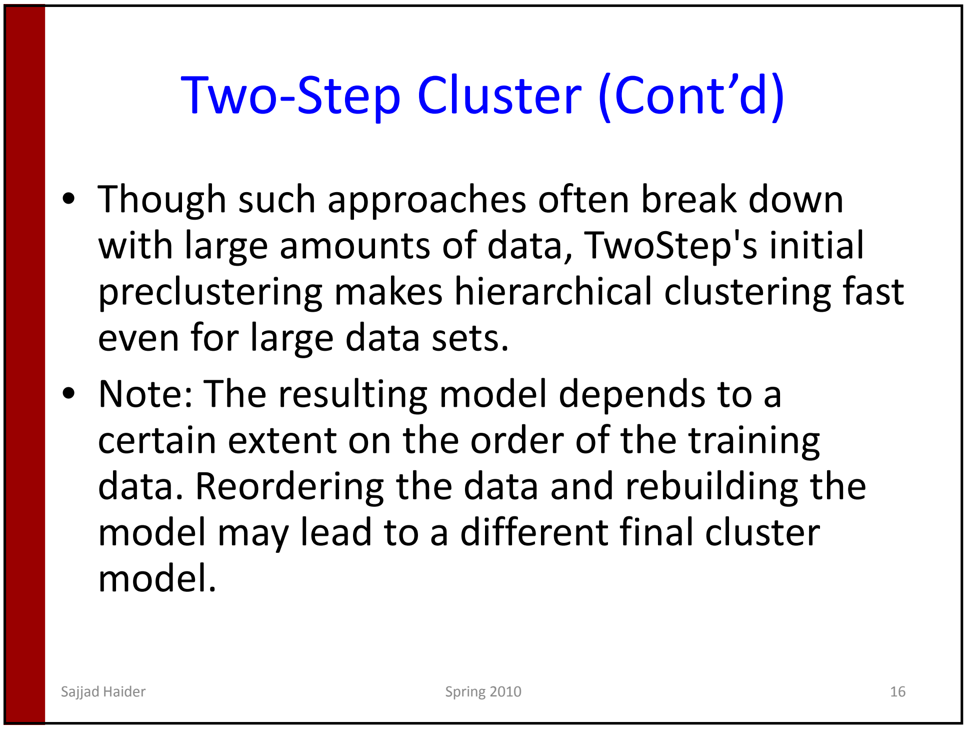
Two-Step Cluster

- The TwoStep Cluster node provides a form of cluster analysis.
- It can be used to cluster the data set into distinct groups when you don't know what those groups are at the beginning.
- As with Kohonen nodes and K-Means nodes, TwoStep Cluster models do not use a target field. Instead of trying to predict an outcome, TwoStep Cluster tries to uncover patterns in the set of input fields.
- Records are grouped so that records within a group or cluster tend to be similar to each other, but records in different groups are dissimilar.



Two-Step Cluster (Cont’d)

- TwoStep Cluster is a two-step clustering method.
  - The first step makes a single pass through the data, during which it compresses the raw input data into a manageable set of subclusters.
  - The second step uses a hierarchical clustering method to progressively merge the subclusters into larger and larger clusters, without requiring another pass through the data.
- Hierarchical clustering has the advantage of not requiring the number of clusters to be selected ahead of time.
- Many hierarchical clustering methods start with individual records as starting clusters and merge them recursively to produce ever larger clusters.



Two-Step Cluster (Cont’d)

- Though such approaches often break down with large amounts of data, TwoStep's initial preclustering makes hierarchical clustering fast even for large data sets.
- Note: The resulting model depends to a certain extent on the order of the training data. Reordering the data and rebuilding the model may lead to a different final cluster model.

Two-Step Cluster (Cont’d)

- Requirements. To train a TwoStep Cluster model, you need one or more In fields. Fields with direction Out, Both, or None are ignored. The TwoStep Cluster algorithm does not handle missing values. Records with blanks for any of the input fields will be ignored when building the model.
- Strengths. TwoStep Cluster can handle mixed field types and is able to handle large data sets efficiently. It also has the ability to test several cluster solutions and choose the best, so you don't need to know how many clusters to ask for at the outset. TwoStep Cluster can be set to automatically exclude outliers, or extremely unusual cases that can contaminate your results.

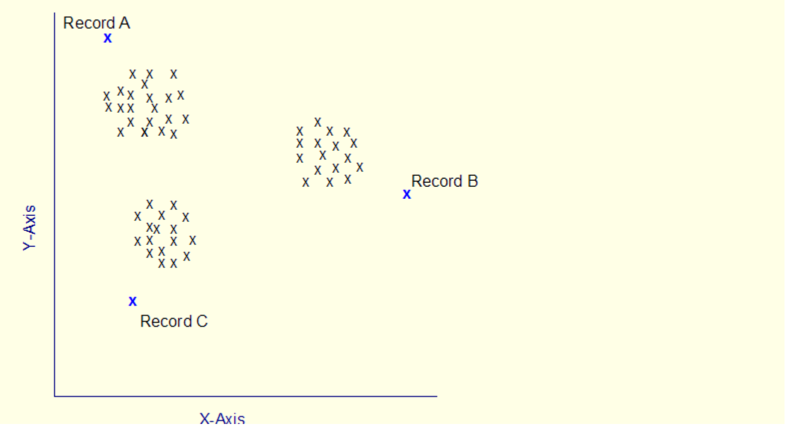
Anomaly Detection

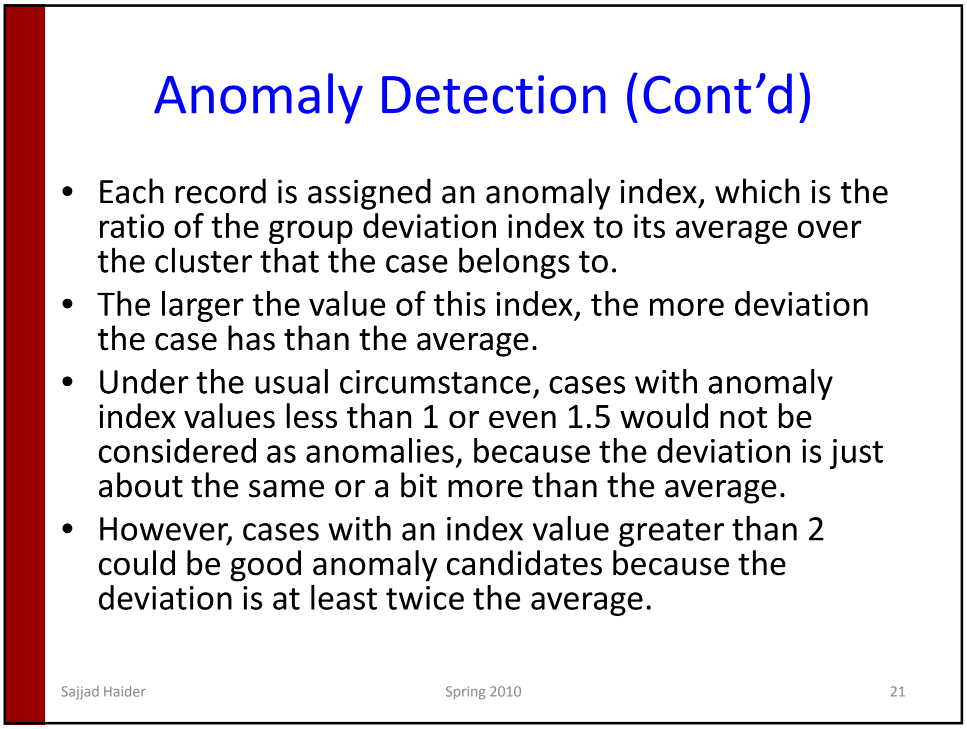
- Anomaly detection models are used to identify outliers, or unusual cases, in the data.
- Unlike other modeling methods that store rules about unusual cases, anomaly detection models store information on what normal behavior looks like.
- This makes it possible to identify outliers even if they do not conform to any known pattern, and it can be particularly useful in applications, such as fraud detection, where new patterns may constantly be emerging.
- Anomaly detection is an unsupervised method, which means that it does not require a training dataset containing known cases of fraud to use as a starting point.

Anomaly Detection (Cont'd)

- While traditional methods of identifying outliers generally look at one or two variables at a time, anomaly detection can examine large numbers of fields to identify clusters or peer groups into which similar records fall.
- Each record can then be compared to others in its peer group to identify possible anomalies.
- The further away a case is from the normal center, the more likely it is to be unusual.
- For example, the algorithm might lump records into three distinct clusters and flag those that fall far from the center of any one cluster.

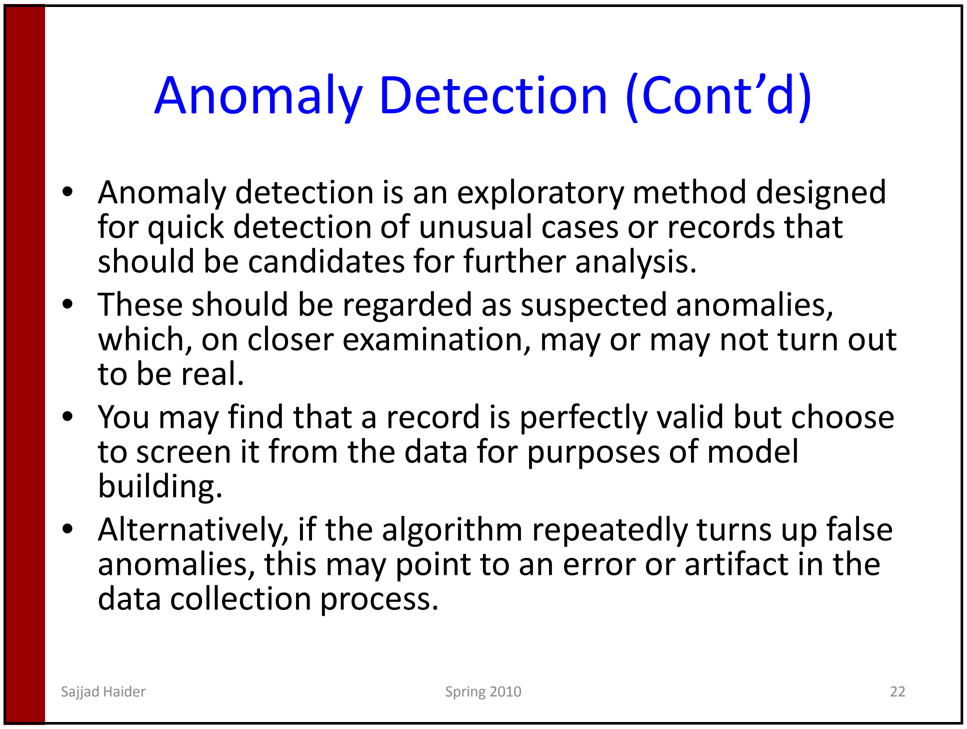
Anomaly Detection (Cont'd)





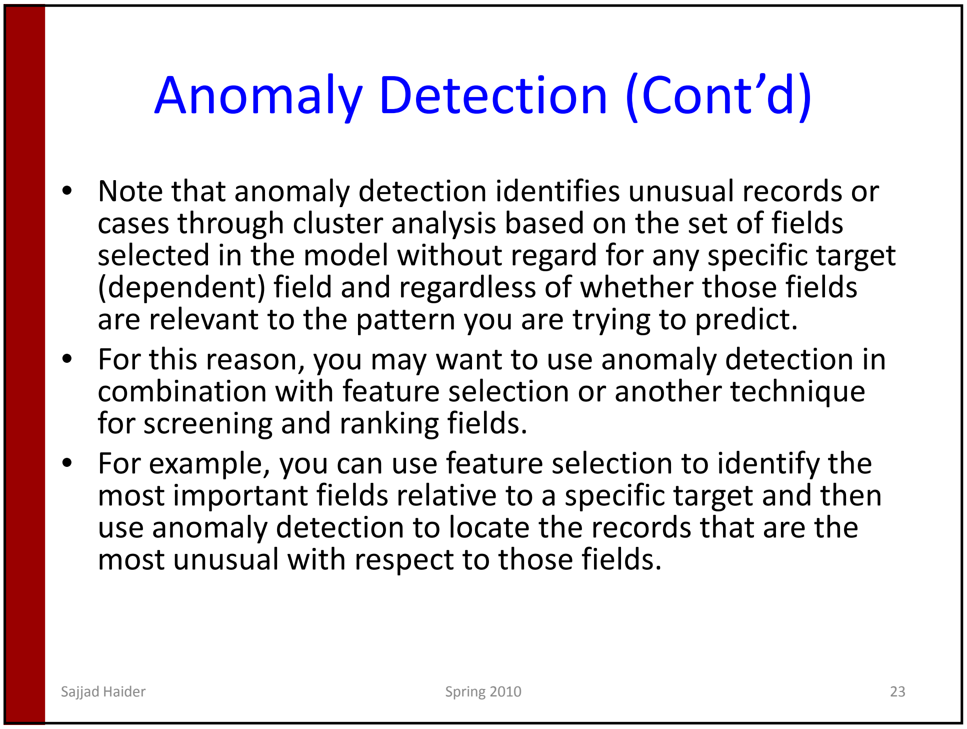
Anomaly Detection (Cont’d)

- Each record is assigned an anomaly index, which is the ratio of the group deviation index to its average over the cluster that the case belongs to.
- The larger the value of this index, the more deviation the case has than the average.
- Under the usual circumstance, cases with anomaly index values less than 1 or even 1.5 would not be considered as anomalies, because the deviation is just about the same or a bit more than the average.
- However, cases with an index value greater than 2 could be good anomaly candidates because the deviation is at least twice the average.



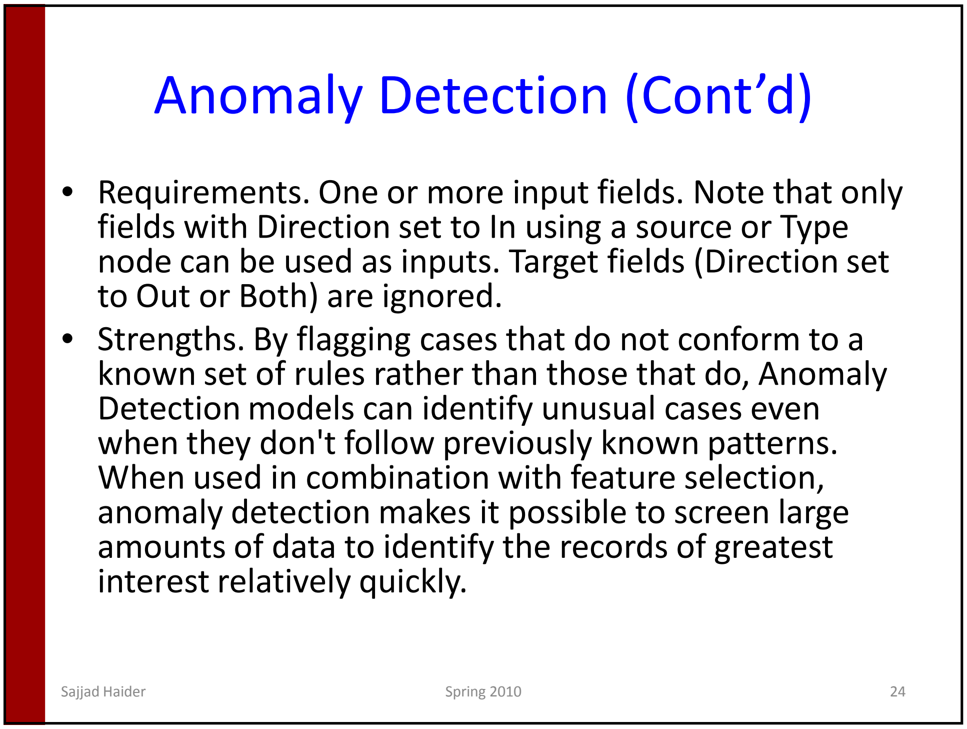
Anomaly Detection (Cont’d)

- Anomaly detection is an exploratory method designed for quick detection of unusual cases or records that should be candidates for further analysis.
- These should be regarded as suspected anomalies, which, on closer examination, may or may not turn out to be real.
- You may find that a record is perfectly valid but choose to screen it from the data for purposes of model building.
- Alternatively, if the algorithm repeatedly turns up false anomalies, this may point to an error or artifact in the data collection process.



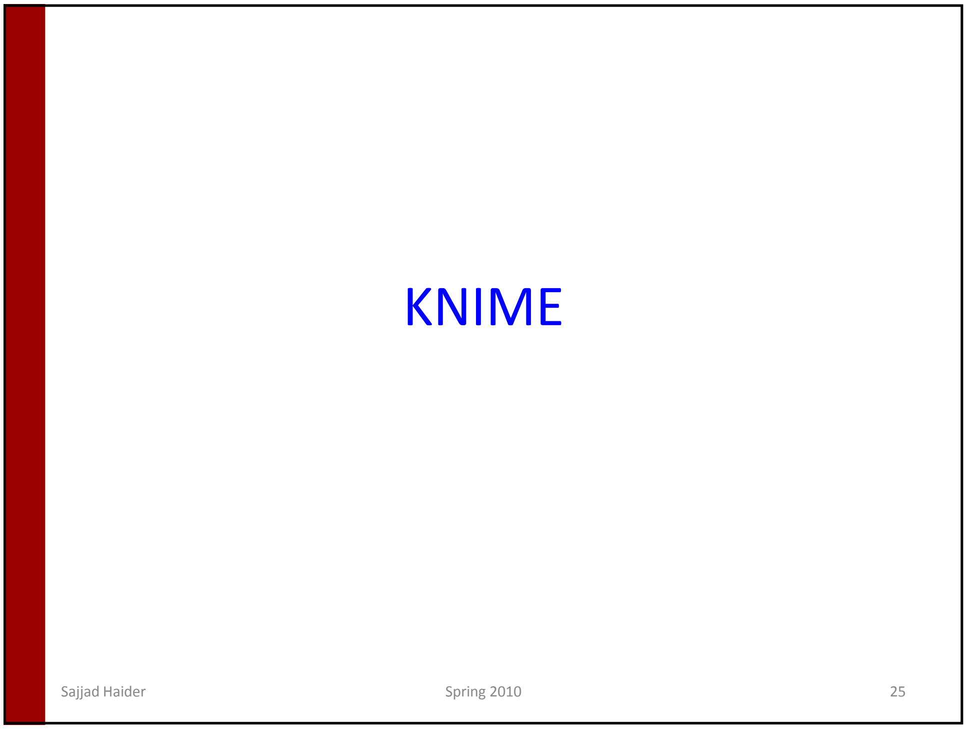
Anomaly Detection (Cont’d)

- Note that anomaly detection identifies unusual records or cases through cluster analysis based on the set of fields selected in the model without regard for any specific target (dependent) field and regardless of whether those fields are relevant to the pattern you are trying to predict.
- For this reason, you may want to use anomaly detection in combination with feature selection or another technique for screening and ranking fields.
- For example, you can use feature selection to identify the most important fields relative to a specific target and then use anomaly detection to locate the records that are the most unusual with respect to those fields.

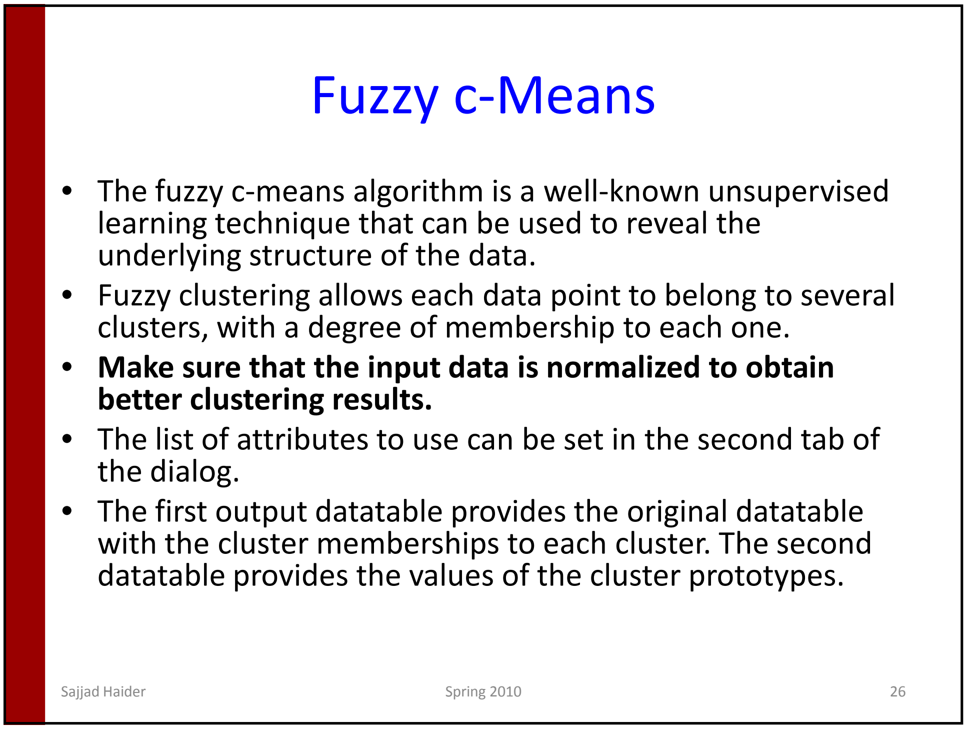


Anomaly Detection (Cont’d)

- Requirements. One or more input fields. Note that only fields with Direction set to In using a source or Type node can be used as inputs. Target fields (Direction set to Out or Both) are ignored.
- Strengths. By flagging cases that do not conform to a known set of rules rather than those that do, Anomaly Detection models can identify unusual cases even when they don't follow previously known patterns. When used in combination with feature selection, anomaly detection makes it possible to screen large amounts of data to identify the records of greatest interest relatively quickly.

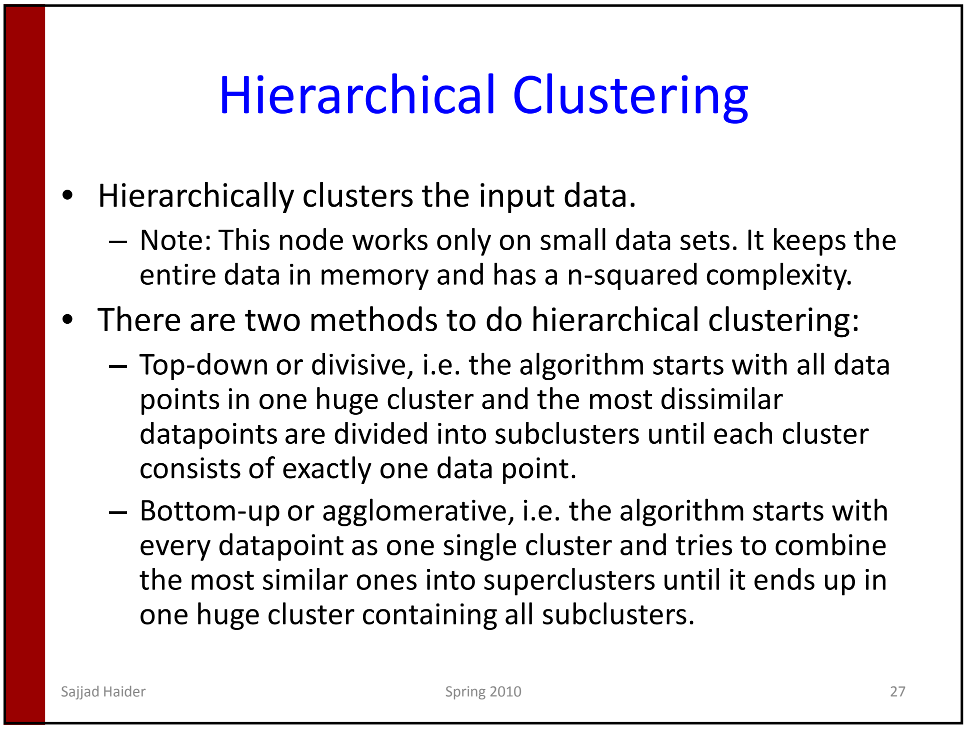


KNIME



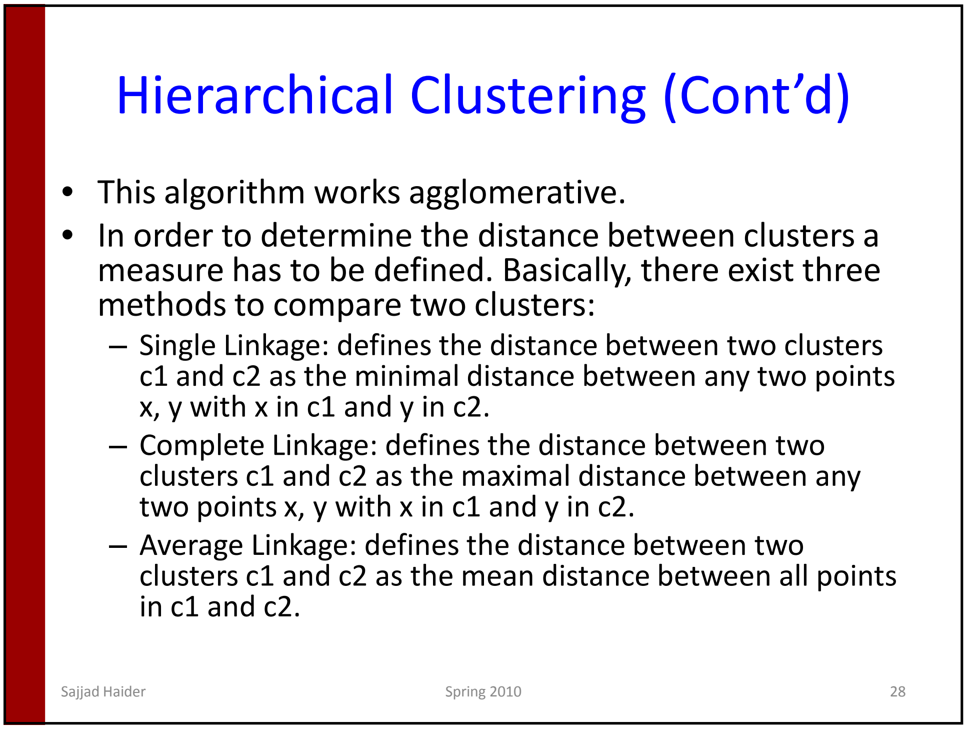
Fuzzy c-Means

- The fuzzy c-means algorithm is a well-known unsupervised learning technique that can be used to reveal the underlying structure of the data.
- Fuzzy clustering allows each data point to belong to several clusters, with a degree of membership to each one.
- **Make sure that the input data is normalized to obtain better clustering results.**
- The list of attributes to use can be set in the second tab of the dialog.
- The first output datatable provides the original datatable with the cluster memberships to each cluster. The second datatable provides the values of the cluster prototypes.



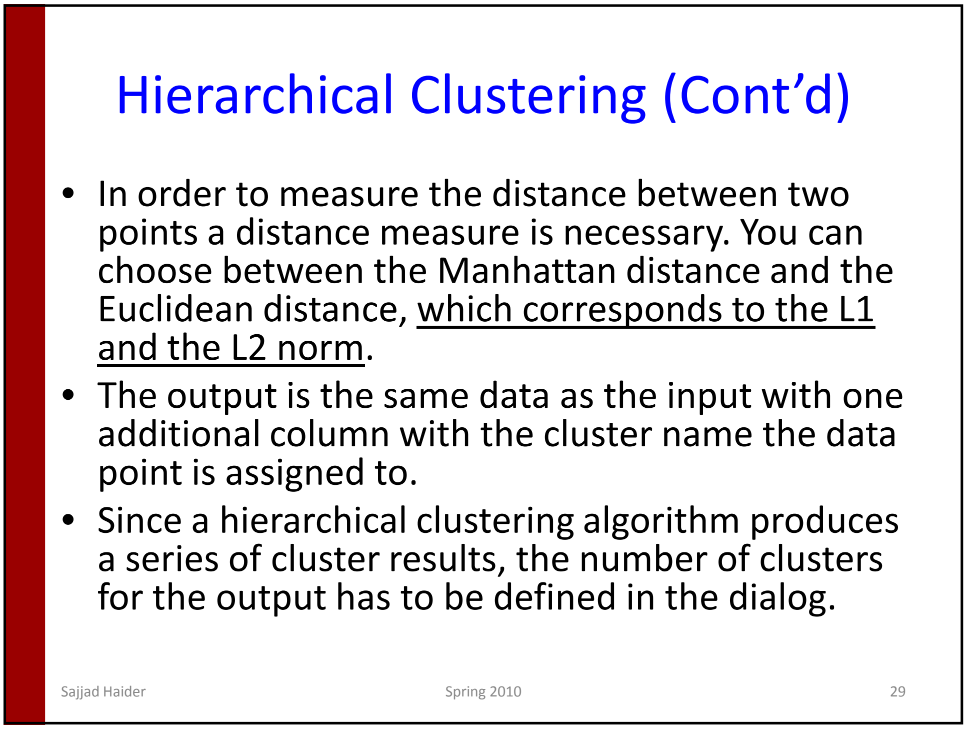
Hierarchical Clustering

- Hierarchically clusters the input data.
  - Note: This node works only on small data sets. It keeps the entire data in memory and has a n-squared complexity.
- There are two methods to do hierarchical clustering:
  - Top-down or divisive, i.e. the algorithm starts with all data points in one huge cluster and the most dissimilar datapoints are divided into subclusters until each cluster consists of exactly one data point.
  - Bottom-up or agglomerative, i.e. the algorithm starts with every datapoint as one single cluster and tries to combine the most similar ones into superclusters until it ends up in one huge cluster containing all subclusters.



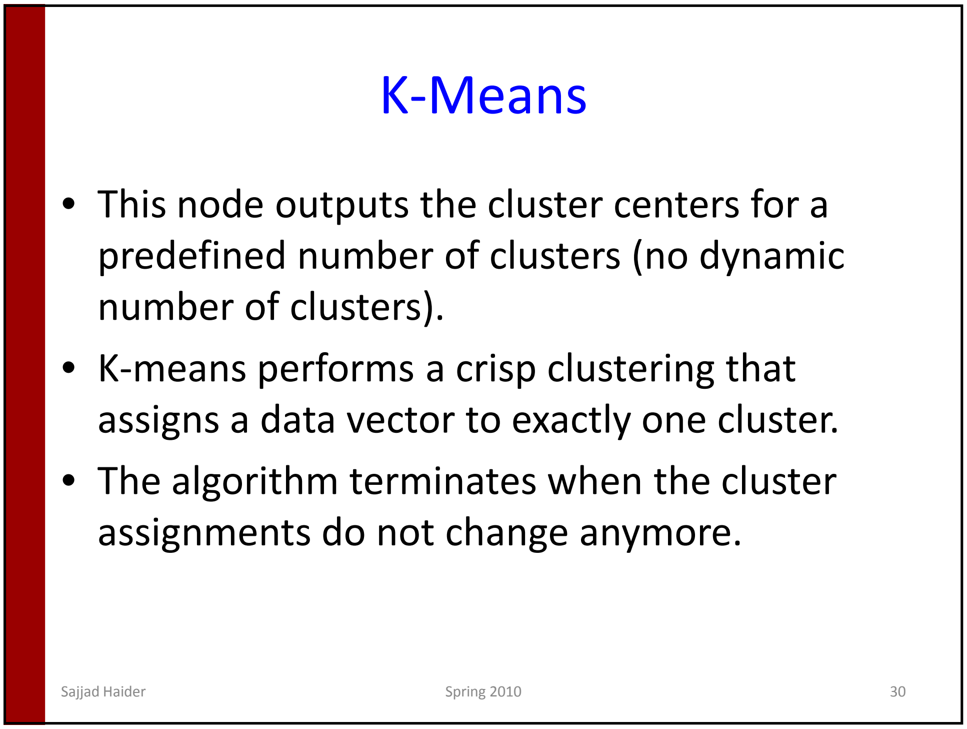
Hierarchical Clustering (Cont'd)

- This algorithm works agglomerative.
- In order to determine the distance between clusters a measure has to be defined. Basically, there exist three methods to compare two clusters:
  - Single Linkage: defines the distance between two clusters c1 and c2 as the minimal distance between any two points x, y with x in c1 and y in c2.
  - Complete Linkage: defines the distance between two clusters c1 and c2 as the maximal distance between any two points x, y with x in c1 and y in c2.
  - Average Linkage: defines the distance between two clusters c1 and c2 as the mean distance between all points in c1 and c2.



Hierarchical Clustering (Cont'd)

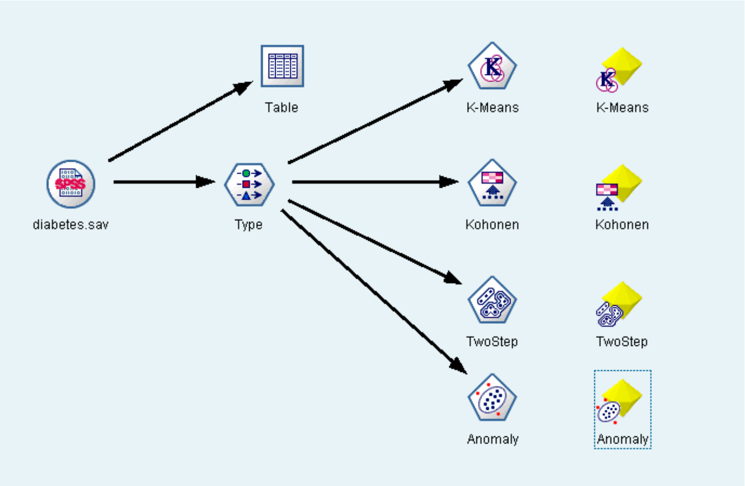
- In order to measure the distance between two points a distance measure is necessary. You can choose between the Manhattan distance and the Euclidean distance, which corresponds to the L1 and the L2 norm.
- The output is the same data as the input with one additional column with the cluster name the data point is assigned to.
- Since a hierarchical clustering algorithm produces a series of cluster results, the number of clusters for the output has to be defined in the dialog.



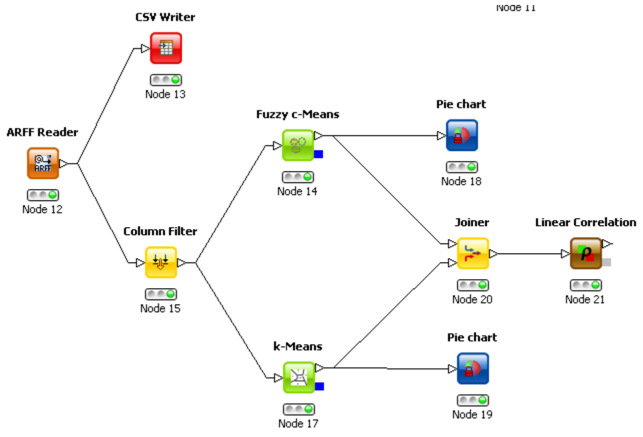
K-Means

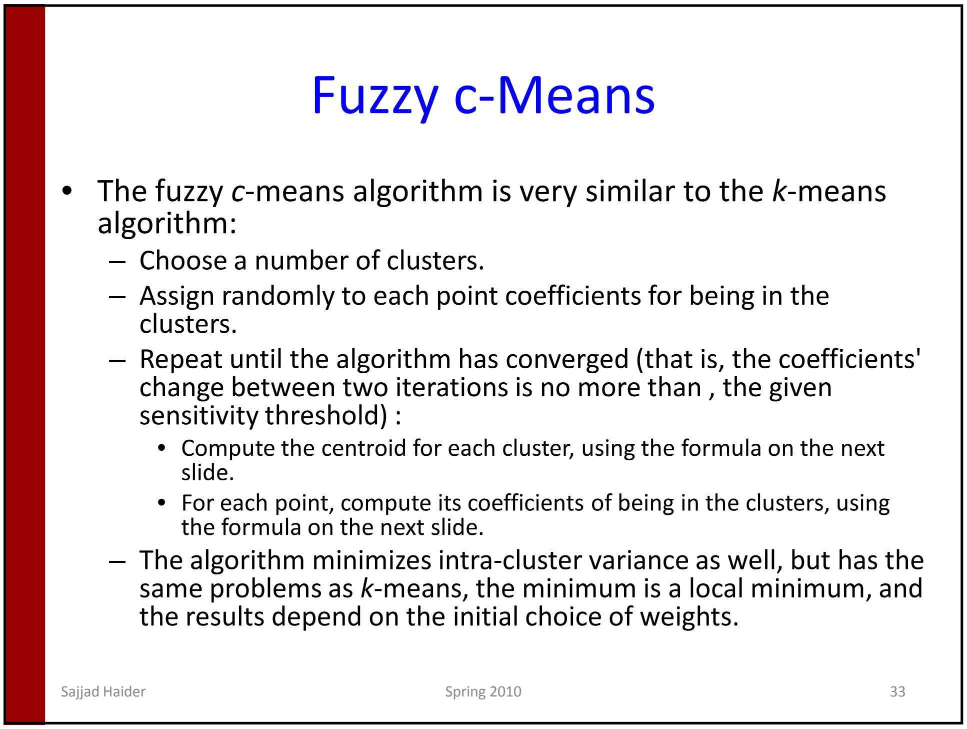
- This node outputs the cluster centers for a predefined number of clusters (no dynamic number of clusters).
- K-means performs a crisp clustering that assigns a data vector to exactly one cluster.
- The algorithm terminates when the cluster assignments do not change anymore.

Clementine Workflow



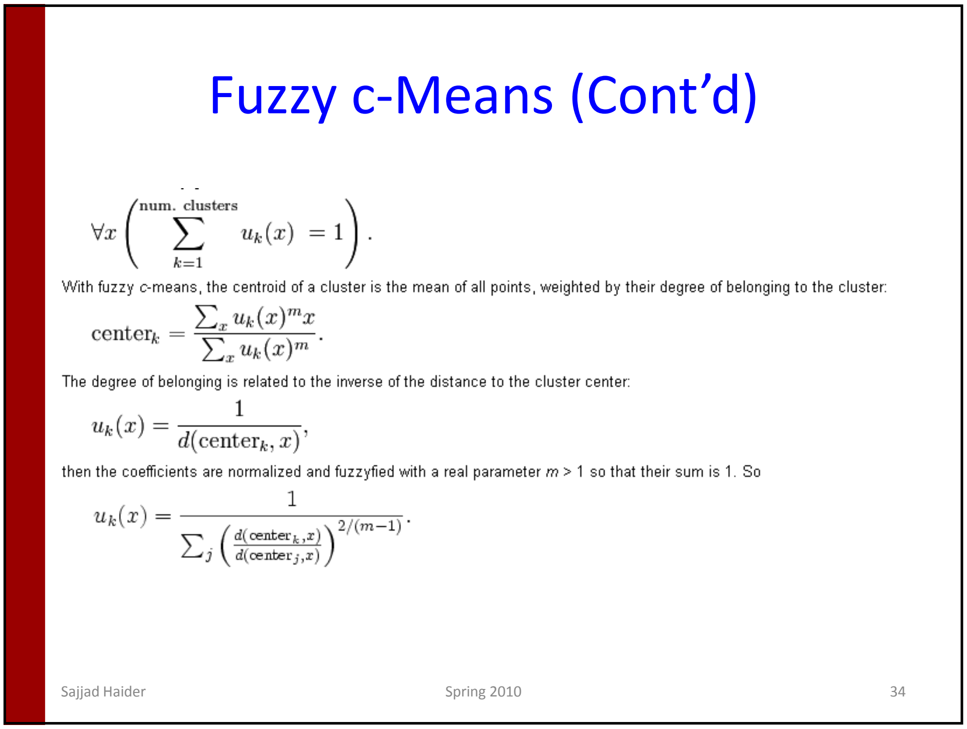
KNIME Workflow





Fuzzy c-Means

- The fuzzy *c*-means algorithm is very similar to the *k*-means algorithm:
  - Choose a number of clusters.
  - Assign randomly to each point coefficients for being in the clusters.
  - Repeat until the algorithm has converged (that is, the coefficients' change between two iterations is no more than , the given sensitivity threshold) :
    - Compute the centroid for each cluster, using the formula on the next slide.
    - For each point, compute its coefficients of being in the clusters, using the formula on the next slide.
  - The algorithm minimizes intra-cluster variance as well, but has the same problems as *k*-means, the minimum is a local minimum, and the results depend on the initial choice of weights.



Fuzzy c-Means (Cont'd)

$$\forall x \left(\sum_{k=1}^{\text{num. clusters}} u_k(x) \ = 1 \right).$$

With fuzzy *c*-means, the centroid of a cluster is the mean of all points, weighted by their degree of belonging to the cluster:

$$\mathrm{center}_k = \frac{\sum_x u_k(x)^m x}{\sum_x u_k(x)^m}.$$

The degree of belonging is related to the inverse of the distance to the cluster center:

$$u_k(x) = \frac{1}{d(\mathrm{center}_k, x)},$$

then the coefficients are normalized and fuzzyfied with a real parameter $m > 1$ so that their sum is 1. So

$$u_k(x) = \frac{1}{\sum_j \left(\frac{d(\mathrm{center}_k, x)}{d(\mathrm{center}_j, x)} \right)^{2/(m-1)}}.$$