Clustering with TeeChart - Preview

Tutorial
David Berneda
June 2011
Steema Software

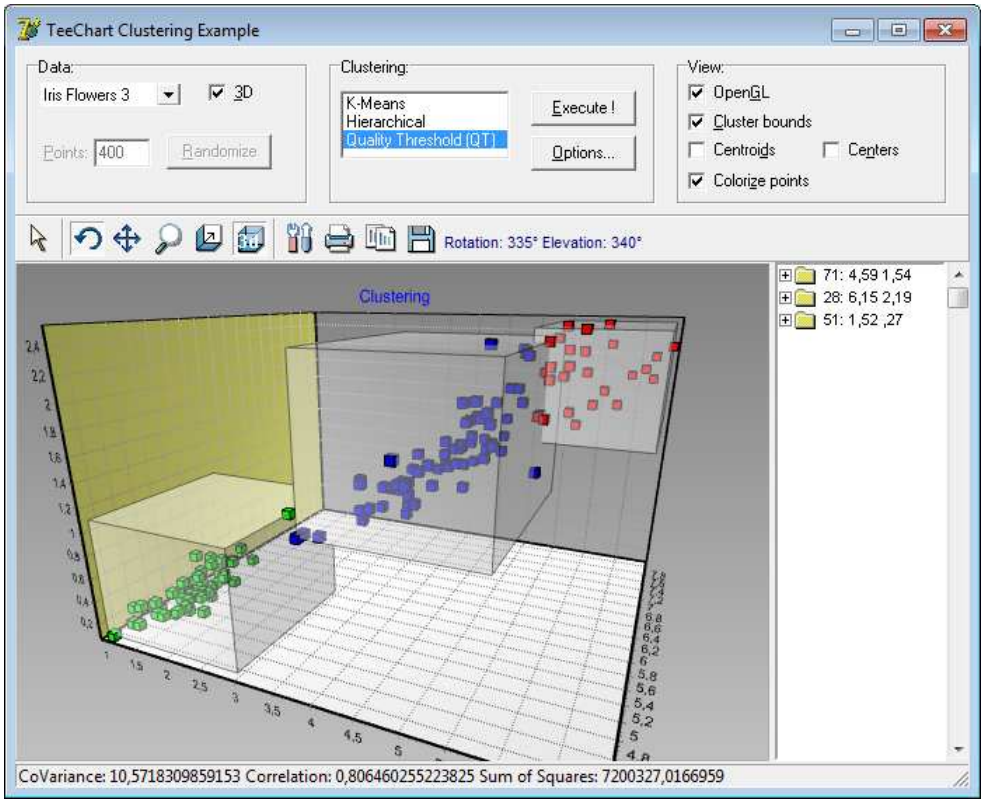
The TeeChart VCL preview library includes classes and components to perform "clustering" on your data.

Clustering is the process of grouping data according to how well related are the individual items.

For more information on clustering visit the following Wikipedia link:
http://en.wikipedia.org/wiki/Cluster_analysis

Download executable example:

http://www.steema.us/files/public/teechart/vcl/demos/clustering/TeeChart_Clustering.zip



Classes and Units

The TeeClustering.pas unit contains abstract "engine" classes that perform the clustering algorithms.

Three different clustering methods are provided:

- TKMeansClustering
- THierarchicalClustering
- TQTClustering (Quality Threshold)

These classes derive from a common abstract class: TBaseClustering

Each clustering type has its own properties that determine how will the clusters be calculated.

After calculating, you can access the Clusters property, which is a TList of TCluster objects.

A TCluster contains child clusters (Items[ ]), so you can check which input data items belong to which cluster, or in the case of the Hierarchical type, access the tree structure (clusters and sub-clusters).

The input data (your data) is not contained by the above classes.
Data is passed to the clustering engine through a "provider" class.

There is currently one kind of data provider (TSeriesProvider) to cluster XY or XYZ Series points.

This class is implemented in the TeeClusteringTool.pas unit, together with a charting Tool class (TClusteringTool) to make things easier and more automatic.

Basic Example

Example runtime code (it can be done at design-time too, without coding) :

```
uses TeeClusteringTool;

var tool : TClusteringTool;

tool:=TClusteringTool.Create(Self);
tool.ParentChart:=Chart1;
tool.Series:=Series1;              // your series
tool.Method:=cmKMeans;
tool.KMeans.NumClusters:=5;

tool.Execute;
```

After execution, you can loop on the resulting output clusters, for example:

```
var t : Integer;
for t:=0 to tool.Clusters.Count-1 do
    Memo1.Lines.Add( 'Cluster: '+IntToStr(t)+' contains:
'+IntToStr(tool.Clusters[t].Count)+' points' );
```

TClusteringTool

This tool will automatically perform clustering using the choosen method and parameters, and it will optionally paint each series point with a different color indicating which cluster they belong to, or draw polygons around each group of cluster's items, among other things.

Properties:

```
ClusteringTool1.Method := cmHierarchical;
ClusteringTool1.ColorEach := True;    // paint Series with one color per cluster
ClusteringTool1.ShowBounds := True;   // draws convex polygons bounding each cluster points
ClusteringTool1.Centers.Visible := True;   // shows cluster centers
ClusteringTool1.Centroids.Visible := True;  // shows cluster centroids
```

Other properties include Brush, Pen and Transparency, used when drawing cluster polygon boundaries.

Methods:

Several helper methods are provided:

```
// Obtain cluster's center and centroid XY points in Series scales:
var P : TPointFloat;
P:=ClusteringTool1.GetClusterCenter( ClusteringTool1.Clusters[3] );
P:=ClusteringTool1.GetClusterCentroid( ClusteringTool1.Clusters[2] );

// Obtain an array of XY points (in screen pixel coordinates), that belong to a cluster:
var PP : TPointArray;
ClusteringTool1.GetClusterPoints( ClusteringTool1.Clusters[4], PP);
...
PP:=nil;

// Get cluster statistics:
var S : TClusterStats;
S:=ClusteringTool1.GetStats(  ClusteringTool1.Clusters[0] );
```

Calculation parameters

Each clustering algorithm needs different parameters:

K-Means:

```
ClusteringTool1.KMeans.NumClusters := 10;   // Number of minimum clusters ("K")
ClusteringTool1.KMeans.MaxIterations := 1000;   // Maximum number of iterations before stopping
```

Hierarchical:

ClusteringTool1.Hierarchical.NumClusters := 8; // Number of tree root clusters

QT:

ClusteringTool1.QTClustering.MinCount := 30; // Minimum number of points to form a cluster
ClusteringTool1.QTClustering.MaxDiameter := 100; // Maximum "diameter" a cluster can grow

Common parameters:

Distance

Cluster calculation is based on the "distance" between a data item and the other data items.

There are several ways to calculate the "distance" between items.
The algorithms are agnostic, they call the Provider (ie: Series provider) to obtain the distances.

For example, on a XY scatter plot, the distance between points can be the hypotenuse (Pythagoras' theorem), that is, the simple Euclidean distance between a point XY and another XY.

Distance calculations implemented:

dmEuclidean, dmSquaredEuclidean, dmManhattan, dmMinkowski, dmSorensen, dmChebyshev

Example:

ClusteringTool1.Distance := dmMinkowski;
ClusteringTool1.MinkowskiLambda := 4;

Linkage

There are several ways to calculate the "distance" between clusters when one or the two clusters have more than one item. This is called "linkage".

The most simple way is using each cluster "center" (this means no linkage occurs).

Other linkage styles implemented:

lmSingle

Also called "minimum". The distance between cluster A and B is the minimum distance between all items in cluster A and all items in cluster B.

lmComplete

Also called "maximum". The distance between cluster A and B is the maximum distance between all items in cluster A and all items in cluster B.

lmAverage

The distance between cluster A and B is the average distance between all items in cluster A and all items in cluster B.

lmWard

The result is the increase on "error sum of squares" when adding cluster B items to cluster A.

Calculation speed

Clustering is a slow process by nature. Each clustering method has different performance bottlenecks, proportional to the number of input data items.

The TeeClustering.pas unit has been greatly fine-tuned to optimize the speed of each algorithm, although much work is needed to find more advanced techniques that require less CPU cycles.

Speed examples:

(Time in milliseconds, Intel i5 430Mobile @ 2.27Ghz)

Algorithm	**Number of input data items**			
	5000	20000	50000	100000
K-Means	171	390	3245	12168

K-Means algorithm chooses initial random clusters, so consecutive executions give different results.

Algorithm	**Number of input data items**			
	500	1000	2000	5000

Quality Threshold	46	172	702	4633

Quality Threshold is much slower than K-Means and needs much more memory.

Algorithm	**Number of input data items**			
	500	1000	2000	5000
Hierarchical	46	156	999	12839

Hierarchical is also slower than K-Means, and speed decreases exponentially on number of input data.