Cluster Hierarchy Benchmark Data Generator - documentation

Micha Spytkowski

Mon Mar 16 14:23:05 CET 2015

1 Running the generator

The generator provided here can be used to produce random data samples along with a specific data model from which they have been drawn. Running the generator results both in the data and the model being saved in the form of four files. Two of the files contain data, two contain the specifications of the model, which itself is drawn from a hypermodel defined in the application. The data generated can then be used to verify clustering methods, specifically such methods that focus on forming a hierarchy of clusters.

The generator is ran by running a Java executable archive file:

```
java -jar HCTM.jar [model name] [n] [d] [alpha 0]
[lambda] [gamma] [p] [q] [min sd] [max sd]
```

All of the parameters are mandatory.

- **[model name]:** the name for the model, generated files will be called accordingly,
- **[n]:** number of data points to be generated from the model, higher numbers give better results, must be 1 or more,
- **[d]:** dimension of the data point, can also be understood as the number of features describing the data, in the model, must be 1 or more,
- **[alpha 0]:** the α_0 parameter for the Tree Structured Stick Breaking Process, any value in $\mathbb{R}+$,
- **[lambda]:** the λ parameter for the Tree Structured Stick Breaking Process, any value in $\mathbb{R}+$,
- **[gamma]:** the γ parameter for the Tree Structured Stick Breaking Process, any value in $\mathbb{R}+$,
- **[p]:** the p parameter for the IRVKernel, any value in $\mathbb{R}+$,
- **[q]:** the q parameter for the IRVKernel, any value in $\mathbb{R}+$,
- **[min sd]:** the minimum standard deviation possible for a Normal distribution describing every feature in a node, exists to eliminate certain numerical errors, suggested value 0.05,
- **[max sd]:** the maximum standard deviation possible for a Normal distribution describing every feature in a node, exists to set a starting point for standard deviations, suggested value 10.0.

2 The header

Each time a new set of data is generated two files with headers are written. One of the files is named **[model name].h.tex** while the other **[model name].h.txt**. These files are equivalent, though one of them is formatted as a LaTeX source file (more comfortable for human reading) while the second is a pure text file (more comfortable for automatic reading). These files contain details about the generated hierarchical structure, the relationships between nodes and the parameters of those nodes. The files are roughly divided into four segments, the fourth segment repeating for each individual node in the model. These four segments are the topic of the following four subsections of this document.

2.1 Tree data

The first segment gives the most general view of the tree.

- **Partition distribution:** Each hierarchical structure is lazily drawn from a Tree Structured Stick Breaking Process, the parameters for the process are given here, the α function is defined as $\alpha(\epsilon) = \alpha_0 \lambda^{|\epsilon|}$;
- **Kernel type:** The Kernel defines the relationship between the parameters of a child note in relation to its parent as well as the parameters of the root node, here the class for the used Kernel is listed, more information is given in the following subsection;
- **Root node name:** The name of the root node is given here.

It is important to consider the effects of the partition distribution on the resulting hierarchy.

The alpha function ($\alpha(\epsilon) = \alpha_0 \lambda^{|\epsilon|}$) controls the average density of data per level. Let us consider four separate cases:

- Both values high (above 1) leads to extremely sparsely populated deep trees, this combination of parameters is not recommended;
- Both values low (below 1) leads to densely populated shallow trees, data is either evenly distributed or grouped closer to the leaves;
- α_0 high, λ low leads to shallow trees, data more dense at lower levels of the tree;
- α_0 low, λ high leads to deep trees, the data generally grouped close to the top of the tree.

When both parameters are close to 1 the general structure of the tree is hard to predict.

The gamma (γ) parameter on the other hand is simpler. It controls the average number of children a node will have. Hight values (generally above 1)

lead to trees with more children per node. Lower values (generally below 1) lead to trees with less children per node on average.

Do note that all three values interact together, the alpha function controls the total data density per level of the hierarchy. As such the data is additionally split up between the children of each node as dictated by the gamma parameter.

2.2 Kernel data

The second section describes the Kernel used to generate the model. At the moment the only used Kernel is the IRVKernel and the description here applies only to the IRVKernel.

- **Distribution:** The parameters of the standard deviation distribution for the Kernel, the function of the distribution is shown at the end of this subsection;
- **Feature number:** The dimension of the model;
- **Feature space:** The space of the features, for the IRVKernel this is a Cartesian product of real number spaces, or specifically $\mathbb{R}^d$, where d is the number of features.

For models built using the IRVKernel a given node is described by d Normal distributions, one for each feature. For the root node the mean of the distribution for each feature will be 0 and the standard deviation will be the one given as the maximal possible standard deviation when running the generator.

For all children the mean of the child ($\mu_{child,i}$) for a given dimension i will be drawn from the parent's distribution for that dimension ($\mu_{child,i} \sim Normal(\mu_{parent,i}, \sigma_{parent,i})$. The child's standard deviation ($\sigma_{child,i}$) for a given dimension will be scaled by ν based on the parent's ($\sigma_{child,i} = \nu \times \sigma_{parent,i}$), where the scaling factor is taken from the Kernel distribution ($\nu \sim Beta(p, q)$). This is done individually for each dimension, the scaling factor being drawn separately in each case.

The ν value will control the rate at which the child's standard deviation declines compared to the parent's. The average value of ν is $\frac{p}{p+q}$, meaning that the bigger q is compared to p the faster the standard deviation will decline. The higher both values are the more stable the rate of decline will be, while if they are both low the drawn values of ν will be more diverse.

2.3 Node data

The third section of the header files holds only one value, **Number of nodes** which is indeed the number of nodes in the model. Subsection 2.4 repeats this many times, giving information on each of the nodes present in the model.

2.4 Specific node data

This fourth section of the file is repeated for each individual nodes, describing them and their relationships with other nodes. The first node listed is always the root node, it is also the only node without a parent and the node specified in the first section to be the root of the tree. The nodes are given in depth first order.

- **Node name:** The name of the node, functionally this allows us to determine which node a point of data belongs to;
- **Entry probability:** Probability of a point of data entering the sub-tree starting at this node when descending from the parent node, symbolically assumed to be 1 for the root node;
- **Retention probability:** Probability of a point of data staying in the node upon entering its sub-tree;
- **Associated nodes - Parent:** The parent node of the current, or **N/A** for the root node;
- **Associated nodes - Children:** A listing of the current node's children, or **N/A** if the node is a leaf;
- **Distributions for features:** A listing of distributions corresponding to each dimension of the model, for the given node, listed in order, and given by name and with all parameters specified, the values of features for the data points are drawn from these distributions;
- **Number of data points:** The number of data points generated by the model that belong to this specific node.

3 The data

In addition to the above files, two data files are generated for a model. The first one is called **[model name].csv**, the second file is called **[model name].r.csv**. Both files have contents formatted in the same way. They are coma separated value files and each line corresponds to one point of data generated from the model. Each line contains $d + 1$ values, first the name of the node the point belongs to, then, in order, the values for each of the d features describing a point. The first file contains the data assigned to the node it was drawn from. The second file contains the data assigned to the node it has the highest likelihood of being drawn from.

In practice the re-assigned file contains less noisy data and as such we suggest using the re-assigned file for testing. Though of course it is data that has been further refined from the original generation of the model.