Inference Engine Basics

Bryan Turner

Sept, 2006

Statistical Inference is the big umbrella for a number of ideas regarding discovery of information about unknown events. Bayesian theory is the technique I am most familiar with, it is similar to Neural Networks and proceeds by updating the probability of each theory based on observations in the environment.

Bayesian Inference can be broken down into three main components:

1) The unknowns (theories), I call these Axis.
2) The system model
3) The observations

As a brief example for a movie rating system, the unknowns are the user's preferences; both the broad categories ("likes action movies", "hates pirates"), as well as the specific ones ("hates 'Pirates of the Caribbean"). The system model is a graph of interdependence among semantic elements, such as "pirate movies usually contain action", and "love stories are usually drama", etc. The observations are specific data points pertaining to the model, such as "Jimmy rated Pirates of the Caribbean 4 stars".

The output of the system is a set of probabilities attached to the nodes which is the knowledge the system has inferred based on the observations. This knowledge can be extracted at a very general level ("Jimmy rates 94% of action movies 3, 4, or 5 stars") or a very specific level ("There is a 76% chance that Jimmy will rate 'Super Bloody Mayhem 6: Revenge of the Throat Cutters' 4 stars").

Common errors which occur due to improper evidence, poor models, etc. include results such as "Jimmy hates Pirates", when actually Jimmy just hates Johnny Depp. Given adequate models and enough evidence, these errors tend to be reduced.

The output can be munged into various formats, including Top 10, Best Of, keywords for searching, etc.

One complexity with Bayesian Inference is the "type" of inference which is occurring; the semantic content of "Jimmy rated this movie 4 stars" and "Jimmy tagged this movie 'action', 'sci-fi', and 'surprise'" are very different. In order to combine different types of observations in a single system, some complex mappings must be built. However, these problems are well known in the domain and solutions to them exist.

Recommendation Engines

As recommender technology goes there are three basic systems; aggregate ratings (amazon), collaborative filtering (movielens), and inferencing (bayesian, monte carlo, etc). They each have pros and cons, and each step increases the rating precision,

accuracy, and personalization. Businesses must make design decisions along these gradients (ie: scalability, precision, personalization). This will determine which algorithm is chosen, how much CPU it will require to calculate, and ultimately how good the recommendations are.

Model-based systems win over aggregate ratings and collaborative filtering - assuming the model is well designed. Many collaborative filtering systems perform pre- or post- filtering, clustering, and other analysis in parallel which approximates the process used in model-based systems.

Using a model-based system, it is possible to make enough observations to infer the absence of a node in the model (or an incorrect linkage). A missing node can be created, linked in, and incorporated into the model. The trick at this step is to "label" the semantics of this new node so that future observations can include it.

Over time, if this process is repeated, a complete (or "sufficient") model will be found. The literature for this process tends to work with models of less than 50 nodes, so it is dubious that it may apply to ratings networks of millions of nodes. But this feature is entirely impossible to achieve using any other system, which makes model-based solutions attractive overall.