



"turning data into dollars"

Tom's Ten Data Tips – November 2008

Neural Networks

Neural Networks (NNs) are sometimes considered the epitome of data mining algorithms. Loosely modeled after the human brain (hence the name), NNs "learn" patterns from repeated exposure to data in much the same way our brain learns to recognize patterns in the outside world. That is how NNs identify fraudulent transactions, high potential prospects, etc. They have been used to improve manufacturing process control, predict exchange and interest rate fluctuations, predict utility usage, and many other applications.

Neural Network research took off in the 60's, but died (at least in the US) around 1969 when some fundamental problems appeared unsolvable. Backpropagation and multi-layer networks needed to be invented before progress could resume. From the late 80's and on, breakthroughs in theory and available computing power enabled this.

Incidentally, the wikipedia resource on NNs is not very helpful, and contains several questionable statements.

1. Neural Networks Are An Elaborate Kind Of Regression Model

Although the way convergence to an "optimal" model takes place is quite different, in essence a NN is 'just' a very special and elaborate case of a regression model. Mathematically, a NN is described in much the same way as a linear combination of terms. In the case of a NN model this will include complex interactions and higher order polynomials.

The way a model is derived from the data is rather different for a NN compared to regression models. Regression is based on traditional statistics, where some criterion in the data set (e.g.: OLS) is minimized. Even in more elaborate stepwise procedures, the process will converge to *one* solution. NN's work rather differently. Backpropagation is the most commonly used procedure and works by running through test data, record by record, and for each record the prediction is compared to the actual target. When they differ, the weights in the network are adjusted a bit, and the next record is fed through the NN. This way, several iterations through the same training records eventually lead to ever decreasing adjustments of the weights.

When the test set is then used, the optimal (leading to overall best accuracy of the NN for the entire test set) number of iterations is determined. At that stage only, the final model is established. All sorts of random fluctuations (like ordering of records and choice of starting weights) make this an essentially unreproducible, but nonetheless sufficiently robust process. NNs tend to generate highly accurate predictions.

2. Neural Networks Come In Directed And Undirected Versions

Although NNs are commonly associated with directed (or supervised) data mining tasks, they *can* also be used for undirected tasks. In the undirected mode, NN's are often called "Kohonen Networks" or "Self Organizing Maps" after the Finnish professor Teuvo Kohonen (1934) who invented this as a special case of a clustering algorithm.

3. Neural Networks Can Model Functions Of Arbitrary Complexity

It has been proven mathematically that given enough training data, a NN can be made to "fit" any function of arbitrary complexity. There are *two* key elements in that phrase: "function" and "arbitrary complexity". The relation between input variables and the output needs to be a function which simply means that for any given array of input values only *one* output value is or can be predicted. Given this mathematical restriction in the relation between inputs and output, the *form* of the function can take *any* shape, no matter how quirky.

4. Careful Preprocessing Can Get You From A Good To An Extremely Good Model

NNs are said to deal well with "noisy" data. That is partially true, they are certainly less vulnerable to missings and outliers than regression. However, if there is a business case to invest in trying to get the best possible model, good preprocessing will likely give you that edge.

Missings are best imputed rather than lumped on to a single category (see also last month's newsletter on missing data). NNs *always* require scaling of inputs, that is mapping all variables on a 0 to 1 (or sometimes -1 to +1) range. Although technically not "necessary", you can really 'help' the NN by calculating meaningful derived variables like ratios or sums, etc. Normalization of an array of variables can be beneficial, in particular when they are 'known' to belong together. Nominal variables with high cardinality (like ZIP-codes) benefit from

organizing in a taxonomy. And last, but certainly not least, when mapping categoricals to a range, insightful scaling can dramatically improve their worth (e.g.: mapping on 0.1, 0.22, 0.9 rather than 0.1, 0.5, 0.9).

5. Preparing Date And Time Variables Requires Multiple Columns

In particular NNs (as opposed to just about any other data mining algorithm) benefit greatly from multi-dimensional unfolding of date and time variables. What this means is that the cyclical nature of date and time is explicitly represented. The fact that 23:55 is actually very close to 0:05 isn't obvious to the NN without the proper representation.

Dates have other features, like which day of the week or month, and number of days before or after a "reference date" like holidays, etc. It is fundamentally impossible to represent all these features in just one column, hence multi-dimensional representation.

6. Good Data Preparation Saves Not Only Time But Also "Data"

Although it is technically true that a NN can learn the "proper" patterns in data when they are poorly prepared, this empirical fact ignores an omnipresent reality. In most, if not all business problems, although there may be enormous volumes of data, it is still usually the case that one of the target categories you are predicting is relatively scarce. Every year, about 50 Bn credit card transactions are screened for fraud. Fortunately, there aren't many *fraudulent* ones. When predicting loan defaults, there are few defaulters compared to "good" credit risks, etc.

Although given *enough* data the NN will learn the critical patterns anyway, both the learning cycles go faster (training converges quicker), but more importantly the predictive accuracy given a *limited* volume of data will be (much) better when the data are well prepared.

7. Use Softmax Scaling To Handle Outliers And "New" Values

To prevent outliers from exerting too much influence on the model, it is good practice to scale these values back using some kind of logarithmic transformation. This keeps the impact of these records in check.

Another issue with scaled variables (see also tip# 4), is that the original range in the training data is brought back between 0-1, but it is always possible that new data may contain values *outside* the range observed in the training data. To ensure such records can be scored you need to take precautions. Softmax scaling reserves 0.05-0.95 for the “expected” range (which was observed in the training data), and the top and bottom 5% use a logarithmic transformation so that every value from $-\infty$ to $+\infty$ will fit into the 0-1 range.

8. Reverse Engineer NNs For Insight

One of the main drawbacks of NNs is the opaque nature of the prediction. nobody can truly “understand” how the NN derives its prediction from the data. Yet it is adamant that *every* prediction be accompanied by at least *some* insight. This is for two reasons. To safeguard against the occasional blooper when the data was trying to pull tricks on the data miner (it can be vicious!), do a sanity check. But also provide insight to inform and educate business partners which will grow their confidence in the model and might inspire innovations in business practices.

Because the NN itself cannot be understood, this is achieved through reverse engineering. You feed data through the NN, and let the classified records become a new training set. This training set “displays” the characteristics of the NN model, which may now be displayed with a decision tree, regression model, profiling, etc. There is also the possibility of sensitivity analysis, to see which changes in input variables have the greatest impact on the prediction (determine the most ‘important’ variables).

9. Network Topology (Largely) Determines Model Accuracy

The topology of a NN is the layout of connections between input variables, one or more so-called “hidden layers”, and the output variable(s). To complicate matters further, even the response functions that connect all these nodes are variable, and need to be chosen wisely. Besides the number of hidden layers and how many neurons they’re composed of, you need to pick a feed forward, limited recurrent or fully recurrent layout. There is no convincing, theoretically sound evidence yet, as to how one best determines the topology of a NN. But there is an enormous body of research to tap into.

Fortunately, most modern NN tools have functionality built-in that will empirically run through a number of different layouts. This saves the

miner from time consuming programming work. Unfortunately, and largely because of a lack of solid research to back this up, this is still more of an art than science. The good news is that most tools, even in a default mode, will do a remarkably good job of coming up with an effective topology layout.

10. A Prediction Is Only As Good As The Model

No, your author not trying to trump Yogi Berra (or Johan Crujff, for that matter) in his infinite wisdom... NN's have been used in many domains, and there appears to be a disproportionate number of applications in predicting exchange rates, stock market fluctuations, etc. Because the dynamics within these markets, and the relations between a myriad of data points today and predictions for tomorrow are ill understood, no "traditional" system can be built for this kind of forecast. We're simply not approaching any definitive model.

The reason for reliance on NNs may well be that we lack a thorough understanding of the underlying fundamentals, and therefore may feel inclined to resort to a black box model (in this case NN) we don't understand either. Maybe magic can help us 'understand' what is essentially unfathomable...