Bayesian Neural Networks for the Performance Prediction Challenge

Radford M. Neal, University of Toronto, `radford@stat.utoronto.ca`

Acronym of my entry: Bayesian Neural Networks

References: The general methods I used are described in my book:

Neal, R. M. (1996) *Bayesian Learning for Neural Networks*, Lecture Notes in Statistics No. 118, New York: Springer-Verlag.

The detailed models used are similar to those I used for two other prediction competitions, described in :

Neal, R. M. and Zhang, J. (2006) ``High Dimensional Classification with Bayesian Neural Networks and Dirichlet Diffusion Trees", in I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh (editors) *Feature Extraction, Foundations and Applications*, Physica-Verlag, Springer

Neal, R. M. (2006) "Classification with Bayesian Neural Networks", in J. Quinonero-Candela, I. Dagan, B. Magnini, and F. D'Alche-Buc (editors) *Evaluating Predictive Uncertainty, Visual Object Classification and Textual Entailment,* Springer.

Methods: I used Bayesian neural network models, implemented using Markov chain Monte Carlo.

Preprocessing: I transformed some features to improve correlation with the response. Specifically, for ADA, I squared feature 15 and took the square roots of features 25 and 32, and for GINA, I took the cube roots of all features. Non-binary features were rescaled to have standard deviation of approximately one.

Feature selection or dimensionality reduction: For all datasets except NOVA, I considered looking at the first 10 or 20 principal components instead of or in addition to the original features. I ended up using the first 20 principal components plus the original features for GINA and HIVA. For NOVA, I did not directly use features that were non-zero in less than four training or validation cases, though these features were incorporated into some constructed features, as described below. No other feature selection was done. However, in all models, hyperparameters were present that could adjust as the model discovered how relevant each of the features was to predicting the class (a method known as Automatic Relevance Determination (ARD)).

Classification: I used multilayer perceptron neural networks with at least two hidden layers of non-linear units (tanh activation function). Sometimes an additional hidden layer of units with identity activation function was added before these two, in order to effectively reduce dimensionality.

Model exploration and selection: Some exploration of various models was done by looking at properties of the data by hand, by seeing how different models trained on the training set performed on the validation set, and by looking at the models' own assessments of their expected performance on the test set. I also checked whether models trained only on the training set appeared to be well calibrated in their predictions for the validation set (eg, whether among cases that were predicted to be in class +1 with probability approximately 0.7, the fraction that were actually in class +1 was about 0.7). No calibration problems were found with any of the models tried.

Note that hyperparameters within each model can have the effect of smoothly adjusting various aspects of the model, such as the degree of non-linearity in the predictions. These hyperparameters were automatically updated as part of the Markov chain Monte Carlo procedure.

Since the focus of this competition was model selection, I submitted only one final entry (though the rules allowed up to five). The models for each dataset were chosen by hand, largely on the basis of the models' own assessments of performance. For HIVA, I averaged the predictions of three models. This process of manual model selection might work much better in real problems, when one would not have to guess at the meaning of peculiar aspects of the data such as described below for HIVA and NOVA.

Models for individual datasets: The results of choosing amongst various models were as follows:

ADA: I used a network with two hidden layers, containing 25 units and 10 tanh units.

GINA: I used a network containing a layer of 20 hidden units with identity activation function that looked at the original features. The outputs of these 20 hidden units along with the first 20 principal component values were fed into two subsequent hidden layers of 20 and 10 tanh units.

HIVA: I averaged the predictive probabilities produced by three models. One model used a layer of 10 hidden units with identity activation function to reduce dimensionality, with the values of these units being fed into two subsequent hidden layers with 20 and 10 tanh units. The other two models looked at the first 20 principal components plus a special composite input (but not the features themselves). They both used two hidden layers of tanh units (one had 10 and 5 units, the other 20 and 10). The special composite input was obtained by first selecting only those features that did not have a statistically significant negative correlation with the class (most have a positive correlation). For each case, the weighted average of these features was computed, with the weights being inversely proportional to the fraction of all cases in which the feature was 1, raised to the power 1.75. This was done because exploratory analysis of the data indicated that most features were positively correlated with the class, more so for those that were mostly 0.

NOVA: This dataset has a large number of features that are almost always zero . I eliminated those that were non-zero in less than four cases (training, validation, or test), but also used three derived features that were intended to capture any useful information contained in the omitted features (eg, perhaps cases with many such rare features are more likely to be in class +1). The network had a layer of 10 hidden units with identity activation function that looked at the common features. The outputs of these 10 hidden units along with the three derived features were fed into two subsequent hidden layers of 20 and 10 tanh units.

SYLVA: I used a network with two hidden layers, containing 25 units and 10 tanh units.

Performance prediction guess: My prediction for the class of a test case was obtained by thresholding the predictive probability of class +1 as produced by the model (averaging over many networks from the posterior distribution). The threshold was set to the fraction of cases that were in class +1. From the probabilities of class +1 for each test case, along with the predictions made, I also computed the expected number of errors in each class, and from this obtained an expected balanced error rate.

Results: I submitted only one entry (not counting entries before validation labels were released). This entry was ninth overall. Only two other entrants submitted better entries, so I ranked third in terms of entrants. My entry was the best in terms of average AUC on the test set, indicating that the predictive probabilities produced by my Bayesian methods are a good guide to the accuracy of predictions on individual test cases.

	My entry					The challenge best entry				
Dataset	Test AUC	Test BER	BER guess	Guess error	Test score (rank)	Test AUC	Test BER	BER guess	Guess error	Test score (rank)
ADA	0.9107	0.1753	0.1656	0.0097	0.1850 (4)	0.9149	0.1723	0.1650	0.0073	0.1793 (1)
GINA	0.9915	0.0418	0.0635	0.0216	0.0635 (31)	0.9712	0.0288	0.0305	0.0017	0.0302(1)
HIVA	0.7627	0.2824	0.2937	0.0113	0.2916 (4)	0.7671	0.2757	0.2692	0.0065	0.2797 (1)
NOVA	0.9878	0.0528	0.0706	0.0178	0.0706 (29)	0.9914	0.0445	0.0436	0.0009	0.0448 (1)
SYLVA	0.9991	0.0066	0.0070	0.0005	0.0069 (11)	0.9991	0.0061	0.0060	0.0001	0.0062 (1)
Overall	0.9304	0.1118	0.1201	0.0122	0.1235 (15.8)	0.8910	0.1090	0.1040	0.0079	0.1165 (6.2)

Code: I used my Software for Flexible Bayesian Modeling, which is available from my web page, at `http://www.cs.utoronto.ca/~radford/`. However, the scripts for this competition are not available, as they use features of my current development version, which has not yet been released. Scripts for the two other competitions referenced above are available from my web page.

Keywords: Bayesian learning, neural networks, PCA, Automatic Relevance Determination.