JMP Neural Network Methodology

Christopher M. Gotwalt
Director of Statistical Research and Development
JMP Division
SAS Institute

1. Introduction

In JMP9, the Neural platform has been completely rewritten to make JMP more competitive in the desktop data minining market. The legacy platform is still available for backward compatibility via scripting and through the user interface via a preference. The design goals for the project were to create a new platform with a richer set of modeling options, with much improved speed performance. Another important goal was to simplify certain decisions, such as setting the penalty parameter, by finding the optimal value of the penalty parameter for the user automatically as a part of the fitting algorithm. This monograph describes the implementation details of the algorithms and statistical methods used by the platform. Below is a summary of the new features of the Neural platform. The features that are marked (JMP-PRO) are only available in JMP-PRO.

- Faster performance through the use of multithreading.
- The early stopping rule is employed to speed the model fit and improve predictive performance.
- Automated, behind the scenes, optimization of the penalty parameter placed on the neural model parameters.
- The platform launch can take a validation column (JMP-PRO).
- The ability to fit two layer fully connected multilayer perceptron models (JMP-PRO).
- A richer set of activation functions (JMP-PRO).
- Modeling of multiple categorical responses as well as mixtures of continuous and categorical responses (JMP-PRO).

- Automated model construction through the use of gradient boosting (JMP-PRO).
- Outlier resistant modeling via an option to train the model vis least absolute deviations (JMP-PRO).
- Modeling of data with missing values in the input variables (JMP-PRO).
- Automated Johnson tranformation of continuous input variables (JMP-PRO)
- A richer set of penalty functions that can be applied to the input variables (JMP-PRO).

2. Modeling Details

The Neural platform in JMP-PRO can fit one and two layer, fully connected, multilayer perceptron (MLP) neural network models. An accessible introduction to these models for those with some statistical training can be found in *The Elements of Statistical Learning* by Hastie, Tibshirani, and Friedman (2001). The interested reader can refer there for the basic description and structure of MLP models, which will not be given in detail here. The two layer models are new to JMP and are offered only in JMP-PRO.

There are significant differences in the Neural platform between JMP9 and JMP9-PRO. The JMP9 version of the platform retains the most of the functionality of the JMP8 platform, and is similar in that it can fit single layer neural models with the tanh activation function. However, the new platform in simpler than the legacy platform in that many of the controls have been suppressed. This was done for a variety of reasons. For example, in previous versions of JMP, the user had to specify a penalty parameter to regularize the network parameters. There isn't any meaning to particular values of the penalty parameter, and other than crossvalidation performance there isn't a reason to favor one value of the penalty parameter over another. For this reason, the optimal value of the penalty parameter is found behind the scenes. Another control that is in the old platform but not the new one is the convergence tolerance. The converenge tolerance was removed because it is actually no longer relavant. The platform now shortcuts out

of the optimization algorithm using a crossvalidation based early stopping rule. This means that the optimizer won't have the opportunity to use the tolerance because it will almost never run to convergence.

There are two signficant features, the Sequence of Fits option and the Profiler confidence intervals, that are in the old platform but not the new version. Sequence of Fits was a JMP8 Neural platform option that automated the modeling process by fitting a set models with a user specified range of numbers of nodes and penalty parameters. This has been superceded by the boosting algorithms in JMP-PRO which build models several nodes at a time in a stagewise fashion, and also by the automatic selection of the penalty parameter which happens every time a model is fit. The Profiler intervals in the legacy platform did not have a valid statistical justification and were at best highly misleading. They should not have been added to JMP in the first place, and the Profiler in the new platform does not have confidence intervals.

2.1 The Neural Loglikelihood

The overall approach to fitting the models is to use penalized maximum likelihood for the estimation of the model parameters. In the case of multiple responses, separate loglikelihoods are computed for each response, and the overall loglikelihood for all the responses is the sum of the loglikelihoods of the individual responses. For continuous responses there are two different types of loglikelihoods used, the Gaussian and Laplacian. The Gaussian likelihood corresponds with least squares and is the default. The Laplacian (also known as the double exponential) is equivalent to least absolute deviations, and is the loglikelihood used in the robust option in JMP-PRO. A likelihood appoach was taken rather than the somewhat more straightforward (in the case of a single continuous response) least squares or least absolute deviations loss functions because composite loglikelihoods functions for multiple responses can be contructed by adding the individual response loglikelihoods. Adding sums of squares across continuous responses, for example, can have problems with responses that are measured on different scales, whereas summing the loglikelihoods leads to scale invariant estimates. It is even less natural to add sums of squares with multinomial loglikelihoods when there are both continuous and categorical responses.

Suppose that a data set has n observations with m input variables denoted by $\mathbf{x}_i$, and a continuous response y_i. Let the predicted value of the neural

model with unspecified structure and model parameters be $f(\mathbf{x})$. The sum of squared errors of the data given prediction formula, f, is

$$SSE = \sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2,$$

and the Gaussian loglikelihood based on the SSE is

$$L_{Gaussian} = \frac{n}{2}\left(\log(SSE) + 1 + \log\left(\frac{2\pi}{n}\right)\right)$$

Similarly, the sum of absolute devations is

$$SAD = \sum_{i=1}^{n}|y_i - f(\mathbf{x}_i)|,$$

and the corresponding Laplacian loglikelihood is

$$L_{Laplacian} = n\left(\log(SAD) + 1 - \log\left(\frac{n}{2}\right)\right).$$

The crossvalidation statistics can be computed using these formulas. In the r^2 statistics, the null model that is used for comparison is the model where $f(x_i) = \hat{\mu}$ for all i, where $\hat{\mu}$ is the sample mean if the loglikelihood is the Gaussian, and the sample median if the Laplacian likelihood is used.

When the response is categorical, the likelihood is somewhat more complicated. Suppose that the response, y, takes on values $1, 2, \ldots, k$. Under the parameterization of the multinomial distribution using the canonical log odds parameter, $\{\theta_\mathbf{j}\}_{j=1}^{k-1}$, the loglikelihood is

$$L_{Multinomial} = \sum_{j=1}^{k-1} I(y_i = j)\theta_j + \log\left(1 + \sum_{j=1}^{k-1} e^{\theta_j}\right),$$

where $I(y_i = j)$ is the indicator function of the event that $y_i = j$. The neural model is such that each θ_j is a linear combination of the uppermost hidden layer nodes and the set of parameters that correspond to θ_j plus an intercept type parameter. In this way, the prediction formula of the probability that $y_i = j$, f_j, is

$$f_j = \frac{e^{\theta_j}}{1 + \sum_{j'=1}^{k-1} e^{\theta_{j'}}}$$

for $j < k$, and

$$f_k = \frac{1}{1 + \sum_{j'=1}^{k-1} e^{\theta_{j'}}}.$$

For testing purposes, one can find the θ parameters from the prediction probabities using the relation, $\theta_j = \log(f_j) - \log(f_k)$. For the r^2 statistics for categorical responses, the null model is the one where $f_j(x_i) = \hat{p}_j$ for all i, where $\hat{p}_j$ is the sample proportion of observations where $y_i = j$.

In the case of multiple responses, the overall likelihood, L_{Total} is the sum of the loglikelihoods across all the individual responses. The likelihood contribution of rows that have missing values of some, but not all, of the responses will have the contribution of the non missing responses, rather that to have effectively dropped the entire row.

The legacy platform was only able to handle a single categorical response or multiple continuous responses, and could not handle mixed contiuous and categorical responses. When there were multiple continuous responses the loss function employed was a weighted sum of the sums of squared errors for the individual components. The weights were the reciprocal variances of the responses. This led to a non-scale invariant estimation approach that has the potential to lead to certain anomalies. For example, responses that have low variance and are independent of the inputs will have more weight than responses that have a large variance as a result of a strong relationship with the input variables. The risk would be that the low variance response with no relationship to the input variable would be overfit, while the other response is underfit due to the weghting scheme.

2.2 Design Row Generation

The first part of the computation of a neural model prediction formula for a particular combination of the input variables is a mapping of the input variables into a design matrix row vector. The values of the lowermost hidden layer nodes are an activation function applied to linear combinations of this design row vector with the parameters that correspond to the hidden layer node. In JMP-PRO there are three activation functions, tanh, $\exp(-x^2)$, and the identity transformation, which are referred to as the TahH, Gaussian, and Linear activation functions in the interface. In the standard version of JMP, only the TanH activation is available, which corresponds to how neural

models were implemented in the legacy platform. The way that each input variable contributes to the design row depends on whether the independent variable is continuous or categorical, and various modeling options in the platform. By default, continuous variables simply contribute their value to the design row in the same way that a continuous variable's contribution to the design row of a main effects linear model.

When the JMP-PRO only Transform Covariates option is enabled, a preprocessing step will happen where a Johnson Su or Johnson Sb distrubution will be fit to each of the continuous input variables. The continuous variable's design row contribution will then be the correponding Johnson transformation to a normal distribution. This preprocessing step uses maximum likelihood, but for speed purposes only 10 interations of Newton's method are done. This means that the resulting transformation may differ somewhat from Johnson transformations done in the Distribution platform. The purpose of the Transform Covariates feature is to transform the inputs to approximate normality as a way to mitigate the impact of input variables with outliers and heavily skewed distributions.

When the JMP-PRO only Missing Value Parameterization is turned on the continuous variables with missing cells will contribute two elements to the design row, rather than just one. When the variable is not missing the first element will simply be the value of the variable (or its Johnson transformed value when the Transform Covariates option is enabled), and the second element will be zero. When the variable is missing the first value will just be the imputed mean of the variable and the second element will be one. So, the second element is a zero-one valued indicator of whether the variable is missing and the first is just it's numeric, possibly transformed value, or an imputed mean. This approach is a simple minded way to make use of as much of the data as possible and build models that can provide reasonable predictions even in the presence of incomplete data.

Categorical variables contribute to the design row in the same way that they do in main effects models elsewhere in JMP, using what is referred to as the effect parameterization. The Missing Value Parameterization simply treats missing values of categorical input variables as if missing values were an additional level of the variable.

For example, suppose that X_1 and X_2 are two input variables where X_1 is continuous and X_2 is categorical with three levels, A, B, and C. Without the missingness parameterization turned on, the design row for (x, A) is $[x, 1, 0, 1]$, where x is some non-missing value of X_1. The first element

corresponds to X_1's contribution, the next two elements correspond to the contribution of X_2 and the last element is the intercept term. Similarly, (x, B) maps to a design row equal to $[x, 0, 1, 1]$, and (x, C) maps to $[x, -1, -1, 1]$. Now with the missingess parameterization turned on there are now two elements that correpond to X_1 and three for X_2, because a missing value for X_2, is now the last category. Now $(., A)$ maps to $[\bar{X}_1, 1, 1, 0, 0, 1]$, (x, C) maps to $[x, 0, 0, 0, 1, 1]$, and $(., .)$ leads to a design row equal to $[\bar{X}_1, 1, -1, -1, -1, 1]$.

During the numerical optimization of the model parameters one additional step occurs in the creation of the design rows. All the elements of the design row other than the intercept are centered and scaled by their mean and standard deviation. This is so that the penalty applied to a scaled and centered version of the parameters which leads to a better fit and is important to improve the optimization of the model. This centering and scaling operation happens behind the scenes and is completely transparent to the user.

3. Model Fitting Details

The general model fitting approach that is employed by the platform is to minimize the negative loglikelihood of the data plus a penalty function that is applied to a scaled and centered subset of the parameters in the model. Unlike the legacy platform, crossvalidation plays a central role in the fitting of the model parameters. The new platform requires a validation set to be specified by the user or will be generated automatically, while the legacy platform did not require a validation set at all. The fitting algorithm consists of an outer loop that optimizes the penalty parameter and an inner step where the objective function, which is the likelihood plus penalty function, is optimized using a quasi-Newton method, BFGS, for a particular value of the penalty parameter. At the beginning, normally distributed random starting values are generated and the penalty parameter is set to zero. After this initial fit, a nonzero candidate value of the penaly parameter is then chosen, and a univariate line search on the penalty parameter is undertaken. Over the course of this search, the model whose parameters led to the best value of the likelihood on the training set is the one that is reported by the plaform.

As the BFGS iterations proceed, the value of the likelihood function of the model on the validation data is monitored. When the crossvalidation likelihood is no longer improving, the BFGS algorithm will terminate, regard-

less of whether the BFGS algorithm has converged in the traditional sense. The parameter vector whose crossvalidation likelihood was the best over the course of the BFGS iterations is the one that is kept. This is commonly referred to as the early stopping rule, and leads both to faster performance and to models with improved predictive capacity. When, as in the legacy platform, the model is fit until convergence on the training set, the model will have often overfit the training data. This tendancy is mitigated to a great extent by retaining the parameter vector at the iteration crossvalidated the best. This typically happens within the first 20-50 BFGS iterations, whereas convergence on the training set may take hundreds or thousands of BFGS iterations.

When K-fold crossvalidation is used, the procedure described is applied to each of the K training-holdback partitions of the data for each trial value of the penalty parameter. After the model has been fit to all the folds, the validation set likelihood is summed across the folds using each of the individual fold's parameter estimates. This is the value that is optimized in the selection of the penalty parameter. The fold whose parameter estimates give the best value on the complete dataset are the parameter estimates that are kept and used by the platform. The training-holdback pair that led to the model that was the best are the pair whose model diagnostics are given in the report.

3.1 The Penalty Function

The new platform and the old one both applied a penalty to the parameters in the optimization of the model as a way to combat the overfitting problem that can happen with Neural models. In the legacy platform the user was required to specify either a value of a penalty parameter, or a sequence of values of the penalty parameter. The platform would then fit a sequence of models using the values of the penalty parameter specified. Generally speaking, the user will only be interested in the best fitting model and, importantly, the value of the penalty parameter has no meaning to the user. In JMP9 this is made simpler for the user by having the platform perform a search behind the scenes that finds the value of the penalty parameter that leads to the best fit. This is advantageous in a number of ways. First, the user no longer needs to guess a range of values for a parameter that has no meaning to them. The user will not know whether, or not, the best value of the penalty parameter is within that range. Second, as the optimization of

the penalty parameter proceeds the values of the preceding fit can be used as starting values for the next optimization step, thereby speeding up the iterations. Another advantage is that once further increasing or decreasing of the penalty parameter is no longer improving the model's crossvalidation likelihood, the algorithm can safely terminate, which also makes the fitting process faster than the Sequence of Fits option in the legacy platform.

In JMP-PRO there are a number of new penalty function options, while in previous versions and in the standard edition of JMP9 the only option was the sum of squared (centered and scaled) parameters penalty. In addition to the squared penalty, JMP-PRO offers the absolute value penalty, the weight decay penalty, $\beta^2/(1+\beta^2)$, and an option for no penalty. The penalty is applied to the subset of parameters that are neither intercept-type parameters nor parameters that lead out from the uppermost hidden later to the predicted values.

3.2 Boosting

In JMP-PRO an effcient and automatic way to fit large neural models is through the use of boosting. Boosting is reviewed in Hastie, Tishirani, and Friedman, with emphasis on boosting of tree models. In the Neural platform, boosting proceeds by selecting a base learner, a number of boosting iterations, and a learning rate whose value should be in the interval $(0, 1]$. The base learner is a fairly simple neural model, such as a single layer model with two tanh units. The base model is repeatedly fit to the data. If the response is continuous, after each iteration the predicted values of the base model multiplied by the learning rate is subtracted from the response, and at the next iteration the base learned is fit to this modified version of the response. If the learning rate were 1.0, then the boosting algorithm would be equivalent to recursively fitting a base model to residuals. The boosting iterations proceed until either the maximum number of boosting iterations is attained, or when for one iteration neither the training or validation likelihood can be further improved. The last base learner fit by the boosting algorithm that improved the validation and training likelihoods is not scaled by the learning rate. If the response is categorical, then the boosting model is additive on the log odds scale, rather on the probability scale as in traditional gradient boosting. This is done so that the overall boosted model can be expressed simply as a large single neural model. The boosting approach is intended to supplant the part of the Sequence of Fits option in earlier versions of JMP

that allowed the user to specify a range of number of nodes in the hidden later that would be fit. The approach outlined above should be both faster and lead to models with better predictive performance than the models fit using the legacy platform.

4. Crossvalidation Statistics

For each of the training, validation, and test sets a variaty of diagnostic measures are provided. These diagnostic measures differ whether the response is categorical or continuous. In the continuous respnse case, the crossvalidation statistics include a generalized r^2, an entropy r^2, the (negative) loglikelihood, the root mean squared error (RMSE), and mean absolute deviation. The r^2 measures use a comparison to a null loglikelihood, that we will refer to L_{Null}. Note the parameter estimates used in L_{Null} are always computed over the subset of the data in question, and this applies regardless of whether the response is continuous or categorical, or any of the other platform options. For example, in the validation set crossvalidation statistics report, the mean and variance estimates in L_{Null} are computed over the validation set, rather than the training set. Note that this is a change from that legacy platform which always used the estimates from the training set for the base model in the crossvalidation statistics. When the Robust Fit option is not turned on L_{Null} is the simple mean and variance normal distribution model loglikelihood. The parameter estimates are the sample mean and sample variance (using the n^{-1} rather than the $(n-1)^{-1}$ definition). When the Robust Fit option is enabled, L_{Null} is a Laplacian loglikelihood using the sample median as the scale parameter and the mean absolute deviation as the scale parameter. The generalized and entropy r^2 measures are computed using the following formulas,

$$\begin{aligned} r^2_{Generalized} &= 1 - e^{\frac{2}{n}(L(\hat{\beta}) - L_{Null})} \\ r^2_{Entropy} &= 1 - \frac{L_{\hat{\beta}}}{L_{Null}}, \end{aligned}$$

where n is the size of the dataset in question (e.g. training, validation, or test set), and $L\hat{\beta}$ is the (negative) loglikelihood of the set in question computed using the model parameters fit on the training data. Because of the way early stopping and penalty functions are applied, degree of freedom adjustments are not a natural fit for these models. As a result, the Neural

platform does not compute adjusted r^2 measures. The generalized r^2 reduces to the standard definition of r^2 in the case of the normal distrution, which is employed when the Robust Fit option is not turned on. The RMSE and mean absolute deviation are computed in the natural way for continuous responses,

$$\begin{aligned} RMSE &= \left(\frac{SSE}{n}\right)^{\frac{1}{2}} \\ MAD &= \frac{SAD}{n}. \end{aligned}$$

When the response is categorical, the missclassification rate is given in addition to the model fit measures described above for continuous responses. The misclassification rate is the proportion of times the actual level of the response is not the one assigned the highest probability by the neural model. The generalized r^2 is the scaled differently,

$$r^2_{Generalized} = \frac{1 - e^{\frac{2}{n}(L(\hat{\beta}) - L_{Null})}}{1 - e^{-\frac{2}{n}L_{Null}}}.$$

This is done so that a perfect fit will lead to a generalized r^2 that is equal to one, which would not be the case without the scaling. The definitions of the RMSE and mean absolute deviation are

$$\begin{aligned} RMSE &= (\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n_{Levels}} I(y_{ij} = j)(1 - f_j(\mathbf{x}_i))^2)^{\frac{1}{2}} \\ MAD &= (\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n_{Levels}} I(y_{ij} = j)|1 - f_j(\mathbf{x}_i)|. \end{aligned}$$

The confusion matrices record the number of times each level of the response is actually one level and predicted under the model to each of the levels. The actual observations equal to each level are along the vertical axis, and the predicted levels are along the horizontal axis. The confusion rate matrix is equal to the confusion matrix with the rows divided by their row totals.