Re: Questions about square errors

Source: http://sci.tech−archive.net/Archive/sci.stat.math/2007−05/msg00206.html

---

- *From*: Greg Heath <heath@xxxxxxxxxxxxxxxx>
- *Date*: 8 May 2007 13:59:42 −0700

On May 2, 2:34 pm, Old Mac User <chendrixst...@xxxxxxxxx> wrote:

> On May 2, 1:15 pm, "aggie2525" <aggie2...@xxxxxxxxxxx> wrote:
>
>> I was not the one who defined 9 input parameters.
>> Based upon experts in the field that I study, these 9 input parameters are
>> important to determine and control the outcome of the output.
>> However, my concern is that these 9 input parameters may not absolutely
>> independent to each other.

Take a look at the 10X10 correlation coefficient matrix and the
eigenstructure of the 9X9 input variable submatrix. The relative size
of the
eigenvalues will determine the practical rank (number of inputs that
are
not linearly dependent) of the input matrix. The components of the
eigenvectors corresponding to very small eigenvalues will reveal the
multicollinearities (near linear dependencies). Comparison with the
the components in the correlation coefficient matrix usually adds
additional
insight.

>> I actually use neural networks to find the best fit solution.
>> I divided my data into 3 different groups (one is for training, another is
>> for validating. The last one is for testing).
>> These 3 different data groups are used to make sure that I do not over−train
>> and create over−fitting solutions best suiting only
>> data that I used in training.
>
>> Thus, each iteration (which include the whole run on training data sets), I
>> should have the mean square error.
>> I will record only the neural network that produces the least mean square
>> error.

On the validation data.

> I feel that the least mean square error may not sufficient to tell us how
> well my program perform.
> I am thinking about using the correlation between the predicted and observed
> outputs to be control my neural network training also.
> Am I correct on this one?

Separately plot for training, validation, test , and combined data:
1. predicted output vs true output (display the linear correlation
coefficient)
2. error vs true output (display the mean−square error)

> In stead of one predicted output for each data set (item) feed in the model,
> I expect to be able to provide a range of outputs with
> a certain confidential level as defined by users.

> In my mind, if I can normalize this error (residual) distribution curve, I
> may be able to give users a range of outputs when users provide a
> confidential level that they want.

Typically, you won't have enough points to do that. You would probably
need
at least 10−to−20 observed errors for each true error bin.

Giving a customer a scatter plot of errors vs true output should
suffice.

> Another concern is that I am also afraid that the error (residual)
> distribution curve may be biased toward certain input patterns.
> Thus, the distribution curve of residuals may not be a good indicator to
> provide an error of each predicted outcome.

A basic assumption of regression is that the training sample is a
sufficient representative
of the population. If it is not, that is the least of your problems.

> Do you have any comment on this?
> How should I handle this problem?

Typically, confidence levels are reported based on the assumption
that
errors are Gaussian.

I'm not sure what is done if that assumption is grossly violated.

> Thank you very much.
>
> "Old Mac User" <chendrixst...@xxxxxxxxx> wrote in
> messagenews:1178127812.620791.191160@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
>
>> On May 2, 9:38 am, "aggie2525" <aggie2...@xxxxxxxxxxx>
>> wrote:
>>
>>> Hi,
>>>
>>> I am working on a model that use about 9
>>> input parameters to predict an
>>> output.
>>> Since I have about 800 data sets (each data
>>> set has 9 input data and an
>>> output).
>>>
>>> I come up with a method that I can predict
>>> an output from 9 given input
>>> data.
>>> Then, I use the model that I have to predict
>>> each output for each set of
>>> 9
>>> input data.
>>> As a result, I have a square error for each
>>> prediction.
>>> Therefore, there should be about 800 square
>>> errors
>>>
>>> My question is if it is OK that I plot all 800
>>> square errors to get their
>>> distribution.
>>> Then, from this distribution curve, I can get

a range of errors with a
specified confidential level for my
prediction.

My concern is that the square errors would
be dependent to certain input
patterns.

Thank you in advance for any help and
reply.

If you have minimized the sum of squares of the differences between
observed and predicted values, then it appears you may have invented
least squares and/or multiple regression.

It's smart to examine and study the differences between observed and
predicted values. In multiple regression, those are called residuals.

But there's a lot more to it than this. For instance, it's a good bet
that your nine predictors are to some degree correlated among
themselves. Your model may have several unnecessary predictors. If so,
then the presence of those probably degrades the capability of your
model for predicting future outcomes. The fact that it may predict
existing data fairly well is not necessarily an indicator of how well
it will predict future experiences.

Then there's the matter of the significance (or absence of
significance) of each of the individual predictors and the confidence

intervals on the estimated value of each predictor.

All of these should be taken into account before moving forward.

If you are not familiar with... and skilled in... the analysis of multivariable data, then I suggest you get some help with this project. OMU

Personally, I would back up to the beginning and use multiple regression. As one of my best friends has said, "Neural Nets are multiple regression without ethics".

The ethics involved in nonlinear regression should have little to do with the model one is using. Unfortunately, the availability of software
in the form of a black box allows the unqualified to try to obtain reliable
results by blindly using whatever inputs are available.

Inputs and outputs to any classification or regression model should be
sufficiently preprocessed and analyzed before the model is created.

In the particular case of Neural Network models I have recommended the consideration of scatter plots, clustering, PCA and linear or logistic
regression before designing Neural Networks with additional nonlinear processing nodes between input and output.

Go to Google Groups and search on

greg−heath pretraining advice

Unfortunately, most published introductions to NN design neither mention this nor recommend references in multiple linear/logistic regression.

By this he means that after
building a NN model you have no way to judge which of the predictors have merit. This is the same as saying (as in my first post)... you have no way to tell whether a certain predictor has a valid estimated value.

This is true of any regression model. Therefore, correlations among predictors and responses should be considered before the model is created.

In the case of NN design, there are algorithms that automatically delete input and intermediate processing nodes that are deemed redundant or irrelevant. Again ... these are rarely found in most readily available black−box software and are rarely discussed in elementary
NN references.

> One more suggestion. Can you write down the model that came from the NN effort. That is, just write it down in the form Output = bo + b1*Predictor1 + b2*Predictor2 + etc. ? From this you might get some insight into which predictors have valid "signs" and which do not. When variables are cross−correlated the signs of some regression coefficients may be reversed from reality. That would give insight into "what is correlated with what".

In spite of the fun I had with the Reefer last year, I do not hesitate to
reiterate that a lot of insight can be obtained by just viewing the all−variable (i.e., inputs and outputs) correlation coefficient matrix
in addition to looking at the components of the null eigenvectors of the
input variable correlation coefficient matrix.

> Or... can you get to the NN model at all?

Typically, yes. However, usually it doesn't help much when the number of inputs is not small.

Unfortunately, understanding how a black−box multi−layer perceptron NN arrives at it's error minimizing configuration of weights and thresholds
is not a task for inexperienced. As you have recommended, it is better
to go back to the linear/logistic model and other pretraining diagnostics.

Hope this helps.

Greg

.