Predicting Turbulence using Partial Least Squares Regression and an Artificial Neural Network

Valliappa Lakshmanan[1,2*]

Abstract

We employ partial least squares regression to transform the input data into components that have a high correlation with the variable to be predicted and preserve the variability in the dataset. Then, the transformed data are presented to a neural network whose output node has a sigmoid transfer function. Along the way, much data processing is done.

1. Introduction

The annual AI contest in its third year (2009) was focused on predicting atmospheric turbulence. The dataset was collected during summer months and consists of aircraft observations of turbulence. These observations were collected in an automated manner and consist of reports of eddy dissipation which is a measure of atmospheric turbulence. In the dataset, reports of eddy dissipation above 0.25 (unit unknown) were treated as indicating atmospheric turbulence.

The training dataset consisted of 136 columns and 103,990 rows. The test dataset consisted of 50,127 data rows but without the eddy dissipation or the turbulence parameters. The aim was to predict turbulence (not eddy dissipation). For more information on the dataset, please see: http://verif.rap.ucar.edu/ams.ai/

[*]Corresponding author: V Lakshmanan, 120 David L. Boren Blvd, Norman OK 73072; lakshman@ou.edu [1]Cooperative Institutute of Mesoscale Meteorological Studies, University of Oklahoma; [2]National Oceanic and Atmospheric Administration / National Severe Storms Laboratory

2. Method

a. *Preprocessing*

Attributes that were not meteorological in nature were removed from the training dataset. These were: the line number in the data set and all aircraft information (the aircraft id, time of measurement, latitude, longitude and altitude). The eddy dissipiation measurement was also removed as it would not be available in the test dataset. This left 128 columns that could be used for prediction and the 129th column in the training dataset that consisted of a binary variable – turbulent (1) or not (0).

b. *Attribute Transformation*

In order to reduce the number of variables, reduce the size of the dataset and to increase the likelihood of finding optimal weights, partial least squares (PLS) regression was carried out. This is similar to principal components analysis, but more tuned to the classification problem. In principal components analysis, the eigen vectors of the attributes matrix are found. Typically, a subset of the eigen vectors are the principal components in that they can explain a substantial fraction of the variance in the dataset. However, one problem is that much of this variance may not be necessary in order to explain a desired predictand. For example, consider using radar observations to predict the occurence or non-occurence of rainfall. The principal components of an attribute matrix consisting of reflectivity, velocity, spectrum width, differential reflectivity, etc. will have many components. However, a principal component consisting of just the reflectivity is likely enough if the aim is to predict whether or not it is raining. Thus, instead of finding the eigen vectors of the attribute matrix, it might be better to find the eigen vectors of the *covariance* of the attribute matrix and the predictand. This is what PLS regression does. For a more rigorous mathematical explanation of PLS, why it can be used to prune attributes and why it tends to require fewer components than PCA, please see Hastie et al. (2001); Mevik and Wehrens (2007).

The Root Mean Square Error (RMSE) of predicting variable 129 (whether or not turbulence was identified) based on components that consist of linear combinations of the first 128 variables is shown in Figure 1. As can be seen, only six components are required; beyond that, there is very little information that would go into the turbulence prediction problem.

The attributes with non-zero weights of the six most significant components are listed below:

1. STONE_Linear

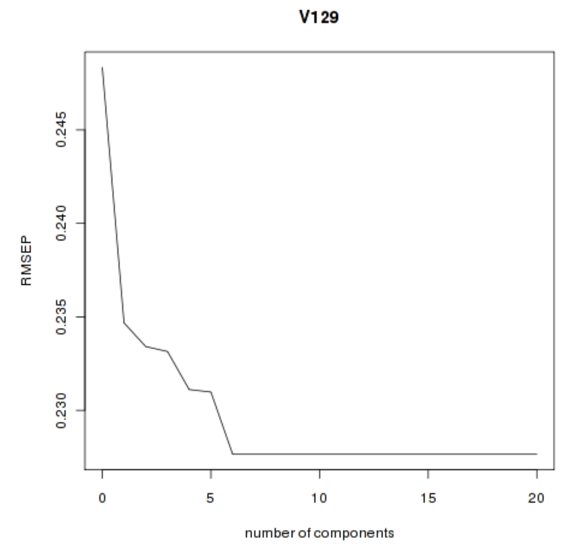

Figure 1: Only six components are required to predict turbulence.

2. RPVORT_Linear and RSTAB_Linear
3. RICH_Linear, ROL_Linear, RPVORT_Linear, RSTAB_Linear, and SATRI_Linear
4. ROL_Linear, RSTAB_Linear and SATRI_Linear
5. PRES_SFC_Linear, ROL_Linear, RSTAB_Linear, and STONE_Linear
6. dbz_DNGood40, dbz_DNGood80 NSSL_DBZ40_Wedge0_5_mindistance, nssl_18dbz_top_DNGood
   PRES_SFC_Linear, ROL_Linear, RSTAB_Linear, SMHGT_Linear, STONE_Linear, and MSLMA_MSL_Linear

These, then, are the most significant attributes in the dataset (these are not in order of importance):

- dbz_DNGood40
- dbz_DNGood80
- NSSL_DBZ40_Wedge0_5_mindistance
- nssl_18dbz_top_DNGood0_80

- PRES_SFC_Linear
- RICH_Linear
- ROL_Linear
- RPVORT_Linear
- RSTAB_Linear
- SATRI_Linear
- SMHGT_Linear
- STONE_Linear
- MSLMA_MSL_Linear

It would be possible to present these significant attributes to the neural network, but we found better results if the input data were first transformed into the vector space defined by the six most significant components i.e. the dot product of the data row with each of the eight components was computed. Thus, the transformed data has 6 inputs and one output (turbulent or not).

c. *Neural Network Training*

The transformed data consisting of 6 inputs and one output were presented to a neural network. For simplicity, the neural network architecture was held constant at 6 input nodes, 3 hidden nodes and 1 output node. The input and output nodes had a sigmoid transfer function while the hidden nodes had a tanh transfer function. The error measure that was minimized was the cross-entropy. Bishop (1995) shows that using a sigmoid output function and cross-entropy yields an output that is a probability from the frequentist standpoint.

The 103,990 input rows were randomly assigned 2:1 into a training and validation set i.e. 69327 rows were used for training and 34663 rows were used for validation. The validation dataset was used to stop the neural network training when the network started to overfit. Even though we found that the neural network in this case did not overfit (see Figure 2) and *early stopping* (Bishop 1995) using the validation dataset was not necessary to prevent overfitting, we retained the use of the validation set.

We found that the transformed data had lots of rows where all the inputs were zero. Of the 69,327 rows used for training, 68,115 were all-zero observations. Of the rest, 400

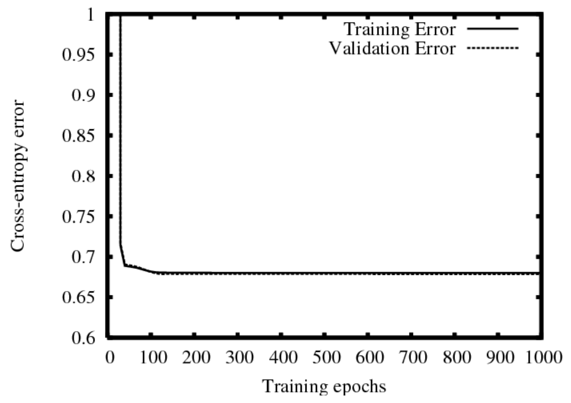

Figure 2: The neural network did not overfit.

corresponded to turbulence and the remaining 828 to no turbulence. About 6% of the all-zero observations were turbulent. The all-zero observations were removed from the training and validation sets. The desired prediction for such all-zero cases is a probability of 0.06, so this value will be output at run-time for any row whose transformed values are all zero. Now, the training set consists of just 1228 rows: 828 are no-turbulence observations and 400 are turbulence observations. The turbulence observations were repeated once so that there would be an approximately equal number of turbulence and no-turbulence observations. Thus, the transformed training set consisted of 1628 rows: 828 no-turbulence and 800 turbulence. This process of removing all-zero observations and repeating turbulence observations was repeated for the validation dataset also.

Then, the neural network with 3 hidden nodes was trained on the modified training data and early stopped using the modified validation data. The resulting training error is shown in Figure 2. The performance of the resulting neural network on the validation set is shown in Figure 3.

3. Result and Discussion

It can be noted from Figure 3 that the neural network has not been able to make much headway against the data – its outputs are all clustered between 0.4 and 0.6. The network was not able to separate out the no-turbulence observations from the turbulence observations very well. This may be because only 1-2% of the provided observations were useful data: recall that we had to throw away the all-zero observations and that these all-zero observations consisted of 98-99% of the data. A better prediction algorithm may be pos-

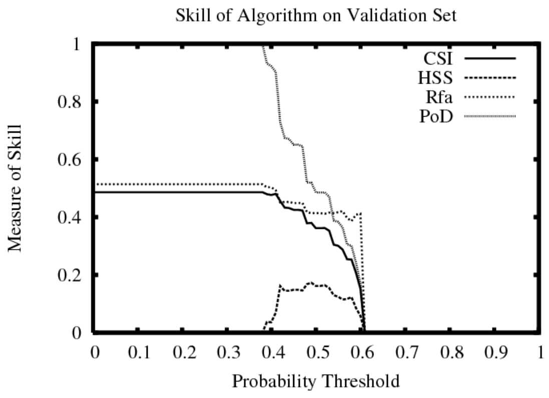

Figure 3: The neural network outputs are clustered between 0.4 and 0.6 indicating that the neural network was not able to separate out the two classes very well.

sible, but only if there are more usable observations of no-turbulence *in the presence of weather*.

The provided test dataset was preprocessed similar to the way the training dataset was – non-meteorological information was removed. Then, the test data set was transformed into the eigen space of the training set using the same eight principal components. Rows in the transformed dataset that were all zero were assigned an output probability of turbulence of 0.06. For other rows, the transformed 6 variable set was presented to the neural network and the output of the neural network was assigned for that row.

In the absence of truth information for the test data, the only metric that can be computed is sharpness. It is shown in Figure 4. Reliability, for example, would require knowledge of the ground truth.

Acknowledgements

Funding for this research was provided under NOAA-OU Cooperative Agreement NA17RJ1227. The author also thanks Gillian Peguero and John Williams for organizing the contest this year. This dataset provided me with the motivation to learn to apply PLS regression.

References

Bishop, C., 1995: *Neural Networks for Pattern Recognition*. Oxford.

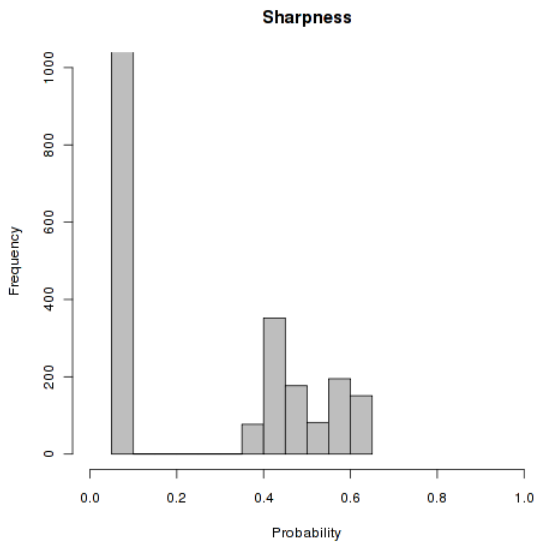

Figure 4: Sharpness of the output of the neural network on the test dataset.

Hastie, T., R. Tibshirani, and J. Friedman, 2001: *The Elements of Statistical Learning*. Springer, 66-72 pp.

Mevik, B. and R. Wehrens, 2007: The pls package: Principal component and partial least squares regression in R. *J. Statistical Software*, **18**, 1–24.