Using a Two-Layer Neural Network and Physicochemical Properties to Classify Glass for Forensic Analysis

Michael Arango
mikearango@gwu.edu

Mark Barna
mark.barna@gmail.com

Paul Brewster
pfbrewster@gmail.com

June 25, 2017

Contents

1 Project Proposal

In this study, we will use the physicochemical properties of glass to determine whether or not a given glass sample was taken from a window. This is a fundamental problem in forensic analysis as it is highly unlikely that glass fragments will be found on people unless they have been present at the time glass breaks. Glass analysis is of vital importance in forensic science as it allows us to test if the glass fragment found on a person is the same as the glass at a crime scene. Since glass is made up of several raw materials and certain elements impart specific properties, we can find out a lot about the glass if we analyze the chemical composition.

The dataset we chose for analysis was made available for download from the UCI Machine Learning Repository and was created by the USA Forensic Science Service. There are 214 observations of 9 different features along with 214 targets that specify whether the glass sample came from a window or not. While we would like more data to train a neural network, we believe the dataset is large enough for our purposes. It is difficult to know before we train a neural network if we have enough data, but the amount of data required is directly related to the complexity of the underlying decision boundary we are trying to implement. We won't know how complex the decision boundary we are trying to approximate is until we train the network, but we feel confident using the dataset as many others have used the dataset and found robust results. Several other papers in the literature use much more complex methods than we will employ and have not found the size of the data to be an issue.

We have chosen a two-layer perceptron network with tangent-sigmoid transfer functions in the hidden layer and *softmax* transfer functions in the output layer. This is a fairly standard network for pattern recognition. Moreover, we will use the *Scaled Conjugate Gradient (SGD)* algorithm to train the network as it is good for pattern recognition problems in which the output layer uses a non-linear transfer function. Since we do not expect the training error to converge to zero, we implement early stopping criteria to prevent overfitting. Lastly, we use *cross-entropy* as our performance index since our targets take on discrete values and it is the optimal performance index for pattern recognition networks that use the *softmax* transfer function in the output layer.

Two different frameworks will be used to implement the neural network. First, we will use the Neural Network Toolbox, specifically the Neural Network Pattern Recognition Tool (`nprtool`) train, validate, and test our network. We use this framework to start with a simple graphical user interface to quickly ensure our specified network architecture is appropriate and to get baseline performance statistics. Then, we will replicate the analysis in Python to gain practical experience building network architectures in a scripting language. Note that since the goal is practical experience, we will not be leveraging the power of the *scikit-learn* package (`sklearn`) for this exercise.

Several reference materials will be consulted to obtain sufficient background knowledge of the subject at hand. First, we plan on doing a thorough review of the forensic chemistry and geology literature to understand the reasons for using physicochemical properties to classify glass. Then, papers on glass analysis will be examined to supplement background knowledge with experiential knowledge.

Considering our problem is one of pattern recognition, a confusion matrix will be used to assess the accuracy of our model and the *false postive* (Type I error) and *false negative* (Type II error) rates. Further, the *Receiver Operating Characteristic (ROC) curve* will be used to compare the true positive rate to the false positive rate. This will help us gain additional knowledge of the predictive power of our network.

We plan to finish our research and submit it by Wednesday, June 28, 2017.

2 Introduction

An overview of the project and an outline of the report (I like to write intro after I finish a project).

2.1 Literature Review

3 Description of the Dataset

The dataset was made available for download from the UCI Machine Learning Repository and was created by the USA Forensic Science Service [MA94]. The purpose of this dataset is to use physicochemical properties to classify whether a certain glass fragment comes from a window or not.

3.1 Inputs

The matrix of inputs contains 214 observations of 9 variables and there is no missing data. Of these variables, eight of the nine measure the percent weight that a given elemental oxide makes up of the total glass sample weight. All the eight elements except silicon are classified as metals on the periodic table of elements. Sodium and potassium are alkali metals whereas magnesium, calcium, and barium alkaline earth metals. Aluminum and iron are classified as poor metals and transition metals, respectively. The last variable in the input matrix represents the refractive index which measures the speed of light in a transparent medium and is known as Snell's law. It can be represented formulaically as the ratio of the velocity of light in a vacuum to the velocity of light in the glass itself: $n = \frac{c}{v}$. A more thorough description of target variables is as follows:

Refractive Index: measures the ratio of the velocity of light in a vacuum to the

velocity of light in the glass itself

Sodium: represents the percent weight in sodium oxide (Na_2O)

Magnesium: represents the percent weight in magnesium oxide (MgO)

Aluminum: represents the percent weight in aluminum oxide (Na_2O)

Silicon: represents the percent weight in silicon oxide (Al_2O_3)

Potassium: represents the percent weight in potassium oxide (K_2O)

Calcium: represents the percent weight in calcium oxide (Ca_2O)

Barium: represents the percent weight in barium oxide (Ba_2O)

Iron: represents the percent weight in iron oxide (Fe_2O_3)

3.2 Targets

The matrix of targets has 214 observations, one for each observation in the training set, where a given target is denoted by $\begin{bmatrix}1\\0\end{bmatrix}$ if the glass sample comes from a window and $\begin{bmatrix}0\\1\end{bmatrix}$ otherwise. Note that the targets are two-dimensional instead of the more common one-dimensional binary encoding. This two-dimensional encoding allows us to have only one neuron firing at a time and tends to result in marginally better performance.

4 Description of the Network Architecture and Training Algorithm

Once the data is preprocessed, the next step is to decide on or create a network architecture. The basic network architecture is determined by the problem we wish to solve. Once the basic network architecture is determined, we decide how many layers, how many neurons in each layer, how many outputs the network should have, and what kind of performance index function we should use for training [Dem+14].

4.1 Network Architecture

The standard neural network architecture for pattern recognition problems is the multilayer perceptron with tangent-sigmoid transfer functions in the hidden layers and *softmax* transfer functions in the output layer. For most problems, including a fairly simple one like ours, one hidden layer usually suffices. Thus, we will implement a two-layer

perceptron. If the results of the network are unsatisfactory after training and testing with one hidden layer, we will retrain with an additional hidden layer, but we do not anticipate having to do this. The *tansig* transfer function is usually preferred to the *logsig* transfer function in the hidden layers since it produces outputs (which are inputs to the next layer) that are centered near zero, whereas the *logsig* transfer function always produces positive outputs.

We also need to select the number of neurons in each layer. The number of neurons we use in the output layer should be the same as the size of the target vector. In our case, this means we should use two neurons in the output layer. On the other hand, the number of neurons we use in the hidden layer is directly proportional to the the complexity of the decision boundary being implemented. Since we do not know the complexity of the decision boundary needed to classify these glass samples before training, we begin with ten neurons, which may be more than we need, and leverage early stopping techniques to prevent overfitting [Dem+14].

Now that we have chosen a network architecture, we can calculate the network output. The output from the hidden layer (the input to the output layer) can be calculated as

$$\mathbf{a^1} = \mathbf{logsig}(\mathbf{W^1 p} + \mathbf{b^1}),$$

while the output from the output layer is

$$\mathbf{a^2} = \mathbf{logsig}(\mathbf{W^2 a^1} + \mathbf{b^2}).$$

The network architecture can be seen in *Figure 1* below.

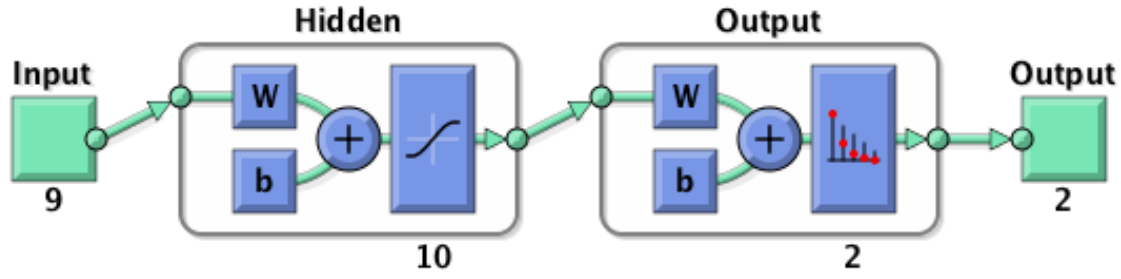

Figure 1: Two-Layer Perceptron Used to Classify Glass Samples

In the 1950's, Frank Rosenblatt developed the perceptron learning rule for training networks to solve pattern recognition problems. While his developments were monumental at the time, we only implement the perceptron learning rule when the input vectors are *linearly separable*. Widrow and Hoff's learning rule suffers form the same disadvantage. When we have more complicated decision boundaries we are trying to implement and have multiple layers, a more robust learning rule ought to be used. This problem prompted

Paul Werbos to derive the backpropagation algorithm in his thesis in the 1970's. Since then, multilayer perceptrons trained by the backpropagation algorithm have become the most popular neural network. [Dem+14].

4.2 Training Algorithm

We chose to use the Scaled Conjugate Gradient (SCG) algorithm to train our network as it is very efficient for pattern recognition problems. For multi-layer networks, the Levenberg-Marquardt algorithm is often used, but it does not work well for pattern recognition as the transfer function in the output layer is operating outside the linear region. The scaled conjugate gradient algorithm is a type of backpropagation.

5 Experimental Setup

5.1 Data Preprocessing

5.2 Implementation of the Network

5.3 Performance Index

6 Results

7 Conclusion

8 Figures and tables

The following is an example of typesetting a table.

```
\begin{table}
\caption{Table caption text.}
\label{key}
The table matter goes here.
\end{table}
```

As always with LaTeX, the `\label` must be after the `\caption`, and inside the figure or table environment. The reference for figures and tables inside text can be made using the `\ref{key}` command.

8.1 Equations

For example, if you type

```
\begin{equation}\label{eq1}
\int^{r_2}_0 F(r,\varphi){\rm d}r\,{\rm d}\varphi = [\sigma r_2/(2\mu_0)]
\int^{\infty}_0\exp(-\lambda|z_j-z_i|)\lambda^{-1}J_1 (\lambda r_2)J_0
(\lambda r_i\,\lambda {\rm d}\lambda)
\end{equation}
```

then you will get the following output:

$$\int_0^{r_2} F(r,\varphi)\mathrm{d}r\,\mathrm{d}\varphi = [\sigma r_2/(2\mu_0)] \int_0^{\infty} \exp(-\lambda|z_j - z_i|)\lambda^{-1} J_1(\lambda r_2) J_0(\lambda r_i\,\lambda \mathrm{d}\lambda) \qquad (1)$$

8.2 Listings

Another frequently displayed structure is a list. The following is an example of an *itemized* list.

- This is the first item of an itemized list. Each item in the list is marked with a '•'.
- This is the second item of the list. It contains another list nested inside it. The inner list is an *enumerated* list.
  1. This is the first item of an enumerated list that is nested within the itemized list.
  2. This is the second item of the inner list. LaTeX allows you to nest lists deeper than you really should.

  This is the rest of the second item of the outer list. It is no more interesting than any other part of the item.
- This is the third item of the list.

References

[MA94] P.M. Murphy and D.W Aha. "UCI Repository of Machine learning Databases [http://www.ics.uci.edu/ mlearn/MLRepository.html], Department of Information and Computer Science". In: *University of California, Irvine, CA* (1994).

[Dem+14] Howard B Demuth et al. *Neural network design*. Martin Hagan, 2014.

A Perceptron

B Backprop

C Subroutine X