Multivariate Analysis Group Project.

November 20, 2023

Artificial Neural Networks: Classification & Group Feature Selection

Avinandan Roy (MB2307),
Aytijhya Saha (MB2308),
Debabrat Sarkar (MB2309)
M.Stat 1st year,
Indian Statistical Institute, Kolkata

Contents

1 Introduction

In the dynamic landscape of statistical modeling, the infusion of neural networks represents a paradigm shift, introducing unparalleled capabilities for discerning intricate patterns and relationships within complex datasets.

The historical trajectory of neural networks traces back to the conceptualization of the perceptron. Developed by Frank Rosenblatt in 1957 (1), the perceptron served as an early attempt to mimic the functionality of biological neurons. However, the perceptron's limitations in handling non-linearly separable data dampened initial enthusiasm.

The resurgence of interest in neural networks occurred with the advent of backpropagation in the 1980s (2). This breakthrough paved the way for the development of Multi-Layer Perceptrons (MLPs), enabling the modeling of complex relationships through hidden layers.

In recent decades, the application of neural networks has witnessed exponential growth, fueled by advances in computing power and the availability of vast datasets. Notable milestones include the success of deep learning architectures in image and speech recognition, such as AlexNet (3) and the subsequent dominance of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) in various domains.

This historical context underscores the transformative journey of neural networks from conceptual challenges to becoming the linchpin of contemporary machine learning. This project seeks to navigate the terrain of Multi-Layer Perceptrons (MLPs), a pivotal architecture within the neural network paradigm, with a keen focus on real-world implementation and the crucial facet of feature selection.

2 Basic Structure

Neural networks, as the name suggests, draw inspiration from the structure and functionality of the human brain. At the foundational level lies the perceptron, a basic computational unit that mimics the functionality of a biological neuron.

2.1 Perceptron: The Building Block

At the heart of neural networks lies the perceptron, a fundamental building block reminiscent of a biological neuron. The perceptron takes multiple input signals, applies weights to them, aggregates the weighted inputs, and produces an output through an activation function.

2.2 Activation Functions: Adding Non-Linearity

Activation functions introduce non-linearity into the neural network, enabling it to model complex relationships within data. Common activation functions include the step function for perceptrons, but more advanced functions like sigmoid, hyperbolic tangent (tanh), and Rectified Linear Unit (ReLU) are prevalent in modern architectures. These functions facilitate the network's ability to capture intricate patterns and nonlinear dependencies.

2.3 Multi-Layer Perceptrons (MLP)

The structure of an MLP is characterized by its layered architecture, involving an input layer, one or more hidden layers, and an output layer. Let's explore each component in detail:

Input Layer: Each node in the input layer represents a feature. The number of nodes in the input layer corresponds to the dimensionality of the input data.

Hidden Layers: Hidden layers are the essence of MLPs, providing the capacity to capture intricate patterns and hierarchical representations within the data. Each node in a hidden layer processes the input signals from the previous layer, applying weights and activation functions to produce output signals. The number of hidden layers and the nodes within each layer are hyperparameters that impact the network's capacity to learn complex relationships.

Output Layer: The output layer synthesizes the information processed through the hidden layers to produce the final predictions or classifications. The number of nodes in the output layer depends on the nature of the task—regression tasks may have a single node, while classification tasks may have multiple nodes corresponding to different classes.

3 Training a Neural Network Model for Classification

3.1 Forward Propagation: Making Predictions

Suppose a given dataset is described by $\{\mathbf{x}^i, \mathbf{y}^i\}_{i=1}^N \subset \mathbb{R}^p \times \mathbb{R}^c$, where $\mathbf{x}^i \in \mathbb{R}^p$ is the i th input and $\mathbf{y}^i \in \mathbb{R}^c$ corresponds to its ideal output, in the present case the class label vector. Let,

$$\mathbf{X} = (x_{i,j})_{p\times N} = \left(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\right) = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_p \end{pmatrix}$$

where $\mathbf{x}_i \in \mathbb{R}^N$ represents vector consisting of the ith feature values over the training data, and $\mathbf{Y} = \left(\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^N\right)$. We consider a single-hidden layer backpropagation neural network, i.e, a Multi-layer Perceptron. Here, N, p, h and c denote the number of data points, number of features (i.e, number of input-layer nodes), number of hidden-layer nodes, and number of classes (i.e, number of output-layer nodes), respectively.

Let $x_0^i \equiv 1 \forall i \in 1, \ldots N$ suppose that $\mathbf{V} = (v_{ki})_{h\times(p+1)}$ is the weight matrix connecting the input layer to the hidden layer, where v_{ki} is the connecting weight between the i th input node and the k th $(k = 1, 2, \ldots, h)$ hidden node. Let $\mathbf{v}_i = (v_{1i}, v_{2i}, \ldots, v_{hi})^T$ for $i = 0, 1, 2, \ldots, p$ be the i th column vector of $\mathbf{V}$ and $\mathbf{U} = (u_{lk})_{c\times h}$ be the weight matrix connecting the hidden and output layers. The lth row of the weight matrix $\mathbf{U}$ is denoted by $\mathbf{u}_k = (u_{1k}, u_{2k}, \ldots, u_{sk})^T$ for $k = 0, 1, 2, \ldots, h$. For simplicity, we combine the weight matrices $\mathbf{U}$ and $\mathbf{V}$ and rewrite $\mathbf{w}$ as an array of total $hp + p + sh + s$ number of weights. Let g and f be the activation functions of hidden and output layer nodes, respectively.

The following vector-valued functions are introduced for convenience:

$$\mathbf{z} = (1, g((VX)_{1*}), \ldots, g((VX)_{h*})^T =: G((VX))$$

$$\hat{\mathbf{y}} = (f(UX)_{1*}, f(UX)_{s*})^T =: F(UZ)$$

Note that $\mathbf{v}_i$ is the weight vector connecting the i th input node to all the hidden nodes. So, $\sum_{i=0}^{p} \mathbf{v}_i \mathbf{x}_i$ denotes the input matrix of the hidden layer and $G(VX)$ is the output of the hidden layer. Multiplied by $\mathbf{U}, \mathbf{U}G(\sum_{i=1}^{p} \mathbf{v}_i \mathbf{x}_i)$ denotes the input of the output layer, and $F(\mathbf{U}G(VX)$ is the output of the constructed neural network. Let, $\hat{\mathbf{y}}^{\mathbf{i}}$ be the output corresponding to ith input, $\mathbf{x}^{\mathbf{i}}$.

3.2 Loss Calculation: Quantifying Prediction Error

The difference between the predicted outputs and the actual outputs is quantified by a loss function. The choice of the loss function depends on the nature of the task.

The empirical square loss function of the neural network is defined as

$$E_0(\mathbf{w}, \mathbf{X}, \mathbf{Y}) = \frac{1}{2N} \sum_{i=1}^{N} \|\hat{\mathbf{y}}^{\mathbf{i}} - \mathbf{y}^{\mathbf{i}}\|_2^2 \tag{1}$$

3.3 Backpropagation: Learning from Mistakes

The backpropagation algorithm (4) (Werbos, 1974) efficiently computes the first derivatives of the error function wrt the network weights $\mathbf{w}$. Each individual component of the gradient can be computed by the chain rule; but doing this separately for each weight is inefficient. Backpropagation efficiently computes the gradient by avoiding duplicate calculations and not computing unnecessary intermediate values, by computing the gradient of each layer – specifically the gradient of the weighted input of each layer – **from back to front**. These derivatives are then used to estimate the weights by minimizing the loss function (say, E) through an iterative gradient-descent method. We start with an initial weight $\mathbf{w}^{(0)}$ and update this using the iterative gradient decent method. Let at th i th iteration our updated weight array be $\mathbf{w}^{(i)}$. Then we update it to $\mathbf{w}^{(i)}$ according to the following update rule:

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \frac{\partial E}{\partial \mathbf{w}^{(i)}},$$

where η is a parameter, known as *learning rate*.

4 Group-feature selection using neural networks

4.1 Importance of Group-feature selection

In many applications, data are obtained from multiple sources and each source may produce several features. For example, "feature-level sensor fusion" involves the extraction and integration of high-level features from raw sensor data (5) and hence the effect of a group of features determines the importance of the corresponding sensor. Group selection approach is very important in many real-life applications for reducing complexity and cost. As an example, suppose there are two sets (groups) of features, one from an MRI scan and the other from X-ray. But, for the target application one is sufficient, so the selection of the most important image modality between the two is important as it reduces the design cost and

complexity of the decision making for the target application. There are some real-life problems where data are collected from several sensors; for example, in the case of an intelligent welding inspection system, the inputs come from different sensors such as radiograph, thermograph and eddycurrent. From each sensory input signal, some features are extracted. So, use of all these sensors would increase the cost of system design as well as the decision-making time. In this kind of application, the designer always tries to reduce the required number of sensors. The electroencephalography (EEG) data are used in many applications. The problem of selection of useful channels/independent components arises in most of the applications of EEG.

So far, group feature selection (GFS) has been successfully applied in several domains, such as gene finding (6) and waveband selection (7). We can think of the conventional feature selection problem as the GFS, with one feature in each group. In other words, group feature/sensor selection is a generalized feature selection problem.

Over time, numerous research endeavors have explored group feature selection, employing regularization techniques such as group lasso (8), sparse group lasso (9), and Bayesian group lasso (10).

4.2 Group Lasso Regularitaion

Group feature selection via group lasso has been used in several studies (11; 12; 13; 14). On the other hand, (11) used group lasso for group feature selection in the context of multivariate linear regression.

Suppose, our data consists of s feature groups, containing $n_1, \cdots, n_s$ features respectively. To effectively eliminate a 'bad' group of features, we require the magnitude of every weight connecting the features of that group with all nodes in the hidden layer to be very small, or practically zero. Then, the group lasso penalty (15) is

$$GL(\mathbf{w}) = \sum_{i=1}^{s} \|\mathbf{v}_i\|_2 \tag{2}$$

where $\mathbf{v}_i$ is the vector of weights that connect all the input nodes corresponding to i th group-feature to all the hidden nodes and $\mathbf{w}$ denotes the vector of all the model parameters.

Finally, combining both the terms (as expressed in equation (1), (2)), the loss function that we consider is as follows:

$$E(\mathbf{w}, \mathbf{X}, \mathbf{Y}) = E_0(\mathbf{w}, \mathbf{X}, \mathbf{Y}) + \mu GL(\mathbf{w}). \tag{3}$$

Here, μ is the regularizing constant that determines the severity of the respective penalty terms.

4.3 Experimental set-up/algorithm

In our experiments, We have used a single hidden layer backpropagation neural network for training, but any number of hidden layers could be used in a general case. In this investigation, the sigmoid function is used as the activation function for both the hidden layer and the output layer. The training process was carried out using the gradient descent method, with a maximum of 500 iterations set for all experiments.

Every data set is first normalized using the formula: $x' = (x - \mu)/\sigma$, where x' is the normalized feature value, and μ and σ are the mean and standard deviation of the feature x, respectively.

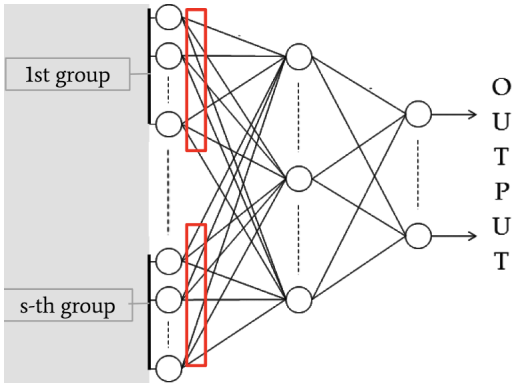

Figure 1: MLP network for GFS

In the first step, the data is randomly split into a training set (80%) and a test set (20%). Then, a 10-fold cross-validation procedure on the training data is adopted to determine the desirable number of hidden nodes. In the second step, an MLP network is trained with loss function as in equation (3). Then, we select the features (group features) with weight vectors having norms greater than or equal to $\theta = 0.1 \times \max_i \|\mathbf{v}_i\|$ to obtain the reduced training and test data. Next, we again fit an MLP network on the reduced training data and calculate the test accuracy on the reduced test data. The entire procedure is repeated 10 times independently and the average test accuracy is reported. The Python code for the implementation is available at https://github.com/Aytijhya/MVA-Class-Project-M.Stat.

4.4 Description of the datasets

We employed four real-world datasets that are collected from UCI Machine Learning Repository (https://archive.ics.uci.edu/) and are summarized in Table 1. We now briefly describe the datasets and the groups of features used in this study.

Table 1: Description of the Used Datasets

Dataset	Dataset Size	Features	Group of Features	Classes
Iris	150	4	2	3
LRS	531	93	2	10
Gas sensor	13,790	128	16	6
Smartphones	10,299	561	17	6

For IRIS data, we group sepal length and width as the first group and the remaining two features as the second group. This is a natural grouping.

The low-resolution spectrometer (LRS) dataset (16) contains 531 high-quality spectra derived from IRAS-LRS database. The Infra-Red Astronomy Satellite (IRAS) was the first attempt to map the full sky at infra-red wavelengths. This data set contains features from two bands, namely blue and red bands. These two bands consist of 44 and 49 flux measurements, respectively. Thus, LRS is a 93-dimensional data set having two groups/sensors. The question we try to investigate is whether one of these two bands is

enough to distinguish various astronomical objects (10 major classes), such as stars, galaxies, interstellar dust, etc.

The Gas Sensor dataset (17) contains information from 16 chemical sensors exposed to 6 gases at a wide variety of concentration levels. Each of those 16 sensors provided 8 features, which resulted in a total of 128 features. The classification task is to discriminate among the six different gases. It is important to know which chemical sensors are useful in discriminating among different gases for research in the field of electronic noses, which are sensor systems designed to mimic the human olfactory system for detecting and identifying odors or gases.

The Smartphones dataset (18) is used for research in the field of human activity recognition (HAR). It contains recordings of 17 signals of various subjects performing six activities (walking, walking upstairs, walking downstairs, sitting, standing, and lying) while carrying a waist-mounted smartphone with embedded inertial sensors. Features such as mean, correlation, or autoregressive coefficients were subsequently extracted from these 17 sensors. This resulted in 17 feature groups with different numbers of features in different groups. It is important to know which sensors are useful in HAR.

4.5 Experimental results

Table 2 provides a summary of the results on a set of popular data sets for group-feature selection with different values of the parameter μ. Table 2 summarizes the results on these datasets. It is evident that as we increase the value of μ, the group lasso penalty increases and consequently, a lesser number of groups of features are selected. But, notably, the impact of the reduced number of features due to higher values of the regularizers on the average accuracy is not much. In fact, with high penalty, in some cases, the test accuracy is also increased. For example, in the case of LRS, with the highest penalty ($\mu = 5$) the test accuracy is best. A similar trend is observed for the Smartphones dataset, underscoring the efficacy and usefulness of this group-feature selection method.

Table 2: Group-feature Selection Result

μ	Dataset	Test accuracy	# distinct groups	Average # groups
0	Iris	**96.30**	2	2
2		95.79	1	1
5		96.00	1	1
0	LRS	82.86	2	2
2		82.73	2	1.6
5		**83.75**	1	1
0	Gas sensor	**98.40**	16	16
2		97.61	7	5.5
5		96.23	5	4.3
0	Smartphones	92.54	16	16
2		**92.64**	7	6.1
5		92.57	6	5.6

In the case of the Iris dataset, setting μ to 2 and 5 results in the exclusive selection of group 2. This aligns with expectations since group 2 encompasses the petal length and petal width—acknowledged as the two most informative features in the Iris dataset. For the LRS dataset, it turns out that only the blue band is selected when the penalty is high ($\mu = 5$). For gas sensor data, $2, 9, 11, 13, 14$th sensors comprise the set of distinct sensors selected in 10 runs with $\mu = 5$. In the case of the Smartphones dataset, $4, 5, 7, 9, 10, 16$th sensors are selected in 10 runs with $\mu = 5$. Table 2 shows that the number of distinct groups selected at least once in 10 runs and the average number of selected groups in 10 runs are very close in all the cases we considered. This consistent selection across multiple runs suggests the importance of these specific sensors in capturing relevant information.

5 Conclusion

In closing this project report on neural networks, our journey through Multi-Layer Perceptrons (MLPs), real-world dataset implementations, and feature selection has illuminated the dynamic landscape where statistical methodologies intersect with the formidable capabilities of neural networks. As we conclude, the horizon of possibilities remains vast. The fusion of statistics and neural networks persists as a continually evolving narrative, urging us to venture into uncharted territories, enhance existing methodologies, and expand the frontiers of our comprehension.

References

[1] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Report: Cornell Aeronautical Laboratory, Cornell Aeronautical Laboratory, 1957.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[4] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Science. Thesis (Ph. D.). Appl. Math. Harvard University*. PhD thesis, 01 1974.

[5] D. L. Hall and S. A. McMullen, "Mathematical techniques in multisensor data fusion. artech house," *Inc., Norwood, MA, USA*, vol. 77, 1992.

[6] L. Meier, S. Van De Geer, and P. Bühlmann, "The group lasso for logistic regression," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 53–71, 2008.

[7] N. Subrahmanya and Y. C. Shin, "Sparse multiple kernel learning for signal processing applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 5, pp. 788–798, 2009.

[8] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[9] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *Journal of computational and graphical statistics*, vol. 22, no. 2, pp. 231–245, 2013.

[10] S. Raman, T. J. Fuchs, P. J. Wild, E. Dahl, and V. Roth, "The bayesian group-lasso for analyzing contingency tables," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 881–888, 2009.

[11] N. Pusponegoro, A. Muslim, K. Notodiputro, and B. Sartono, "Group lasso for rainfall data modeling in indramayu district, west java, indonesia," *Procedia computer science*, vol. 116, pp. 190–197, 2017.

[12] M. Yunus, S. Asep, and M. S. Agus, "Characteristics of group lasso in handling high correlated data," *Applied Mathematical Sciences*, vol. 11.20, pp. 953–961, 2017.

[13] C. Du, C. Du, S. Zhe, A. Luo, Q. He, and G. Long, "Bayesian group feature selection for support vector learning machines," in *Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, PAKDD 2016, Auckland, New Zealand, April 19-22, 2016, Proceedings, Part I 20*, pp. 239–252, Springer, 2016.

[14] F. Tang, L. Adam, and B. Si, "Group feature selection with multiclass support vector machine," *Neurocomputing*, vol. 317, pp. 42–49, 2018.

[15] H. Zhang, J. Wang, Z. Sun, J. M. Zurada, and N. R. Pal, "Feature selection for neural networks using group lasso regularization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 4, pp. 659–673, 2019.

[16] "Low Resolution Spectrometer." UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C5B02R.

[17] A. Vergara, "Gas Sensor Array Drift Dataset at Different Concentrations." UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5MK6M.

[18] A. D. G. A. O. L. Reyes-Ortiz, Jorge and X. Parra, "Human Activity Recognition Using Smartphones." UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C54S4K.