Variance-Based Feature Importance in Neural Networks

Cláudio Rebelo de Sá(✉)

Data Science Research Group, University of Twente, Enschede, Netherlands
c.f.pinhorebelodesa@utwente.nl

Abstract. This paper proposes a new method to measure the relative importance of features in Artificial Neural Networks (ANN) models. Its underlying principle assumes that the more important a feature is, the more the weights, connected to the respective input neuron, will change during the training of the model. To capture this behavior, a running variance of every weight connected to the input layer is measured during training. For that, an adaptation of Welford's online algorithm for computing the online variance is proposed. When the training is finished, for each input, the variances of the weights are combined with the final weights to obtain the measure of relative importance for each feature. This method was tested with shallow and deep neural network architectures on several well-known classification and regression problems. The results obtained confirm that this approach is making meaningful measurements. Moreover, results showed that the importance scores are highly correlated with the variable importance method from Random Forests (RF).

1 Introduction

Effectively measuring the relevance of features in Artificial Neural Networks (ANN) can foster its usage in new domains where some interpretability is required. Current studies show that there has been some effort to bring more interpretability to Artificial Neural Networks in the recent years [3,5]. However, despite the various approaches available in the literature, there is a lack of simple, and yet reliable, variable importance approaches for ANN.

The classic and most simple architecture of a feed-forward neural network is composed by one input layer, one or more hidden layers and one output layer. These layers are connected by weights and each is composed of a certain number of neurons. Each neuron in the input layer represents one independent variable, or feature, from the data. These neurons connect to the first hidden layer, which in turn connects either to the next hidden layer (and so on) or to the output layer. The neurons in this output layer represent the target variable.

During the training of ANN, the weights (which connect the neurons) are constantly being changed for better fitting the data. This process occurs for every batch of data, and it lasts until all the epochs are finished. Once this process is finished, it is natural to assume that, the higher the absolute values of the weights, the more important a variable would be [2]. However, on the other

P. Kralj Novak et al. (Eds.): DS 2019, LNAI 11828, pp. 306–315, 2019.
https://doi.org/10.1007/978-3-030-33778-0_24

hand, we also know that regularization techniques force the weights to become smaller (e.g., L1 and L2). Besides, the choice of the initialization of these weights can also interfere with their final absolute value.

In this paper, a very simple method is proposed to measure the relative feature importance (or variable importance) of ANN models. Its underlying principle assumes that the more important a feature is, the more the weights, connected to the respective input neuron, will change during the training of the model. This means that, it expects bigger changes in the weights connected to more relevant variables, independently from their absolute value. Under this assumption, by measuring the total variance of the weights connected to each input node, one should be able to measure its relative importance.

Since the weights are being changed at every batch and neural networks could easily take hundreds of epochs to be trained, it is not practical to store all the values for later computing its variance. For this reason, the running variance of each weight that is connected to the input layer is used instead. For that, an adapted version of Welford's online algorithm [9] is proposed. This algorithm updates the mean and variance of the weights, connected to the input layer, at an user defined step (e.g., per batch or per epoch).

Finally, the variances from all the weights connected to a feature are combined into a single value, which are then used for assessing their relative importance. This contrasts with most of the approaches available in the literature, because the variable importance is not measured on the absolute values of the weights of the network, but on their variance during the training.

Some well-known regression and classification datasets were used to test the proposed approach, which included one artificial dataset. Empirical results presented in this paper, using shallow and deep ANN, show that this approach holds promise and can be used to effectively assess the relative importance of variables in different datasets.

2 Variable Importance in FNN

Most approaches assess the feature importance based on the final weights of the trained neural networks [2,4,5]. One of the most well-known was proposed in 1991 by Garson [2] and it is still being used [3,8]. It basically consist in adding up the absolute values of the weights between each input node and the response variables. In other words, all the weights connecting a given input node, including the hidden layers, to a specific response variable will contribute to measure feature importance. Finally, the total score of the input nodes is scaled relatively to all other inputs.

However, we know that during the training phase, the weights of a neural network are being modified. These updates of the weight are repeated until the model reaches its final state. Therefore, we propose to measure the variance of these weights in order to get the relative variable importance based for ANN.

In [6], the authors observed that the difference between quartiles of the distributions seemed to be related with their relative importance. In this work,

a similar approach is taken, however the focus is in the variance instead of the interquartile range.

Considering the size of current ANN and the high number of epochs to train them, storing the values of the all the weights to compute the variance would be computationally expensive. However, a method proposed by Welford in 1962 [9] (See Sect. 2.1) allows one to compute and update the variance as the measurements are given, one at a time. This has the advantage that, the values do not need to be saved to compute the variance in the end.

A simple adaptation of Welford's online variance, proposed here, makes this approach much simpler to implement. It is similar to the method in [9], except that this was adapted to deal with matrices. These matrices represent the weights connecting the input layer with the first hidden layer.

2.1 Welford's Online Variance

The variance of a sample of size n is defined as:

$$S_n^2 = \frac{SS_n}{n-1} = \sum_{i=1}^{n} \frac{(x_i - \overline{x}_n)^2}{n-1} \tag{1}$$

where the corrected sum of squares SS_n is:

$$SS_n = \sum_{i=1}^{n} (x_i - \overline{x}_n)^2 \tag{2}$$

and the mean, $\overline{x}_n$, is defined as:

$$\overline{x}_n = \sum_{i=1}^{n} \frac{x_i}{n} \tag{3}$$

However, we can write the corrected sum of squares SS_n as:

$$SS_n = SS_{n-1} + \left(\frac{n-1}{n}\right)(x_n - \overline{x}_{n-1})^2$$

This way, if we replace this in Eq. 1, we can update the variance of a sample, originally with size $n-1$, by adding one more measurement to the sample, x_n [9]. This can be represented as:

$$Var(x_n) = \frac{SS_{n-1}}{n-1} + \frac{(x_n - \overline{x}_{n-1})}{n} \tag{4}$$

where n represents the total number of updates. This computes the online variance of the weights (also known as running variance).

2.2 Online Variance of the Weights

Let us define a dataset $D = \{\langle v_i \rangle\}$, $i = 1, \ldots, z$ with z instances, where v_i is a vector containing the values $v_i^j, j = 1, \ldots, m$ of m independent variables, $\{\mathcal{A}_1, \ldots, \mathcal{A}_m\}$, describing instance i.

To represent the weights between layers in ANN, we define $w_{a,b}$ as the weight connecting node a to node b. As mentioned before, m represents the number of input variables and q the number of neurons in the first hidden layer. Now we can represent the Variance-based feature Importance of Artificial Neural Networks (VIANN) score of the weights as:

$$\text{VIANN}\left(\mathcal{A}_s\right) = \sum_{k=1}^{q} Var\left(w_{s,k}\right) \times |w_{s,k}| \tag{5}$$

where t represents the total number of updates and $w_{s,k}$ the weights of the first hidden layer connected to the input $\mathcal{A}_s$. This means that the final score will depend on both the final weights and the variance of the weights during the training.

We will use the variance as in Eq. 5, to score the importance of the features. The assumption is that, the more the $w_{a,b}$ varies in the training phase, the higher the relevance of the nodes a to the prediction. When using VIANN we need to define at which steps of the training we update the variance. Several options can be considered, per iteration (after every batch), per epoch or with an user defined interval. For simplicity, in this work we update the variance of the weights at each epoch.

3 Experimental Setup and Results

In this section we explain how we setep the experiments and describe the architecture of the neural networks tested. We also present the results obtained by the proposed approach and compare with the most used algorithm to measure feature importance in Neural Networks, the Garson's algorithm. We also describe the datasets which were selected for this study.

3.1 Experimental Setup

In this experimental setup, many parameters could have been modified and studied. However, as proof of concept, we tried to make some simple and reasonable choices for the design of the architectures and its parameters.

Since we are testing both regression and classification tasks, the neural networks were built in such a way that the last layer (the output layer) can change to classification or regression mode. When the task is *classification*, the last layer will have the same number of neurons as the number of classes and an activation function *softmax*. The loss, in this case is *categorical cross-entropy*. On the other hand, when the task is regression, the last layer has one neuron and the activation is *linear*. The loss function in this case is the *Mean Squared Error*.

Besides that, two types of activation functions are tested in the remaining layers. One neural network has a *linear* activation function (NN1) and the other has the RELU (NN2). Both NN1 and NN2 have 3 hidden layer, one with 50 neurons, other with 100 and the last hidden layer with 50. Besides NN1 and NN2, we also tested the approach in a deeper neural network, DeepNN. The dimensions of the three neural networks are:

NN1: input, 50, 100, 50, output
NN2: input, 50, 100, 50, output
DeepNN: input, 500, 1024, 2048, 4096, 2048, 1024, 500, output

We note that the input and output layers are adapted according to the size of the number of features and number of classes per dataset, respectively.

In previous experiments, we observed that the final accuracy of the model, strongly affects the scoring of the most relevant features. This means that, if the accuracy is low, the importance scores tend to be misleading. For this reason, a simple procedure was used to find a more appropriate number of training epochs per dataset. An early stopping function monitored the validation accuracy in the classification datasets during training. When the validation accuracy is >95% the training stops. The maximum number of epochs was set to 1000 and the minimum to 5. In the regression datasets, the number of epochs was fixed to 100 epochs.

The optimizer used in the experiments was set to the default parameters of the Stochastic Gradient Descent (SGD) optimizer from the python package *Keras*. Due to the different dataset sizes, the batch size was adapted for each dataset. It was defined as the rounded number obtained from the division of the number of instances in the dataset by 7.

We compare the obtained scores of the variables with other approaches such as Random Forests (RF) [1] and Garson's algorithm (GA) [2][1]. We also measured the loss of the models when each feature was removed (replaced with zeros) which is the same as the Leave-One-Feature-Out (LOFO) approach. Then, the features, which after being removed, resulted in the highest loss, are considered more important. Finally, to compare the orders between the different techniques, we use the Kendall's tau correlation coefficient.

3.2 Datasets

In the experiments several classification and regression datasets from the *scikit learn* python package [7] were used (Table 1). These particular datasets were chosen to illustrate the effectiveness of this approach. All the features of the dataset were normalized to zero mean and standard deviation 1.

[1] All the results presented in this paper can be replicated using the python file in https://github.com/rebelosa/feature-importance-neural-networks.

Table 1. Datasets used in the experiments

Name	#Features	#Instances	Type	#Classes
Breast cancer	30	569	Class	2
Digits	64	1797	Class	10
Iris	4	150	Class	3
Wine	13	178	Class	3
Boston	12	506	Regr.	-
Diabetes	10	442	Regr.	-

3.3 Testing the Feature Importance

In this experimental part, we compare the ranking of the features obtained with VIANN, for NN1 and NN2. As previously mentioned, there are several approaches to measure the relative variable importance in datasets. Besides comparing with the RF variable importance measure, we also compare with the Leave-One-Feature-Out (LOFO) approach. By measuring the loss difference after removing each variable, we can sort the variables by the ones which have a greater impact.

The presented in Table 2 show the Kendall tau correlation between the different variable rankings when using the neural network with *linear* activation function, NN1. We can see that VIANN obtains a higher correlation than the Garson technique, as compared with the LOFO. Besides, the relative order of the variables seems to fluctuate much more when obtained with the Garson approach.

Table 2. Kendall tau correlation between the different variable importance techniques using the NN1 (linear activation function)

Dataset	VIANN-LOFO	VIANN-RF	Garson-LOFO	Garson-VIANN
Breast cancer	**0.43**	0.85	0.17	0.20
Digits	**0.96**	0.66	0.90	0.87
Iris	**0.99**	0.88	0.73	0.62
Wine	**0.93**	0.64	0.91	0.85
Boston	**0.95**	0.79	0.84	0.80
Diabetes	**0.86**	0.68	0.85	0.79

In Table 3 we observe a very similar behavior of the one observed in Table 2. In this case the neural network NN2 is exactly the same as before, except that it has a *RELU* activation function instead of *linear*. In general, we observe that the correlation VIANN-LOFO is worst when using the *RELU*, specially for the classification datasets.

Table 3. Kendall tau correlation between the variable importance technique VIANN and other approaches with the NN2 (RELU activation function)

Dataset	VIANN-LOFO	VIANN-RF	Garson-LOFO	Garson-VIANN
Breast cancer	**0.78**	0.76	0.43	0.61
Digits	**0.94**	0.76	0.92	0.84
Iris	0.92	0.87	**0.97**	0.81
Wine	0.88	0.50	**0.95**	0.93
Boston	**0.96**	0.81	0.76	0.60
Diabetes	**0.98**	0.90	0.64	0.60

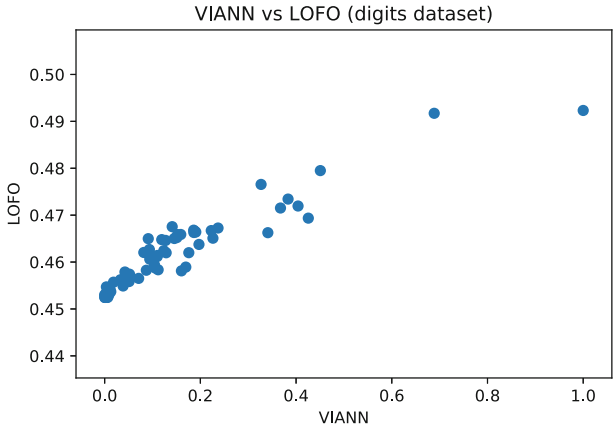

Fig. 1. Plot of the feature importance scores obtained with the NN2 using VIANN (x axis) and LOFO (y axis) in the *Digits* dataset.

In Fig. 1 we can see how the LOFO and VIANN relate in terms of feature importance scores. It seams that the variance of the weights combined with the final weights has a linear relation with the increase in the loss of the model.

3.4 Deep Neural Network

Finally, despite the small size of the dataset, we wanted to test the approach in the deep learning context. For this reason, we used a deeper network, which we refer as *DeepNN*. The results obtained are presented in Table 4. In this case, we observe that VIANN is still better at giving a more meaningful score of the variables. The exception is the *Wine* dataset where the correlation is not so high.

We observe in Fig. 2, how the scores of VIANN with the DeppNN are related with the variable importance scores of RF. Even though the two models have different biases, the importance of the variables is only slightly changed, which intuitively makes sense.

Table 4. Kendall tau correlation between the variable importance technique VIANN and other approaches with the DeepNN (RELU activation function)

Dataset	VIANN-LOFO	VIANN-RF	Garson-LOFO	Garson-VIANN
Breast cancer	**0.60**	0.45	0.22	0.37
Digits	**0.83**	0.80	0.60	0.46
Iris	**0.90**	0.98	0.73	0.49
Wine	0.41	0.43	**0.74**	0.51
Boston	**0.76**	0.86	**0.76**	0.79
Diabetes	**0.86**	0.88	0.64	0.80

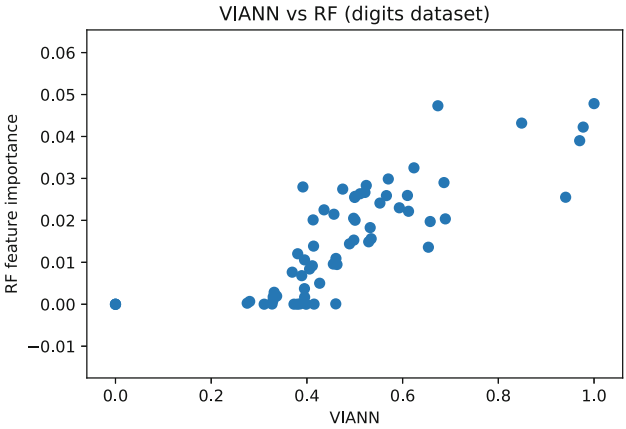

Fig. 2. Plot of the feature importance scores obtained with the DeepNN using VIANN (x axis) and RF (y axis) in the *Digits* dataset.

In general, we observe that the feature importance scores are better than the Garson's technique, which shows that VIANN has potential as a measure of importance of features in shallow and deep networks. Even considering that different features can be more or less important for different models, it does not seem likely that the features will have completely distinct relevance for each model. Therefore, we believe that VIANN is measuring some phenomenon that is closely related with the importance of the features.

3.5 Evolution of Weights During Training

Since the motivation was to use the variance to measure the feature importance, we wanted to understand if the behavior of the weights of the most relevant features was actually changing more than the others. Therefore, in this experiment we trained the NN2 and captured the weights in every iteration between the input layer and the first hidden layer during the training phase.

The result can be seen in Fig. 3, where the subplots are sorted by the relative importance of the respective inputs (higher to lower), obtained with the LOFO approach. One pattern that can be observed, is that, in fact, the weights seem to change more in the first inputs (e.g. input 12,0 and 9). On the other hand, the weights which affect less the loss, are mostly constant during the entire training. Besides that, the absolute value of the weights seems to also be higher in more important features. These observations support the motivation of this paper (Eq. 5).

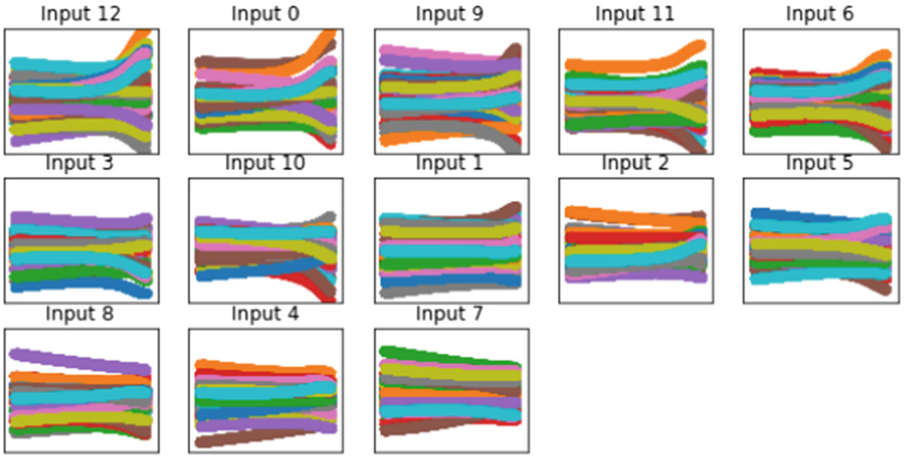

Fig. 3. Evolution of the weights (coloured lines) of the NN2, per iteration, between the input layer and the first hidden layer trained in the *Wine* dataset (x-axis: iterations; y-axis: weights)

4 Conclusions

In this work we compare the performance of a feature importance technique which is based on the variance of the weights during the training of neural networks. We compare our results with one of the most widely used variable importance techniques in ANN, Garson's algorithm. The results showed that this approach is more reliable in identifying the order of the variables which have a greater influence in the loss. In comparison to Garson's method it has the advantage that it does not require that the first and last hidden layers have the same number of neurons.

We also observed that, when the validation accuracy is low, the scores of the features can be misleading. Some tests, which are not reported in this paper, indicated that without proper regularization techniques, the variable importance scores also do not make sense.

Considering the results obtained, this approach holds promise to effectively measure the relevance of features (or even just any neuron). That is, the simplicity of VIANN makes it straightforward to measure the relevance of every node

and not only the input layer. Moreover, it can be easily extended to other neural network layers, such as recurrent or convolutional.

Since this work is only a preliminary study, it does not provide a comprehensive overview of feature importance for ANN. However, as future work we would like to study if VIANN can be used to obtain the importance of the variables on distinct ANN architectures. A thorough experimental study should be made in the future to fully understand the advantages and limitations of this approach.

Acknowledgments. I gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research.

References

1. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
2. David Garson, G.: Interpreting neural-network connection weights. AI Expert **6**(4), 46–51 (1991)
3. Heaton, J., McElwee, S., Fraley, J.B., Cannady, J.: Early stabilizing feature importance for tensorflow deep neural networks. In: 2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, 14–19 May, 2017, pp. 4618–4624 (2017)
4. Martínez, A., Castellanos, J., Hernández, C., de Mingo López, L.F.: Study of weight importance in neural networks working with colineal variables in regression problems. In: Multiple Approaches to Intelligent Systems, 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE-99, Cairo, Egypt, May 31 – June 3, 1999, Proceedings, pp. 101–110 (1999)
5. Olden, J.D., Jackson, D.A.: Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks. Ecol. Model. **154**(1), 135–150 (2002)
6. Paliwal, M., Kumar, U.A.: Assessing the contribution of variables in feed forward neural network. Appl. Soft Comput. **11**(4), 3690–3696 (2011)
7. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
8. Shavitt, I., Segal, E.: Regularization learning networks: deep learning for tabular datasets. In: Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3–8 December 2018, Montréal, Canada, pp. 1386–1396 (2018)
9. Welford, B.P.: Note on a method for calculating corrected sums of squares and products. Technometrics **4**(3), 419–420 (1962)