Deep Learning Method for Identification and Selection of Relevant Features

Dr. Vejendla Lakshman
lakshmanv58@gmail.com

Abstract
Feature Selection have turned into the main point of investigations particularly in bioinformatics where there are numerous applications. Deep learning technique is a useful asset to choose features, anyway not all calculations are on an equivalent balance with regards to selection of relevant features. To be sure, numerous techniques have been proposed to select multiple features using deep learning techniques. Because of the deep learning, neural systems have profited a gigantic top recovery in the previous couple of years. Anyway neural systems are blackbox models and not many endeavors have been made so as to examine the fundamental procedure. In this proposed work a new calculations so as to do feature selection with deep learning systems is introduced. To evaluate our outcomes, we create relapse and grouping issues which enable us to think about every calculation on various fronts: exhibitions, calculation time and limitations. The outcomes acquired are truly encouraging since we figure out how to accomplish our objective by outperforming irregular backwoods exhibitions for each situation. The results prove that the proposed method exhibits better performance than the traditional methods.

Keywords: *Feature selection, deep learning, neural networks, preprocessing, data extraction.*

1. Introduction

Variable and feature selection have become the focus of much research, especially in bioinformatics where there are many applications. Machine learning is a powerful tool to select features, however not all machine learning algorithms are on an equal footing when it comes to feature selection[1]. Indeed, many methods have been proposed to carry out feature selection with random forests, which makes them the current go-to model in bioinformatics[2]. This mainly comes from the fact that random forests are well known to be good \out-of-the-bag" algorithms and they do not need huge amount of data in order to achieve good results[3]. On the other hand, thanks to the so-called deep learning, neural networks have benefited a huge interest resurgence in the past few years.

However neural networks are blackbox models and very few attempts have been made in order to analyze the underlying process[4]. Indeed, quite a few articles can be found about feature extraction with neural networks (for which the underlying inputs-outputs process does not need to be understood), while very few tackle feature selection[5]. Furthermore, neural networks are known to require lots of data and computation time in order to achieve good performances. Since data are often hard to obtain in bioinformatics, this is already a burden for neural networks. Nevertheless, some attempts were made to select features using neural network, unfortunately most of them used very shallow networks and others were directed to very specific datasets[6].

Consider a binary classification problem with a class corresponding to a positive outcome (for example an alarm activation) and the other to a negative outcome (the alarm doesn't activate). Also consider a binary classification model which is used to classify an input (for example a motion detector) to one of the two classes[7]. For each data sample, the classification made by the model belongs to one of the following categories:

True positive (TP). This occurs if the model activates the alarm when it should have been. No error is made.

True negative (TN). This occurs if the model doesn't activate the alarm rightfully so (i.e. the alarm should not have been activated). No error is made.

False positive (FP). This occurs when the model activates the alarm although it should not have been. An error is made and leads to Type 1 error.

False negative (FN). This occurs when the model doesn't activate the alarm although it should have been. An error is made and leads to Type 2 error.

Neural network can be built in a plentitude of ways and are subject to many parameters, neural architecture being the first one[8]. Indeed, neural networks can take many forms, ranging from very shallow to very deep and very narrow to very wide. Many constraints can also be added in the architecture itself, convolutional and encoder layers are some of them. All of these parameters can be changed regarding the problem we are facing[9]. In our case we decided to limit ourselves to test our algorithms on networks with fully connected hidden layers. We did this choice since our data didn't give us a priori reasons to introduce structure into our network. Furthermore, this is the more generic and \simplest" architecture that can be found.

Dropout can be seen as an \ensemble" method for neural networks. Indeed, the principle is to train only a subpart of the network at each iteration. In fact, each neuron has a given probability to be temporarily removed" at train time. At test time, all neurons are used and their weights are adapted regarding their

Manuscript received May 5, 2024
Manuscript revised May 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.5.24

probability of being kept at training time[10]. This can be seen as training multiple networks and averaging their predictions at test time (although this is not really what happens, but it would be too costly to train multiple networks).

2. Literature Survey

Li, Y et al [1] proposed multiple formulae in order to carry out feature selection. They were separated into three categories: zero order, first order and second order methods. They were directly based on the parameters of the network while first and second order methods were respectively based on the derivative and second derivatives of those parameters.

Marbach, D et al [2] proposed to use one of the formulae mentioned into tackle deeper neural networks. Indeed, in a back-propagation method is used to compute feature importance. Let i be the neuron whose importance score we are calculating, and Ni the set of neurons in the next layer (closer to output) that i feeds into.

Montavon et al [3] gave some insight on how to associate neural activation and feature importances. The idea here consists of analyzing the activation of the neurons for each input sample and averaging over all samples, thus using each data sample values and not basing the formula only on the network's intern parameters. This technique is proposed and works as follows. Let xi be the ith dimension of the input example x connected to jth hidden neuron by w_{ji} and b_j the bias of hidden neuron j.

Unfortunately, the regularization method we used doesn't allow to select redundant features. Indeed, imagine we have two features representing the same information[11]. Since the regularization is linear, it is equivalent cost-wise to have one big and one small weights rather than two medium one (note that this is beneficial when one wants to select as few variables as possible, i.e. to solve the minimal optimal problem. To counter this (i.e. to solve the all-relevant problem), quadratic regularization (elastic net) could be introduced and would help the network selecting both of the variables in order to minimize the cost[12].

3. Proposed Method

The goal of this subsection is to give a formula to compute an importance defined as how much a given neuron contributes to the output \variability", to this end we will go through the 5 following parts :

1. Importance metric definition. This paragraph will define an importance measure of each neuron for a given data sample. This measure is based on neural activation[13]. However, they only analyze the contribution of the inputs on the first hidden layer, whereas here we propose a formula that takes the whole network structure into account.

2. Initialization. In this paragraph, a method for initializing the algorithm will be discussed. We will also give some clues on how this technique could be refined in different settings[14]

3. Back propagation. We will explain how the two first parts are put together to obtain the algorithm.

4. Results. We will show an example of importance that are obtained using this method and show that the results seem reasonable.

5. Extensions. In this sub subsection we will show that given the results obtained, this method might also be used in order to prune neural networks without hurting accuracy.

Only intern parameters are used with that method, whereas ours also uses data samples to compute neural activation. Our algorithm is presented hereunder:

Algorithm III. General algorithm for back-propagation feature selection methods.

1. Train a network (or use a pre-trained network).

2. For each training sample, do the following steps:

   (a) Initialization phase: Assign importance to the neurons of the network's last layer, by propagating the training sample through the network.
   (b) Back-propagation (step one): Use the importance of neurons from layer i to compute those of layer i 1, where layer i is the one for which importance have already been assigned and is the furthest away from the output.
   (c) Back-propagation (step two): Repeat step (b) until importance have been assigned to the input layer's neurons.
   (d) Store importance: The features importance for this input sample correspond to the neurons importance of the input layer and need to be stored.

3. Repeat step (2) for each training sample and sum all the feature importance.

4. The sum finally obtained corresponds to the overall feature importance's.

In a single output regression problem setting, we also need to consider negative output values the same way as positive ones. This leads to the following initialization (with w_i the weights connecting the last hidden layer to the output) :

$$Imp(n_i) = |Out(n_i) * w_i|$$

In multi-output regression settings, we make the hypothesis that each output neuron has the same importance. This way, if we let n_1; :::; n_k be the neurons of the last hidden layer and m1; :::;ml be the output neurons (l = 1 if it is a single output problem). We have (with wix the weights connecting the last hidden layer to output x):

$$Imp(n_i) = \sum_{x=1}^{l} |Out(n_i) * w_{ix}|$$

In classification problems, softmax layers are often added before the output layer such that the output neurons correspond to probabilities of belonging to a given class. Softmax layers map a N-dimensional vector v of arbitrary real values to another N-dimensional vector soft(v) with values in the range [0; :::; 1] that add up to 1. This is done by using the following formula:

$$soft(v)_j = \frac{e^{v_j}}{\sum_{n=1}^{N} e^{v_n}}, \text{ for j} = 1,...$$

This method might not be optimal since we consider each neuron of the softmax layer as equally important, and we do not consider negative and positive values differently. For example, for a binary classification problem, we make no difference if the inputs of the softmax layer are [0:7; 0:2] or [0:7;0:2].

4. Results

We are now going to look at the results for a regression problem. The dataset we will use has 5000 input features [x1; :::; x5000]. The regression problem has been generated using the following formula, where the weights have been chosen uniformly at random between 0 and 100:

$$y = \sum_{i=1}^{25} w_i * x_i$$

This means that only 25 out of the 5000 input features are actually useful to predict the output. Figure 1 shows the importance of each neuron (layer by layer). In this case the neural network used has 4 hidden layers of 500 neurons each. Layer 0 is the input layer and thus corresponds to the feature importance. As we can see, there are peak importance on the first neurons of layer 0, this is expected (due to the problem nature, which only uses the 25 first features) and proves that the technique used seems reasonable.

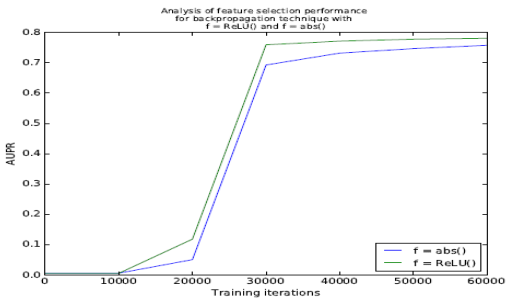

Figure 1: Comparison between of feature selection performance

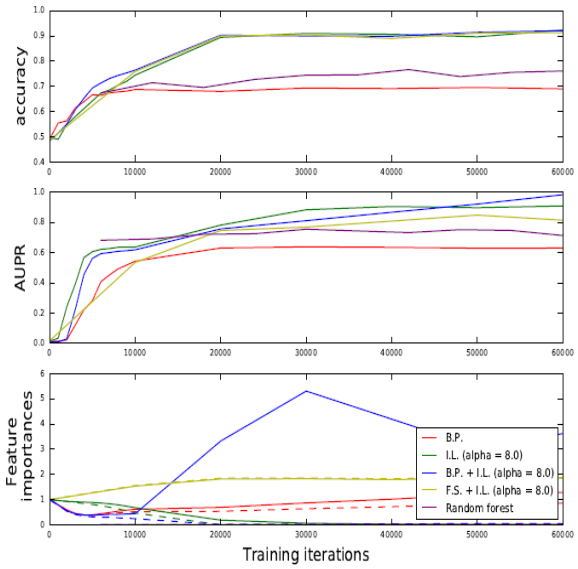

Figure 2: Accuracy, AUPR and weights evolution of a network for different algorithms.

5. Conclusion

Multiple algorithms have been presented with their advantages and drawbacks, the choice of which to use is thus situation dependant. If the goal is to select features on a un-noised dataset or to use a pre trained network, then the B.P. algorithm should be used. Otherwise using the B.P + I.L. technique is the way to go most of the time. It must not be forgotten that as shown in Subsection 4.2.2 neural architecture plays a big role. Indeed, results can vary greatly with the number of hidden layers/neurons per layer. However we showed that since the higher the accuracy the better the feature selection, this problem can be addressed by using cross-validation to find a near optimal architecture. Finally, we showed that the computation time of the B.P and B.P + I.L. algorithms is of the order of an epoch, which is really not a huge deal compared to the training time. Therefore, only the swapping techniques suffer from their computation time. It is also very important to remember the extensions given for each algorithm. First, remember that the regularization can be modified as explained in Section 3.1 according to the problem. For example, it is sometimes considered useful to detect redundant features but can also be detrimental. Also as stated, multiple algorithm initialization methods could be imagined and we have given clues on what could be changed to the current algorithm in order to further enhance the results. As an example we have given another way to initialize the B.P. algorithm in the case of binary classification with Equation 3.7, which would likely result in enhanced performances.

References

[1]. Li, Y., yu Chen, C., and Wasserman, W. W. (2015). Deep feature selection: Theory and application to identify enhancers and promoters. JOURNAL OF COMPUTATIONAL BIOLOGY, pages 1{15.

[2]. Marbach, D., Scha_ter, T., Mattiussi, C., and Floreano, D. (2009). Generating realistic in silico gene networks for performance assessment of reverse engineering methods. Journal of Computational Biology, 16(2):229{239.

[3]. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and M uller, K.-R. (2017). Explaining nonlinear classi_cation decisions with deep taylor decomposition. Pattern Recognition, 65:211{222.

[4]. Nair, V. and Hinton, G. E. (2010). Recti_ed linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10), pages 807{814.

[5]. Nilsson, R., Pe~na, J. M., Bj orkegren, J., and Tegn_er, J. (2007). Consistent feature selection for pattern recognition in polynomial time. Journal of Machine Learning Research, 8(Mar):589{612.

[6]. Qi, Y. (2012). Random forest for bioinformatics. In Ensemble machine learning, pages 307{323. Springer.

[7]. Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. Cognitive modeling, 5(3):1.

[8]. Saeys, Y., Inza, I., and Larraaga, P. (2007). A review of feature selection techniques in bioinformatics. Bioinformatics, 23(19):2507.

[9]. Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation di_erences. arXiv preprint arXiv:1704.02685.

[10]. Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from over_tting. Journal of Machine Learning Research, 15(1):1929{1958.

[11]. Stolovitzk, G., Monroe, D., and Califano, A. (2007). Dialogue on reverse- engineering assessment and methods: The dream of high-throughput pathway inference. Annals of the New York Academy of Sciences, 1115:11{22.

[12]. Stolovitzky, G., Prill, R., and Califano, A. (2009). Lessons from the dream2 challenges. Annals of the New York Academy of Sciences, 1158:159{95.

[13]. Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. BMC bioinformatics, 9(1):307.

[14]. Wang, M., Chen, X., and Zhang, H. (2010). Maximal conditional chi-square importance in random forests. Bioinformatics, 26(6):831{837.

[15]. Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2):301{320.

[16]. Altmann, A., Tolo_si, L., Sander, O., and Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. Bioinformatics, 26(10):1340{1347.

[17]. Chen, X., Liu, C.-T., Zhang, M., and Zhang, H. (2007). A forest-based approach to identifying gene and gene{gene interactions. Proceedings of the National Academy of Sciences, 104(49):19199{19203.

[18]. Debaditya, R., Sri, R. M. K., and Krishna, M. C. (2015). Feature selection using deep neural networks.

[19]. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and M uller, K.-R. (2017). Explaining nonlinear classi_cation decisions with deep taylor decomposition. Pattern Recognition, 65:211{222.

[20]. Nair, V. and Hinton, G. E. (2010). Recti_ed linear units improve restricted boltzmann machines. In Proceedings of the 27th international conference on machine learning (ICML-10), pages 807{814.

[21]. Nilsson, R., Pe~na, J. M., Bj orkegren, J., and Tegn_er, J. (2007). Consistent feature selection for pattern recognition in polynomial time. Journal of Machine Learning Research, 8(Mar):589{612.

[22]. Qi, Y. (2012). Random forest for bioinformatics. In Ensemble machine learning, pages 307{323. Springer.

[23]. Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. Cognitive modeling, 5(3):1.

[24]. Saeys, Y., Inza, I., and Larraaga, P. (2007). A review of feature selection techniques in bioinformatics. Bioinformatics, 23(19):2507.

[25]. Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation di_erences. arXiv preprint arXiv:1704.02685.

[26]. Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from over_tting. Journal of Machine Learning Research, 15(1):1929{1958.

[27]. Stolovitzk, G., Monroe, D., and Califano, A. (2007). Dialogue on reverse- engineering assessment and methods: The dream of high-throughput pathway inference. Annals of the New York Academy of Sciences, 1115:11{22.

[28]. Stolovitzky, G., Prill, R., and Califano, A. (2009). Lessons from the dream2 challenges. Annals of the New York Academy of Sciences, 1158:159{95.

[29]. Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008). Conditional variable importance for random forests. BMC bioinformatics, 9(1):307.

[30]. Wang, M., Chen, X., and Zhang, H. (2010). Maximal conditional chi-square importance in random forests. Bioinformatics, 26(6):831{837.