Feature Salience for Neural Networks: Comparing Algorithms

Theodor Heinze, Martin von Löwis, and Andreas Polze

Hasso-Plattner-Institute for Software Systems Engineering

{Theodor.Heinze, Martin.vonLoewis, Andreas.Polze}@hpi.uni-potsdam.de

Abstract. One of the key problems in the field of telemedicine is the prediction of the patient's health state change based on incoming non-invasively measured vital data. Artificial Neural Networks (ANN) are a powerful statistical modeling tool suitable for this problem. Feature salience algorithms for ANN provide information about feature importance and help selecting relevant input variables. Looking for a reliable salience analysis algorithm, we found a relatively wide range of possible approaches. However, we have also found numerous methodological weaknesses in corresponding evaluations. Perturb [11][7] and Connection Weight (CW) [1] are two of the most promising algorithms. In this paper, we propose an improvement for Connection Weight and evaluate it along with Perturb and the original CW. We use three independent datasets with already known feature salience rankings as well as varying topologies and random feature ranking results to estimate the usability of the tested approaches for feature salience assessment in complex multi-layer perceptrons.

Keywords: Feature Salience, Sensitivity Analysis, Neural Networks, Machine Learning, Telemedicine.

1 Introduction

Working in the field of telemedicine, one of the key problems is the estimation of patient's health based on the history of incoming non-invasively measured vital data [20]. Artificial Neural Networks (ANN) are a powerful statistical model suitable for modeling this type of non-linear problems. One drawback is the "black box" outward appearance of ANN since contributions of each input variable in the prediction process are not trivially determinable. To address this issue, a wide range of so called feature salience/sensitivity analysis algorithms has been introduced already. A recent study [5], comparable to our current work, has employed many of them to estimate the importance of input features in the analysis of vital measurement data. Also, [5] hints that current feature salience algorithms are not reliable enough to be used standalone and considers ensemble results.

Since the 90ies, sensitivity analysis and its variants have been the dominating paradigm [6][4][10][12]. Garson's algorithm [8], a widely used approach, was found to

The project on which this report is based was funded by the Ministry of Economy of Brandenburg and the EU under grant No. 80136574. This publication reflects the views only of the author.

be quite weak by several comparisons [1][11][5]. It has been the field of ecology in particular, in which many different methods have been developed and tried [3][9][18][1]. Unfortunately, some of ecology-based evaluations employ datasets with effectively unknown true feature salience. Expert opinion is used to rank the input variables and then proposed algorithms are tested for compliance [11]. Also, in many cases only one topology of ANN is used, usually a very small one [1][13]. Another frequent evaluation weakness is the employment of only one dataset.

The main contribution of this paper is the introduction of a ranking-based testing methodology for feature salience algorithms and applying it to the two most promising techniques: Perturb [11][7] and Connection Weight [18][1]. Also, we introduce an improvement for the latter. Three independent datasets with already known feature salience ranking are used, also several topologies and varying training quality to analyze algorithm behavior under different conditions and compare its quality with random ranking. We also show how cumulative analysis of a set can distinctively improve ranking performance.

2 Feature Salience Algorithms

Feature salience algorithms are used to analyze a trained Neural Network and determine the importance of each input feature. Usually, *real* [+∞…-∞] values are assigned to all input features so than a ranking can be drawn. If several trained neural networks are available, it is possible to sum the assigned values corresponding to the input features over several networks and compute a cumulative ranking then. Assigned values have to be normalized each step, so that the contribution of the analysis of one single network does not dominate the whole set.

2.1 Perturb

Perturb seems to be the method of choice for many researchers. In fact, its concept is very logical and we have not seen one single review where its results have been questioned. Perturb measures the change in the root mean squared error (RMSE) of the network, while noise is added to the input variables successively. The input features are then ranked by the RMSE change they induce. A recent analysis [7] shows that the optimum input perturbation ratio range is around [-20%, 20%].

```
given: a dataset and a trained neural network
initialize Array RMSEchangeSum[input features]
FOR all samples in dataset
  Measure output RMSE error
  FOR all input features
    perturb feature and measure change in RMSE
    add RMSE change to RMSEchangeSum[input feature]
  NEXT input feature
NEXT sample
Rank input feature salience by values in RMSEchangeSum
```

[Pseudo-code for the Perturb Algorithm]

2.2 Connection Weight

Connection Weight (CW) is another powerful and also logical approach for feature salience. It "calculates the product of the raw input-hidden and hidden-output connection weights between each input neuron and sums the products across all hidden neurons" [1]. A comprehensive introduction and review is offered by [1]. This method is similar to the formerly very popular Garson's algorithm [8], though is reported to perform much better [11][5].

2.3 Connection Impact

Our algorithm exploits ideas used in already known methods, mainly CW. Though, we do not use the raw connection weights, rather for every data sample and every weight, an absolute impact is computed, which describes the percentage of contribution of the connection to the target neuron. The impact is computed by multiplying the raw connection weight with the activation of the source neuron and then normalizing. The impact values are used to backtrack the square error from the output to the input neurons, where it is then summed over the data samples.

```
given: a dataset and a trained neural network
initialize Array Salience[input neurons]
FOR all samples in dataset
  process sample with trained network, compute output
  map input range to -1 ... 1  (= activation input layer)
  FOR every weight:
      Impact[weight] = weight * activation(source neuron)
  NEXT weight
  Normalize Impacts, so that for every target neuron,
      incoming impacts (absolute values) sum up to 1
  FOR all neuron layers starting with output layer
    FOR all neurons
      IF output layer THEN
        neuron_value = (correctOutput - computedOutput)²
        CONTINUE with next neuron
      ELSE neuron_value = ∑(outgoingImp *
                                  neuron_value(target))
      IF input layer THEN
              Salience[neuron] += absolute(neuron_value)
    NEXT neuron
  NEXT layer
NEXT data sample
Rank input feature salience by values in Salience
```

[Pseudo-code for the Connection Impact algorithm]

3 Testing Methodology

For every dataset we will train a set of neural networks, which are then analyzed by the assessed algorithms. In single network analysis, an algorithm computes a ranking from one dataset/network combination, while in cumulative analysis a dataset and a (sub)set of its trained networks are employed to calculate one cumulative ranking. Since all tested algorithms work with accumulations of values corresponding to the impact of input features, we will test if accumulating over several trained networks affects ranking accuracy. The ranking error of all input features is computed by calculating the Euclidean distance between the true and the estimated ranking vector.

If e.g. out of three input features the 2nd feature is the most important and the 1st feature is the least important, then the true ranking is {3, 1, 2}. If an algorithm estimates a wrong ranking {2, 1, 3}, then the ranking error is (1).

$$\sqrt{(3-2)^2 + 0 + (2-3)^2} = \sqrt{2} = 1{,}414 \quad (1)$$

The maximum ranking error for a set of *n* features is (2).

$$maxErr = \sqrt{\frac{n^3-n}{3}} \quad (2)$$

For better comparability between problem sizes, we normalize the actual ranking error with this maximum.

3.1 Artificial Neural Networks and Testing Procedure

In our experiments we have used standard feed-forward multi-layer neural network design with sigmoid activation function and squared error function. Network topology has been varied over the datasets to account for different complexity levels of the data. Training was performed with the standard-backpropagation approach [16] along with RPROP enhancement [15] for weight adjustment. We have used a training and a validation set for training and a test set to determine the true performance of the trained network. Input/output normalizing, early stopping techniques and error measures from [14] have been used to standardize our results.

3.2 Datasets

We have used three different datasets to review the considered algorithms. Our main concern was to have more than one dataset and also try different topologies to test resulting performance changes. Also, it is crucial to know the true feature salience in advance.

The first dataset was *Demosaicing* with 36 input and 12 output variables. Demosaicing is an image processing technique which is used to reconstruct three channel (red, green and blue) 2x2 pixel matrix information from digital sensor data which only captures information of one channel (red, green or blue) for each pixel location. Thus, the goal is the estimation of the missing color values. The input vector corresponds to a 6x6 crop from the digital sensor around the 2x2 matrix (output vector).

We have used an imaging model described in [2] to artificially simulate a digital sensor and thus generate 10.000 data samples from 400 digital images.

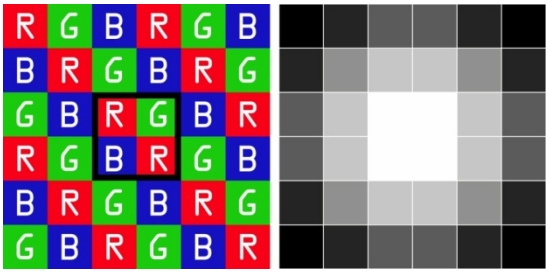

Fig. 1. Demosaicing problem and feature salience distribution (brighter = better)

We had used a very similar approach before [17] to train a neural network which was then able to properly render real-world images from data generated by a real digital imaging sensor. Obviously, due to the distribution of light, input features in the center of the sensor crop are more relevant for the restoration of the true color values than the pixels on border and edges of the crop. We used the Euclidean distance function in true feature ranking to account for that (Figure 1).

Our second dataset, Tic-Tac-Toe (TTT), is taken from the UCI Machine Learning Repository [19]. TTT contains a of 958 endgame positions from the homonymous game (sometimes also called *Noughts and Crosses*) classified as *won* or *not-won* for the first player. Here, the 9 input features correspond to the occupation of the 9 fields: *cross*, *nought* or *empty*. Since from the center field there are four possibilities to build a row and only three and two from edges and border accordingly, the true ranking of feature salience is pretty clear (Figure 2).

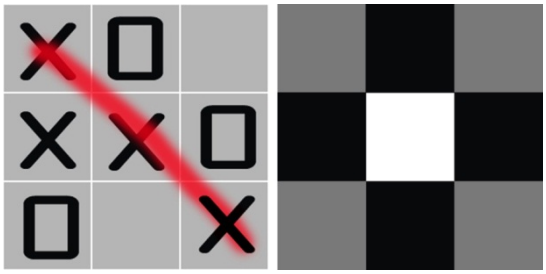

Fig. 2. TTT: typical endgame position (left) and feature salience distribution (brighter = better)

The third dataset, simulated data, is based on a artificially generated function with 20 input variables which correlate with the output variable in a range between 0.1 and 0.9, similarly to [1]. This dataset contains 1000 samples.

4 Results

For all datasets, we have trained a set of 100 networks. Network topology was set to one hidden layer (10 neurons) for TTT and the artificial dataset, while for Demosa-

icing it was 2 hidden layers (25 + 16). RMSE [14] was fairly low, as shows Figure 3. For Tic-Tac-Toe, we picked 100 samples with perfect training (≈zero error).

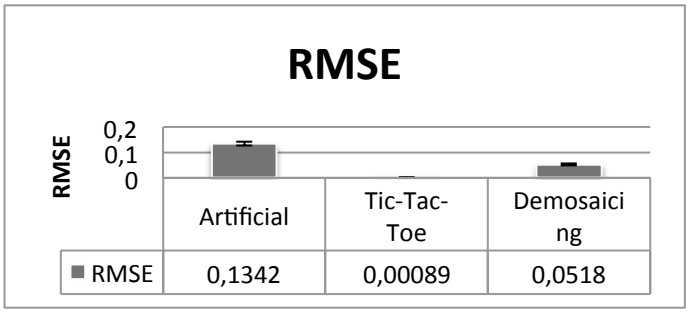

Fig. 3. RMSE and standard deviation, 100 samples.

4.1 Single network analysis

Figure 4 shows the mean ranking errors (% of max. error) for every algorithm/set combination. We have added random ranking error values to give the impression how useful the results are. All algorithms performed reasonably well analyzing networks trained with the artificial dataset. Unlike in [1], Perturb performed better than CW. We suppose the reason is the increased number of input features (20 vs. 10) and the more complex network topology. Perturb also performed clearly best on the rest of the tested sets and was in some cases even able to determine the perfect ranking for the TTT dataset, while CW showed rankings not so much better than random.

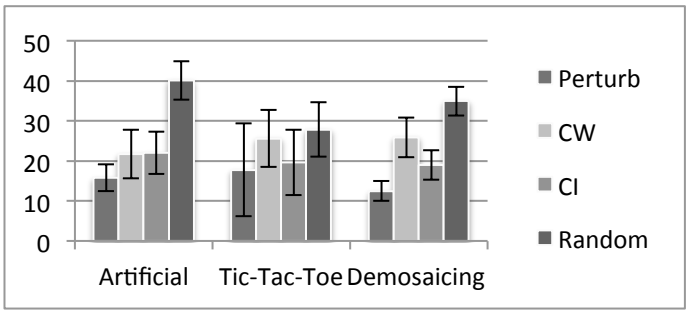

Fig. 4. Single network analysis: mean errors and standard deviation, 100 samples.

4.2 Cumulative Analysis

In many cases there are several trained networks available for a dataset, so why not use that extra information for feature salience analysis? To test algorithm performance in this case, we partitioned the available trained networks into 10 subsets with 10 networks each. Then we calculated the cumulative ranking for every subset/algorithm combination (Figure 5). Error rates improved in most cases. Considering accuracy and

precision, Perturb is still the clear winner. Especially remarkable is its performance on the TTT set, where it was able to perfectly rank the features in 7/10 cases. In the last experiment, the whole set of 100 networks is employed to compute the cumulative ranking (Figure 6). Perturb and CI were able to perfectly rank the features of the TTT dataset. CI slightly outperformed Perturb on the artificial dataset.

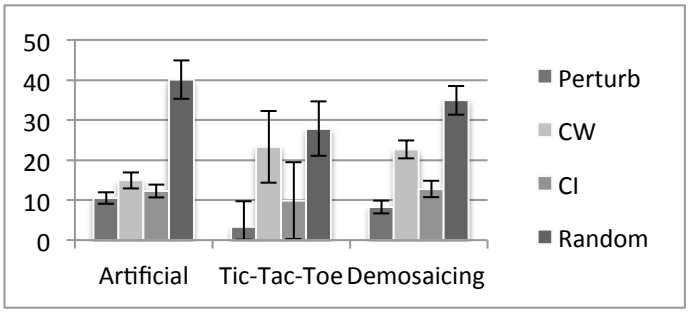

Fig. 5. Cumulative analysis: mean errors and standard deviation, 10 samples.

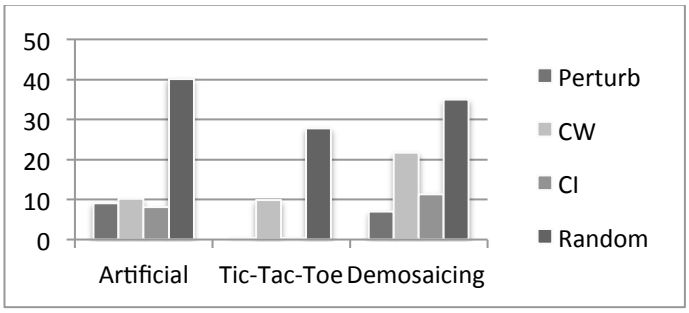

Fig. 6. Cumulative analysis: ranking error

5 Conclusion and Outlook

The original Connection Weight performed worst, e.g. calculating a ranking only slightly better than Random for Demosaicing. CI did better overall, but was also not able to compete against Perturb which consistently presented best results and generated usable (clearly better than Random) rankings. With cumulative analysis, situation improved even further. The choice of a feature salience algorithm for our future work is now clear.

References

[1] J. D. Olden, M. K. Joy, and R. G. Death, “An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data,” *Ecological Modelling*, vol. 178, no. 3–4, pp. 389–397, Nov. 2004.

[2] S. Farsiu, M. Elad, and P. Milanfar, “Multi-frame demosaicing and super-resolution of color images,” *IEEE Trans. on Image Processing*, vol. 15, pp. 141–159, 2006.

[3] S. Lek, M. Delacoste, P. Baran, I. Dimopoulos, J. Lauga, and S. Aulagnier, “Application of neural networks to modelling nonlinear relationships in ecology,” *Ecological Modelling*, vol. 90, no. 1, pp. 39–52, Sep. 1996.

[4] W. Wang, P. Jones, and D. Partridge, “Assessing the Impact of Input Features in a Feedforward Neural Network,” *Neural Computing & Applications*, vol. 9, no. 2, pp. 101–112, 2000.

[5] X. R. Cui, M. F. Abbod, Q. Liu, J. S. Shieh, T. Y. Chao, C. Y. Hsieh, and Y. C. Yang, “Ensembled artificial neural networks to predict the fitness score for body composition analysis.,” *The journal of nutrition health aging*, vol. 15, no. 5, pp. 341–348, 2011.

[6] T. Tchaban, M. J. Taylor, and J. P. Griffin, “Establishing impacts of the inputs in a feedforward neural network,” *Neural Computing & Applications*, vol. 7, no. 4, pp. 309–317, 1998.

[7] R. Bai, H. Jia, and P. Cao, “Factor Sensitivity Analysis with Neural Network Simulation based on Perturbation System,” *Journal of Computers*, vol. 6, no. 7, Jul. 2011.

[8] G. D. Garson, “Interpreting neural-network connection weights,” *AI Expert*, vol. 6, no. 4, pp. 46–51, Apr. 1991.

[9] I. Dimopoulos, J. Chronopoulos, A. Chronopoulou-Sereli, and S. Lek, “Neural network models to study relationships between lead concentration in grasses and permanent urban descriptors in Athens city (Greece),” *Ecological Modelling*, vol. 120, no. 2–3, pp. 157–165, Aug. 1999.

[10] J. J. Montaño and A. Palmer, “Numeric sensitivity analysis applied to feedforward neural networks,” *Neural Computing & Applications*, vol. 12, no. 2, pp. 119–125, 2003.

[11] M. Gevrey, I. Dimopoulos, and S. Lek, “Review and comparison of methods to study the contribution of variables in artificial neural network models,” *Ecological Modelling*, vol. 160, no. 3, pp. 249–264, Feb. 2003.

[12] A. Y. Cheng and D. S. Yeung, “Sensitivity analysis of neocognitron,” *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 29, no. 2, pp. 238–249, May 1999.

[13] Y. Dimopoulos, P. Bourret, and S. Lek, “Use of some sensitivity criteria for choosing networks with good generalization ability,” *Neural Processing Letters*, vol. 2, no. 6, pp. 1–4, 1995.

[14] L. Prechelt, “Proben 1,” presented at the Technical Report 21/94, 1994.

[15] M. Riedmiller and H. Braun, “RPROP - A Fast Adaptive Learning Algorithm,” Proc. of ISCIS VII), Universitat, 1992.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *, Nature, published online: 09 October 1986; | doi:10.1038/323533a0*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[17] T. Heinze, M. von Lowis, and A. Polze, “Joint multi-frame demosaicing and super-resolution with artificial neural networks,” in *Systems, Signals and Image Processing (IWSSIP), 2012 19th International Conference on*, 2012, pp. 540 –543.

[18] J. D. Olden and D. A. Jackson, “Illuminating the ‘black box’: a randomization approach for understanding variable contributions in artificial neural networks,” *Ecological Modelling*, vol. 154, no. 1–2, pp. 135–150, Aug. 2002.

[19] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets.html

[20] T. Heinze, R. Wierschke, A. Schacht, and M. von Löwis, “A Hybrid Artificial Intelligence System for Assistance in Remote Monitoring of Heart Patients,” in *Hybrid Artificial Intelligent Systems*, vol. 6679, E. Corchado, M. Kurzynski, and M. Wozniak, Eds. Springer Berlin / Heidelberg, 2011, pp. 413–420.