Deep Taylor Decomposition of Neural Networks

Grégoire Montavon[1] GREGOIRE.MONTAVON@TU-BERLIN.DE
Sebastian Bach[2] SEBASTIAN.BACH@HHI.FRAUNHOFER.DE
Alexander Binder[3] ALEXANDER_BINDER@SUTD.EDU.SG
Wojciech Samek[2,5] WOJCIECH.SAMEK@HHI.FRAUNHOFER.DE
Klaus-Robert Müller[1,4,5] KLAUS-ROBERT.MUELLER@TU-BERLIN.DE

[1] Department of Electrical Engineering and Computer Science, Technische Universität Berlin, 10587 Berlin, Germany
[2] Department of Video Coding & Analytics, Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany
[3] Information Systems Technology and Design, Singapore University of Technology and Design, 487372, Singapore
[4] Department of Brain and Cognitive Engineering, Korea University, Seoul 136-713, Korea
[5] Berlin Big Data Center (BBDC), Berlin, Germany

Abstract

We summarize the main concepts behind a recently proposed method for explaining neural network predictions called *deep Taylor decomposition*. For conciseness, we only present the case of simple neural networks of ReLU neurons organized in a directed acyclic graph. More structured networks with special layers are discussed in the original paper (Montavon et al., 2015).

1. Introduction

Deep neural networks have become a tool of choice for a number of machine learning problems from automated image classification (Krizhevsky et al., 2012; Szegedy et al., 2015) to natural language processing (Collobert et al., 2011; Kim, 2014). In contrast to their very high predictive performance, deep networks are often perceived as "black-boxes", limiting in practice their even broader applicability, in particular, for sensitive tasks where a consensus needs to be reached between the machine learning prediction and the human expert.

Among the multiple recently proposed techniques for explaining neural network predictions (Simonyan et al., 2013; Zeiler and Fergus, 2014; Bach et al., 2015), we can identify two distinct approaches: *sensitivity analyses* (e.g. Gevrey et al., 2003), that find the input variables that causes the output to vary locally, and *decompositions* (e.g. Poulin et al., 2006) that seek to redistribute the function output on the input variables in a meaningful way.

Presented at the ICML 2016 Workshop on Visualization for Deep Learning

Techniques such as *layer-wise relevance propagation* (Bach et al., 2015) produce a decomposition for deep neural networks, by building for each neuron a redistribution rule, and applying these rules in a backward pass until the input variables (e.g. pixels) are reached. For image recognition tasks, the procedure results in a "heatmap" that indicates what pixels of the image are important for a particular neural network prediction. An example of pixel-wise decomposition is shown in Figure 1. We can observe that the classifier output is mainly redistributed on the pixels representing the actual object to detect, and not on the background.

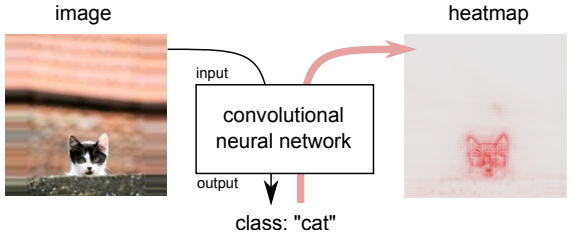

Figure 1. Image processed by a neural network whose prediction is decomposed onto input variables. The resulting decomposition forms a heatmap indicating what pixels in the image are the most relevant for the prediction.

Parameters of the decomposition procedure (e.g. the choice of propagation rules) can be selected such that a measure of quality of the decomposition is maximized. While some techniques such as "pixel-flipping" (Samek et al., 2015) have been proposed as a measure of quality, it is still an open question how to best measure decomposition quality, and how to do so in a fully unbiased manner.

Here, we use a different strategy: A basic model for decomposition based on first-order Taylor expansions is con-

sidered, and is viewed as correct for a simple class of models (e.g. single neurons). Then, the model complexity is gradually increased by constraining the input domain, and adding multiple layers of neurons, while at the same time trying to deviate as little as possible from the original decomposition method. As a result of this iterative process, we obtain the deep Taylor decomposition method.

2. Decomposing a ReLU Neuron

Consider a simple neuron receiving a real-valued input vector $(x_i)_i$ and producing the output

$$x_j = \max(0, \sum_i x_i w_{ij} + b_j)$$

where $b_j \leq 0$. We would like to decompose the neuron output in terms of input variables. It can be remarked that the neuron function is linear on the subset of the input space that produces $x_j > 0$ (we call it "active set"). On this subset, the output can be written as a first-order Taylor expansion

$$x_j = \sum_i \frac{\partial x_j}{\partial x_i}\Big|_{(x_i)_i=(\widetilde{x}_i)_i} \cdot (x_i - \widetilde{x}_i),$$

where $(\widetilde{x}_i)_i$ is a *root point* in the active set, with near-zero output, and at which the Taylor expansion is performed. The decomposition of x_j onto input variables is given directly by identification of the elements of the sum:

$$[x_j]_i = \frac{\partial x_j}{\partial x_i}\Big|_{(x_i)_i=(\widetilde{x}_i)_i} \cdot (x_i - \widetilde{x}_i)$$

Note that the property $\sum_i [x_j]_i = x_j$ is satisfied, and we say in that case, that the redistribution from x_j to $(x_i)_i$ is *conservative*.

One degree of freedom of the analysis is the choice of root point $(\widetilde{x}_i)_i$ at which the Taylor expansion is performed. Montavon et al. (2015) showed that searching the root point following a direction $(v_{ij})_i$ in the input space starting from the actual data point $(x_i)_i$ gives the explicit decomposition formula

$$[x_j]_i = \frac{v_{ij} w_{ij}}{\sum_{i'} v_{i'j} w_{i'j}} x_j \tag{1}$$

that involves both the neuron orientation and the search direction. A graphical depiction of the root search process for two different search directions, and two different neurons is given in Figure 2. We study two specifications of the search direction motivated by constraints on the input domain, each of them leading to a different redistribution rule.

2.1. Unconstrained input and the w^2-rule

If the input space is unconstrained ($(x_i)_i \in \mathbb{R}^d$), the nearest root point is obtained by searching along the gradient direction, i.e. we set $v_{ij} = w_{ij}$, and obtain the redistribution rule

$$[x_j]_i = \frac{w_{ij}^2}{\sum_{i'} w_{i'j}^2} x_j.$$

This rule is called "w^2-rule" and can be viewed up to a constant normalization factor as input sensitivities ($(\partial x_j / \partial x_i)^2 = w_{ij}^2 \cdot 1_{x_j>0}$) multiplied by the saliency of the input pattern (as measured by the neuron activation x_j), that is,

$$\text{decomposition} = \text{sensitivity} \times \text{saliency}.$$

2.2. Positive input and the z^+-rule

We consider the case $(x_i)_i \in \mathbb{R}_+^d$ occurring when the neuron receives as input the output of other ReLU neurons. Searching for a root point along the gradient direction does not guarantee that it will obey the positivity constraints. Searching for the nearest root point subject to the positivity constraints is possible, however it would require solving an optimization problem.

Instead, we further restrict the search domain by fixing a search direction $(v_{ij})_i$ which is not necessarily optimal, but for which a root point is guaranteed to be found in the input domain. One such direction is given by $v_{ij} = x_i \cdot 1_{w_{ij}>0}$. That is, we move towards the origin (where $x_j = 0$) in the subspace of neurons with positive associated weights such as only negative weighted activations persist, and a root point is eventually reached. Injecting this new search direction into Equation 1 gives the redistribution rule

$$[x_j]_i = \frac{x_i w_{ij}^+}{\sum_{i'} x_{i'} w_{i'j}^+} x_j,$$

where the decomposition on input variables is now determined based on both the input vector and the weights. This rule is called "z^+-rule" and can be applied to any ReLU neuron with positive inputs and negative bias.

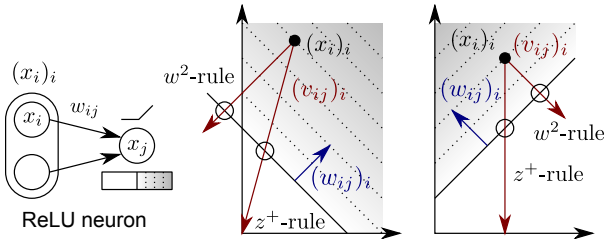

Figure 2. Illustration of a root point search in the two-dimensional input space of a ReLU neuron, for different choices of weights $(w_{ij})_i$ and bias b_j. The data point $(x_i)_i$ is represented as a black dot, and the possible root points $(\widetilde{x}_i)_i$ are depicted as circles.

3. Decomposing a Neural Network

Consider a neural network mapping an input vector $(x_p)_p$ to an output scalar x_f, through an interconnection of many ReLU neurons arranged in a directed acyclic graph. The output neuron x_f is first decomposed on its input neurons. Then, the decomposition on these neurons is redistributed on their own inputs, and the redistribution process is repeated until the input variables are reached. For this purpose, we define the messages $[[x_f]_j]_i$ designating how much of x_f is redistributed from an arbitrary neuron x_j to one of its input x_i. Redistributed terms coming from the neurons $(x_j)_j$ to which x_i contributes are summed:

$$[x_f]_i = \sum_j [[x_f]_j]_i$$

A relevant portion of a simple neural network exhibiting both redistribution and summing in the propagation phase is shown in Figure 3.

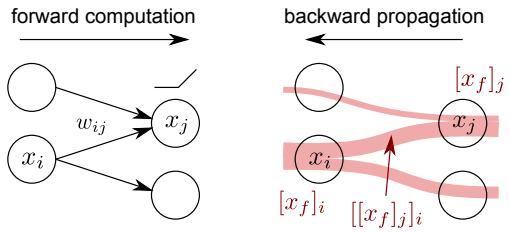

Figure 3. Portion of a neural network annotated with relevant variables, showing a neuron computation x_j, and a neuron x_i on which the neural network output is redistributed.

We introduce an induction mechanism that allows us to redistribute for any neuron x_i its share $[x_f]_i$ on its predecessors. Assume that for all neurons $(x_j)_j$ to which x_i contributes, we can write $[x_f]_j = x_j c_j$, i.e. a product of the neuron activation and a constant term c_j. The following sequence of equations show that the same holds for $[x_f]_i$:

$$[x_f]_i = \sum_j [[x_f]_j]_i = \sum_j [x_j c_j]_i = \sum_j [x_j]_i c_j = x_i c_i$$

where

$$c_i = \sum_j \frac{w_{ij}^+ [x_f]_j}{\sum_{i'} x_{i'} w_{i'j}^+}.$$

The variable c_i is indeed approximately constant under a perturbation of x_i due to the latter having its effect in c_i diluted by summing over many other neurons. As a boundary condition, the output can also be written as $[x_f]_f = x_f c_f$ with $c_f = 1$.

Having shown by induction that the decomposition onto any neuron in the graph has a product structure, and having made explicit the propagation rule between these neurons, the decomposition $([x_f]_p)_p$ onto the input variables $(x_p)_p$ can be easily computed by application of these propagation rules in a backward pass. Note that because the redistribution rule for each neuron is always conservative, the decomposition for the whole network is also conservative, that is, $\sum_p [x_f]_p = x_f$.

An application of deep Taylor decomposition to the GoogleNet neural network (Szegedy et al., 2015) results for a selected image in the heatmap of Figure 1. Details of how specific layers are handled in such network, e.g. pooling layers, normalization layers, input layer, and linear layers with positive biases, are described in the original paper on deep Taylor decomposition.

References

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10 (7):e0130140, 07 2015.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

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*, 160(3):249–264, 2003.

Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.

G. Montavon, S. Bach, A. Binder, W. Samek, and K.-R. Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *CoRR*, abs/1512.02479, 2015.

B. Poulin, R. Eisner, D. Szafron, P. Lu, R. Greiner, D. S. Wishart, A. Fyshe, B. Pearcy, C. Macdonell, and J. Anvik. Visual explanation of evidence with additive classifiers. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence*, pages 1822–1829, 2006.

W. Samek, A. Binder, G. Montavon, S. Bach, and K.-R. Müller. Evaluating the visualization of what a deep neural network has learned. *CoRR*, abs/1509.06321, 2015.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision - ECCV 2014 - 13th European Conference*, pages 818–833, 2014.