Reasoning and Provenance on Neural Networks

Antoine Groudiev
L3, DI, ENS

Silviu Maniu
Slide Team, LIG, UGA

24th June 2024

Abstract

Recently, neural networks allowed computers to solve numerous problems from diverse machine learning fields, such as natural language processsing and computer vision. Compared to traditional algorithms, machine learning models have proven both more successful and more difficult to interpret. Neural networks are considered as black boxes unable to easily explain themselves, that is justifying the reasons that led them to make a prediction. Layer-wise Relevance Propagation (LRP) is a technique that has been introduced to provide explanability by identifying the input features relevant to the output choice. In parallel, research in the databases field developed annotations techniques to compute provenance for queries. In this paper, we extend LRP propagation rules to semiring-based provenance annotations of the network, for LRP to gain in expressivity.

1 Introduction

1.1 Problem statement

Deep neural networks have proven successful for solving with high accuracy machine learning problems. The expressivity of the class of functions generated by neural networks, combined with the relative simplicity of their training, make such models versatile tools to learn the relationship between the inputs and outputs of a dataset.

However, this versatility comes at the cost of poor interpretability: a neural network simply represents a function from one high-dimensional space to another, but provides no justification nor explanation for a given execution. If metrics such as the accuracy over a testing set provide confidence in the fact that the model is able to correctly classify inputs similar to the training set, no guarantee is given that the model generalizes well. Real-world examples show that networks can overfit the input data, or even take shortcuts instead of learning the intended solution [2]. For the user to have confidence in its predictions, a neural network should therefore be able to highlight the patterns in the input data that it actually learned.

1.2 Layer-wise Relevance Propagation

Layer-wise Relevance Propagation (LRP) [1] has been introduced as a technique to explain an execution of a neural network. LRP is a procedure propagating the output of the function backward in the network, using diverse rules to compute the *relevance* of a neuron depending on the relevances of the neurons of the upstream layer. LRP introduces the notion of *relevance score* for a neuron, intuitively quantifying the contribution of this neuron to the classification of final classification. A high relevance score indicates that the neuron led to the activation of the considered output; a negative relevance score represent neurons that increased the activation of another output neuron instead of the one considered.

1.2.1 Setup and notations

In the following, we consider a deep neural network used for a classification task. We assume that it uses the rectifier activation function[1], which is the case in most applications. To ease the notation, we will not consider biases but instead assume that the first neuron of each layer represents the bias.

Let L be the number of layers of the network. We denote by $\left(a_k^{(l)}\right)_k$ the activations of the network. Notably, $\left(a_k^{(1)}\right)_k$ is the input data, and $\left(a_k^{(L)}\right)_k$ is the ouput prediction. We denote by $\left(w_{j,k}^{(l)}\right)_{j,k}$ the weights connecting the l-th layer to the $(l+1)$-th layer. To simulate the weights, we set:

$$\forall l \in [\![1, L]\!], \quad a_0^{(l)} = 1$$

and we define $w_{0,k}^{(l)}$ to be the bias of the k-th neuron of the $(l+1)$-th layer. The forward propagation rule of a deep rectifier network is therefore:

$$\forall l \in [\![1, L-1]\!], \forall k, \quad a_k^{(l+1)} = \mathrm{ReLU}\left(\sum_{j=0} a_j^{(l)} w_{j,k}\right) = \max\left(0, \sum_{j=0} a_j^{(l)} w_{j,k}\right) \tag{1.2.1}$$

We denote by $R_j^{(l)}$ the relevance of the j-th neuron of the l-th layer. We assume that the output layer represents a one-hot encoding, that is that the belonging of the input to the i-th class is represented by an output vector is of the form $(0, \ldots, a_i^{(L)}, \ldots, 0)$, that where the only non-null coefficient is in the i-th position. Finally, we denote by y the label of a classified input. To the label $y = i$ is associated the output vector $(0, \ldots, a_i^{(L)}, \ldots, 0)$.

1.2.2 Propagation rules

Relevance scores are initialized for the output layer, and are set to the output activation for the correct class, that is:

$$R_i^{(L)} = \begin{cases} a_i^{(L)} & \text{if } i = y \\ 0 & \text{otherwise} \end{cases} \tag{1.2.2}$$

The simplest LRP rule is called LRP-0. It propagates the relevance to a neuron of the lower layer proportionnaly to its contribution to each of the neuron of the next layer:

$$R_j^{(l)} = \sum_k \frac{a_j^{(l)} w_{j,k}}{\sum_{j'} a_{j'}^{(l)} w_{j',k}} R_k^{(l+1)} \tag{1.2.3}$$

The denominator $\sum_{j'} a_{j'}^{(l)} w_{j',k}$ guarantees a conservation property, that is that for any layer l:

$$\sum_j R_j^{(l)} = \sum_k R_k^{(l+1)}$$

This allows to keep the information regarding the total activation of the final layer.

The application of this simple rule can lead into noisy results that do not scale well. An overview of variations of LRP-0 is provided by [4]; not all rules are suitable for all layers. Complex deep neural networks architectures benefit from enhanced rules such as LRP-ε or LRP-γ, which provide more stable explanations.

[1]That is $\mathrm{ReLU}(x) = \max(0, x)$, where ReLU stands for *Rectified Linear Unit.*

Notably, the input layer must be handled using a different rule, since it does not receive its input from ReLU activations, but directly from the input data. The $z^{\mathcal{B}}$ rule is used in [4] to propagate from layer 2 to 1 (input layer):

$$R_j^{(1)} = \sum_k \frac{x_k w_{j,k} - l_j w_{j,k}^{+} - h_j w_{j,k}^{-}}{\sum_{j'} x_k w_{j',k} - l_j w_{j',k}^{+} - h_j w_{j',k}^{-}} R_k^{(2)} \tag{1.2.4}$$

where $(\cdot)^{+} = \max(0, \cdot)$, $(\cdot)^{-} = \min(0, \cdot)$. The parameters l_i and h_i respectively define the theoretical minimum and maximum values of the inputs x_i; for instance we might have $l_i = 0$ and $h_i = 255$ for pixels over 8 bits.

1.3 Semiring-based provenance annotations

In parallel, the notion of data provenance in databases theory developed formal solutions to a similar problem to ours. Data provenance aims at *explaning* a query by highlighting the tuples in the original database that led to the presence of a certain tuple in the query result. If contexts are different, deep neural networks explanation and data provenance share the same general setup: identifying a subset of the input that directly implied a certain output.

A framework to approach data provenance is *provenance semirings*, introduced in [3], annotates tuples using abstract elements of a semiring and apply semiring operations to the tuples appearing in the query. As the query is executed, information about the provenance of the intermediate results is aggregated, resulting in an abstract formula that can be concretized by substituting abstract elements and operations by a concrete semiring. Similarly, in the context of graph databases, edges can be annotated to derive a variety of properties of the query result. [5]

In the following, we provide a mathematical definition of a semiring as well as semiring examples suited for the application to deep neural networks.

Definition (Semiring)**.** A *semiring* is an algebraic structure generalizing the notion of rings. A semiring $(\mathbb{K}, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ is composed of a set $\mathbb{K}$, binary operators $\oplus$ and $\otimes$ such that $\otimes$ distributes over $\oplus$, verifying the following properties:

- $(\mathbb{K}, \oplus, \mathbf{0})$ is a commutative monoid
- $(\mathbb{K}, \otimes, \mathbf{1})$ is a monoid such that $\mathbf{0}$ is absorbing

Example. $(\mathbb{R}, +, \times, 0, 1)$ is a semiring. While it has no direct interpretation is the context of databases, we will see that is corresponds to the basic real-valued LRP.

Example (Boolean semiring)**.** $(\{\bot, \top\}, \vee, \wedge, \bot, \top)$ is a semiring. Its use in databases provenance interprets as the existence of a path between two vertices, using edge weights as the number of different paths between two adjacent vertices.

Example (Counting semiring)**.** $(\mathbb{N}, +, \times, 0, 1)$ is a semiring. For a non-cyclic graph database, its use allows to compute the total number of paths between two vertices, using edge weights as the number of different paths between two adjacent vertices.

References

[1] Sebastian Bach et al. 'On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation'. In: *PLOS ONE* (2015), pp. 1–46. DOI: `10.1371/journal.pone.0130140`. URL: `https://doi.org/10.1371/journal.pone.0130140`.

[2] Robert Geirhos et al. 'Shortcut learning in deep neural networks'. In: *Nature Machine Intelligence* 2 (2020), pp. 665–673.

[3] Todd J Green, Grigoris Karvounarakis and Val Tannen. 'Provenance semirings'. In: *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 2007, pp. 31–40.

[4] Grégoire Montavon et al. 'Layer-Wise Relevance Propagation: An Overview'. In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer International Publishing, 2019, pp. 193–209. URL: `https://doi.org/10.1007/978-3-030-28954-6_10`.

[5] Yann Ramusat, Silviu Maniu and Pierre Senellart. 'Provenance-Based Algorithms for Rich Queries over Graph Databases'. In: *EDBT 2021 - 24th International Conference on Extending Database Technology*. 2021. URL: `https://inria.hal.science/hal-03140067`.