Explaining Recurrent Neural Network Predictions in Sentiment Analysis

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek

Explaining Recurrent Neural Network Predictions

Input (e.g. image or text) $x_d$

NN forward

Predict

NN backward

Explain

(prediction score for a class $c$)

$f_c(x) = \max(0,x_1) + \max(0,x_2)$

Intuition - Toy Example

Two ReLU neurons:

LRR: Two neurons contribute to the prediction

SA is unstable: class points might have different relevance

SA is stable: relevance continuous

Quantitative validation of results

For these experiments:

- use as target class the true sentence class
- consider only sentences with length >= 10 words

Conclusion: most pertinent impact obtained by LRP

LRP is most appropriate to identify words speaking for or against the classifier's decision

Relevance distr. over sentence length

For these experiments:

- use as target class one of the classes
- consider all sentences with length >= 10 words
- use total or only real from left to right encoder

Relevance Statistics:

- divide sentence length into 16 intervals and sum up absolute words relevance per interval, then normalize to one

Use Case: Recurrent NN Model & Task

Contribution:

- Extend LRP to recurrent nets and compare to SA.

Model:

- word-based bidirectional LSTM
- word embeddings of dimension 60; one hidden layer of size 60
- take as input a sequence of word embeddings ($x_1, x_2, \ldots$)

Task:

- five-class sentiment prediction

Decomposing Sentiment onto Words

Pattern Recognition 2017

ArXiv 2017

Montavon et al. 2017

Public Release: 27.01.2018

www.heatmapping.org

Explaining with LRP

How does LRP work?

Layer-wise Relevance Propagation (LRP)

Bach et al. 2015, Arras et al. 2017

LRP: backward relevance redistribution

LRP initialisation: single output neuron $R_k = f_k(x)$

- Element-wise Activation

$z_j = g(z_j)$

Idea: Redistribute the "identity" $R_k$ to $R_j$

The activated neuron value $z_j$ is used to compute $R_j$

- Weighted Linear Connection

$z_j = \sum z_i \cdot w_{ij} + b_j$

Idea: Redistribute relevance proportionally to contribution in forward pass

Step 1: compute relevance messages

$R_{aj} = z_j \cdot w_{ij} + b_j$

Idea: Recursively apply LRP to lower layers

Step 2: sum up incoming messages

$R_j = \sum R_{aj}$

- Multiplicative Interaction

$z_j = z_j \cdot x_j$

Idea: Redistribute relevance to the source

$R_k = R_k \cdot R_j$

The neuron values $z_j$ and $x_j$ are used to compute $R_j$

Advantages over SA?

Sensitivity Analysis (SA)


SA: squared partial derivative

$LRR$ backward relevance redistribution

$R_{aj} = R_k \cdot \frac{\partial f_k}{\partial x_j}$

obtained by standard gradient backpropagation

$R_j = \left(\frac{\partial f_k}{\partial x_j}\right)^2$

SA vs. LRP:

- LRP relevance is signed, while SA relevance are positive (i.e., SA does not distinguish between positive and negative evidence).

- LRP resolves the classification decision on the current input, while SA reveals sensitivity of classifier to small changes in the input values (i.e., SA does not explain the prediction $f_c(x)$).

Previous Work:


Further information and code: www.heatmapping.org