Layer-wise Relevance Propagation in Neural Networks to have more interpretable Machine Learning models

Ariyan Zarei

University of Arizona

ariyanzarei@email.arizona.edu

February 25, 2020

Overview

Motivation

Having More interpretable Neural Networks

- Interpretable Machine Learning (ML) Theme in our Colloquium
- Medical Applications of ML, specially Medical Image Analysis
- Deep Learning (DL) for analyzing histopathological Slides

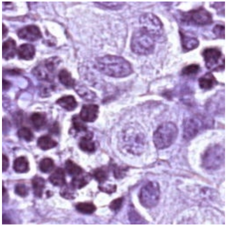


Figure: A sampled window inside the cancerous region of a Slide

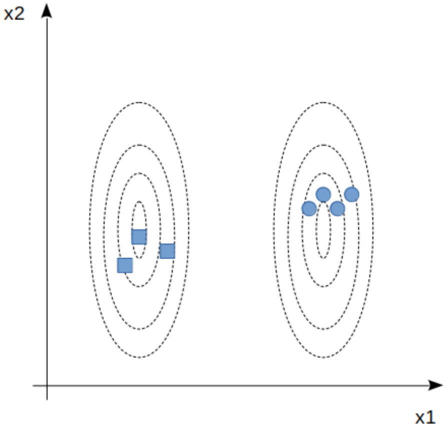
Deep Learning Shortcomings

- Paying Attention to irrelevant and spurious features

- Feature Selection not useful.

Deep Learning Shortcomings

- Paying Attention to irrelevant and spurious features
  Simple example:



Deep Learning Shortcomings

- Deep Neural Networks' Challenges Medical Sciences
  - Fix this problem
  - Explain the predictions of the Models

Papers and Demo

- Layer-Wise Relevance Propagation: An Overview (Explainable AI: Interpreting, Explaining and Visualizing Deep Learning Chapter 10)
- Explaining nonlinear classification decisions with deep Taylor decomposition (Elsevier Pattern Recognition)

Demo: Link

Introduction

Why the neural network is making a particular decision.

- Assess and Validate the prediction and the reason behind it with another inexpensive method.
- Given the final output of a class (softmax), where in the input the network is attending.
- Which parts of the input affect the prediction (positively and negatively).

Terminology and Notations

Note: we focus on images and CNNs in this talk but LRP can be applied to all other forms of data and networks and models.

- Input Image: $x \in \mathbb{R}^d = \{x_p\}$, $p \in \{1, 2, ..., d\}$
- Prediction: $f(x) : \mathbb{R}^d \rightarrow \mathbb{R}^+$ quantifies the presence of an object in the input.
  - Zero: absence of the object
  - Other values: degree of certainty
- Relevance: $R(x) : \mathbb{R}^d \rightarrow {\mathbb{R}^+}^d$ Heatmap with the same size as the input

Relevance Properties

1. Conservation: $\forall x : f(x) = \sum_p R(x)_p$
2. Being Positive: $\forall x, p : R(x)_p \geq 0$
3. Consistent: if properties 1 and 2 hold. if $f(x) = 0 \Rightarrow \forall p : R(x)_p = 0$

Examples of Relevance

1. Put all relevance to one pixel
2. Divide the relevance equally between all input pixels
   $\forall p : R(x)_p = \frac{1}{d} f(x)$
3. Natural Decomposition: if the function f has some sort of natural decomposition between the input pixels.
   $f(x) = \sum_p f_p(x_p) \Rightarrow \forall p : R(x)_p = f_p(x_p)$
4. Taylor Decomposition around a reference point.
   $f(x) = f(\tilde{x}) + (\frac{\partial f}{\partial x}|x = \tilde{x})^\top (x - \tilde{x}) + \epsilon$

   $f(x) = 0 + \sum_p \frac{\partial f}{\partial x_p}|x = \tilde{x}(x_p - \tilde{x_p}) + \epsilon$

   $\forall p : R(x)_p = \frac{\partial f}{\partial x_p}|x = \tilde{x}(x_p - \tilde{x_p})$

Taylor Decomposition as Relevance

Taylor Decomposition around a reference point.

$$f(x) = f(\tilde{x}) + (\frac{\partial f}{\partial x}|x = \tilde{x})^\top (x - \tilde{x}) + \epsilon$$

$$f(x) = 0 + \sum_p \frac{\partial f}{\partial x_p}|x = \tilde{x} \times (x_p - \tilde{x_p}) + \epsilon$$

$$\forall p : R(x)_p = \frac{\partial f}{\partial x_p}|x = \tilde{x} \times (x_p - \tilde{x_p})$$

- Relevance: The amount of change in f when we substitute the reference point with our input image.
- Not good in practice:
  - Shattered (Noisy) Gradients
  - Adversarial Examples: small perturbation in x, changes f a lot.

Layer-wise Relevance Propagation

- Propagating prediction $f(x)$ backwards through the network to the input layer using local propagation rules.
- Highlight relevant and irrelevant regions over the input to the value of the prediction for a given class.
- Conservation property holds, both locally and globally.

Figure: Relevance of each pixels for the class 'Castle'

Local Layer-wise Relevance

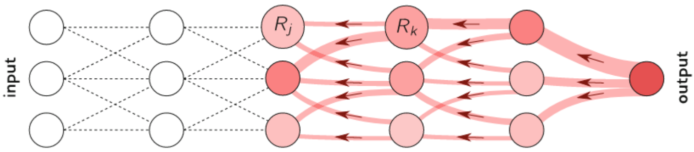


Figure: LRP in a glance

- Propagating relevance from neurons k at layer l_2 onto neuron j of the lower layer l_1 with the following rule:

$$R_j = \sum_{k \in A} \frac{z_{jk}}{\sum\limits_{j \in B_k} z_{jk}} R_k$$

Where ,
$A = \{n | n \in l_2, n \in N(j)\}$
$\forall k \in A, B_k = \{m | m \in l_1, k \in N(m)\}$
Note: be aware of the notation change!

Local Layer-wise Relevance

- Propagating relevance from neurons k at layer l_2 onto neuron j of the lower layer l_1 with the following rule:

  $$R_j = \sum_{k \in A} \frac{z_{jk}}{\sum_{j \in B_k} z_{jk}} R_k$$

  Where ,
  $A = \{n | n \in l_2, n \in N(j)\}$
  $\forall k \in A, B_k = \{m | m \in l_1, k \in N(m)\}$

- z_{jk} is the extent that neuron j has contributed to make neuron k relevant (i.e. activation of j times weight).

- $\frac{z_{jk}}{\sum_{j \in B_k} z_{jk}}$ resembles the proportion of relevance propagated from neuron k to neuron j. (Conservation property)

Notes on Relevance Rules

- Activations should be ReLu.
- Substitute z_{jk} with activation times weights:

$$R_j = \sum_{k \in A} \frac{a_j w_{jk}}{\sum_{j \in B_k} a_j w_{jk}} R_k$$

- The Rule:

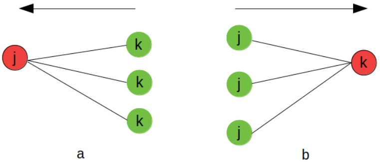


Figure: Propagation Rule. 'a' corresponds to the outer sum where we want to calculate the total amount of relevance going to the neuron j. 'b' corresponds to the inner sum in the denominator where we calculate the total amount of signal going to neuron k in order to calculate the proportion by which j has contribute to make k relevant

General Algorithm

1. **Forward Pass**: Start by feeding the image into the network and running the forward pass. Keep the activation values at each neuron.
2. **Initialize Relevance**: At the final layer (output), choose a class c (may not be the predicted class) and set the value of the relevance of that neuron R_c to its activation a_c * (softmax or sigmoid). Set the rest of the output neurons relevance to zero.
3. **Apply Relevance Rules**: propagate the relevance using the relevance rule(s) backward until you reach to input layer.
4. **Visualize**: by generating a heatmap over the relevance of input nodes, visualize the results.

LRP Rules

The general form of the LRP Rule:

$$R_j = \sum_k \frac{a_j \rho(w_{jk})}{\sum_j a_j \rho(w_{jk})} R_k$$

- LRP-0
- LRP-ϵ
- LRP-γ

LRP-0

The basic case which we saw earlier. $\rho(.)$ is identity function here.

$$R_j = \sum_k \frac{a_j w_{jk}}{\sum_j a_j w_{jk}} R_k$$

- We can show that this is simply Gradient $\times$ Input (the form we have in backprop algorithm). Thus it is unstable.

LRP-Epsilon

First enhancement of LRP-0. $\rho(.)$ here is again identity function. But a small positive term is added to the denominator to absorb weak or contradictory contribution.

$$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_j a_j w_{jk}} R_k$$

- Sparser and less noisy relevance.

LRP-Gamma

Another enhancement of LRP-0. $\rho(.)$ * here is the following function:

$$\rho(x) = (1+\gamma)^{\frac{sign(x)+1}{2}} x$$

If we apply this function to the LRP-0, we will get:

$$R_j = \sum_k \frac{a_j(w_{jk} + \gamma w_{jk}^{+})}{\epsilon + \sum_j a_j(w_{jk} + \gamma w_{jk}^{+})} R_k$$

- This favors the positive contributions more than negative ones. $(.)^{+}$ is basically $max(0,.)$.
- As we increase γ, the negative contributions become less and less important.

LRP Rules Comparison

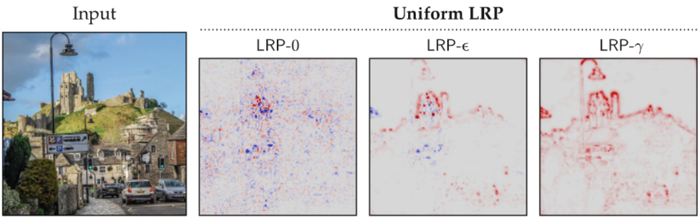


Figure: Comparison of using different LRP rules uniformly across the whole network.

Which Rule to use for each layer

- Measure of explanation quality (active research topic) *
  - Fidelity: accurate representation of the selected output neuron
  - Understandability: Easy to interpret for a human

Two strategies:
  - Uniform LRP
  - Composite Strategy

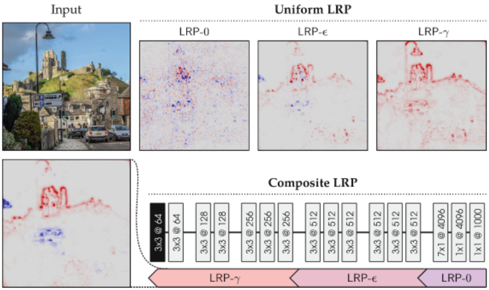


Figure: Comparing different LRP rules

Which Rule to use for each layer

- Uniform LRP-0
  - Tends to pick many local artifacts of the prediction functions (shattered gradient problem).

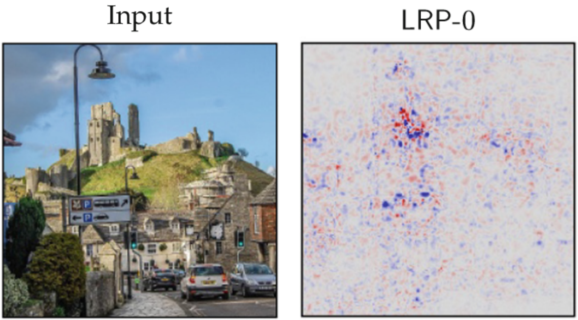


Figure: Input relevance using uniform LRP-0

Which Rule to use for each layer

- Uniform LRP-ϵ
  - Faithful and accurate representation, but due to sparsity it is hard to interpret by human.

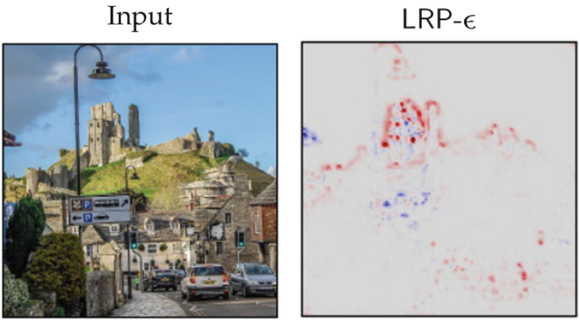


Figure: Input relevance using uniform LRP-ϵ

Which Rule to use for each layer

- Uniform LRP-γ
  - It is understandable by humans because of dense highlighted features.
  - But it picks unrelated features such as lamp post.

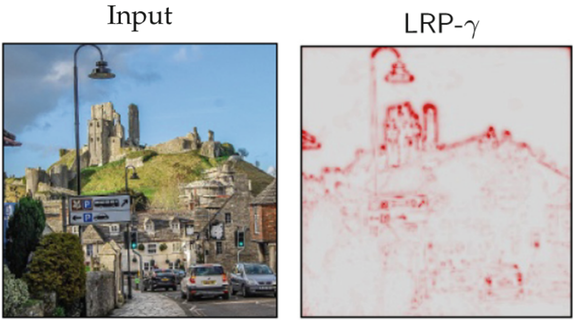


Figure: Input relevance using uniform LRP-γ

Which Rule to use for each layer

- Composite LRP
  - Upper Layers (fully connected in top part): LRP-0 Concepts are entangled. Gradient is less sensitive to these entanglements. (Here we can tolerate the gradient problems because of these entanglements).
  - Middle Layers: LRP-ϵ Weight sharing in convolution introduces spurious variations which can be filtered out using this rule. Only important explanations remain.
  - Lower Layers: LRP-γ Same problem as middle layers. Either ϵ or γ should work. But later is better because it has a stronger effect in spreading the explanations to features rather than actual pixels.

Which Rule to use for each layer

- Composite LRP
  - As you see, we have both fidelity and understandability.

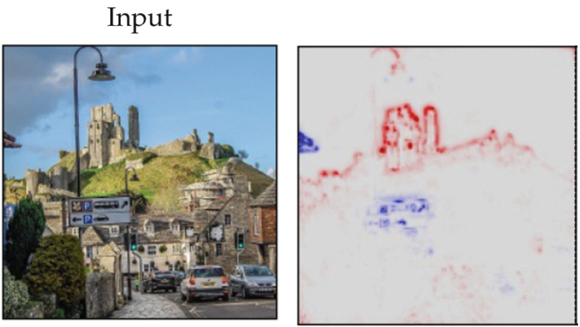


Figure: Input relevance using composite LRP

Different starting relevance for the output layer *

- What we tried to explain so far:
  - Version 1:

$$R_c = P(z_c) = \frac{e^{z_c}}{\sum_{c'} e^{z_{c'}}}$$

  - Version 2 (This one is more stable):

$$R_c = z_c = \sum_k a_k w_{kc}$$

Different starting relevance for the output layer **

- Instead we can try to explain another type of score:
  - Explain the presence of an object, when other objects from other classes are present in the image. (I guess we can do this in a pairwise manner too)

$$\eta_c = log(\frac{P(z_c)}{1 - P(z_c)}) = log(\frac{P(z_c)}{\sum_{c'' \neq c} P(z_{c''})})$$

$$z_{c,c''} = log(\frac{P(z_c)}{P(z_{c''})}) = log(\frac{\frac{e^{z_c}}{\sum_{c'} e^{z_{c'}}}}{\frac{e^{z_{c''}}}{\sum_{c'} e^{z_{c'}}}}) = log(\frac{e^{z_c}}{e^{z_{c''}}})$$

$$= log(e^{z_c - z_{c''}}) = z_c - z_{c''} = \sum_k a_k(w_{kc} - w_{kc''})$$

Different starting relevance for the output layer **

- Now we can use this $z_{c,c''}$ to calculate a new Relevance for the neuron c in the output.

$$z_{c,c''} = log(\frac{P(z_c)}{P(z_{c''})}) = log(\frac{\frac{e^{z_c}}{\sum_{c'} e^{z_{c'}}}}{\frac{e^{z_{c''}}}{\sum_{c'} e^{z_{c'}}}}) = log(\frac{e^{z_c}}{e^{z_{c''}}})$$

$$= log(e^{z_c - z_{c''}}) = z_c - z_{c''} = \sum_k a_k(w_{kc} - w_{kc''})$$

$$R_{c,c''} = z_{c,c''} \times \frac{e^{-z_{c,c''}}}{\sum_{c' \neq c} e^{-z_{c,c'}}}$$

Different starting relevance for the output layer **

- The new relevance:

$$R_{c,c''} = z_{c,c''} \times \frac{e^{-z_{c,c''}}}{\sum_{c' \neq c} e^{-z_{c,c'}}}$$

- This will result in better explanations.

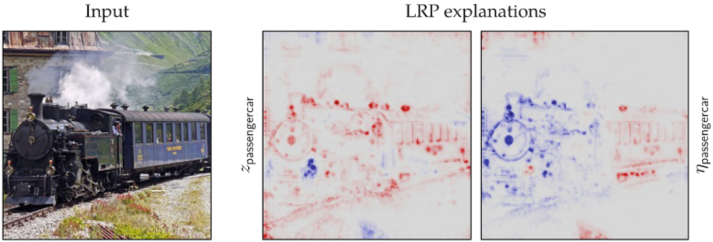


Figure: Comparing old initialized relevance for c and the new one.

Conclusion

- There are still some vague things for myself.
  - Is manipulating rules to get better explanations OK?
  - All those *s!
- We can illustrates the regions that the network is paying more attention (positive or negative)
- We can explain why the network is making (or not making) a particular decision
- We can use Deep Learning for sensitive tasks with a little bit more peace of mind.

Thank You for your attention!