Interpretable Deep Learning: Towards Understanding & Explaining DNNs

Part 2: Methods of Explanation

Wojciech Samek, Grégoire Montavon, Klaus-Robert Müller
What Will be Covered in Part 2

interpreting predicted classes

explaining individual decisions
Q: Where in the image do the neural networks see evidence for a car?
Examples of Methods that Explain Decisions

- Baehrens'10 Gradient
- Sundarajan'17 Int Grad
- Zintgraf'17 Pred Diff
- Haufe'15 Pattern
- Zurada'94 Gradient
- Symonian'13 Gradient
- Zeiler'14 Occlusions
- Ribeiro'16 LIME
- Poulin'06 Additive
- Lundberg'17 Shapley
- Fong'17 M Perturb
- Kindermans'17 PatternNet
- Zeiler'14 Deconv
- Zeiler'14 Contrib Prop
- Montavon'17 Deep Taylor
- Shrikumar'17 DeepLIFT
- Caruana'15 Fitted Additive
- Landecker'13 Taylor
- Bach'15 LRP
- Zhang'16 Excitation BP
- Springenberg'14 Guided BP
- Zhou'16 GAP
- Selvaraju'17 Grad-CAM
Q: In which proportion has each car contributed to the prediction?
Explaining by Decomposing

**Goal:** Determine the share of the output that should be attributed to each input variable.

\[ \sum_{i=1}^{d} R_i = f(x_1, \ldots, x_d) \]
Explaining by Decomposing

**Goal:** Determine the **share** of the output that should be attributed to each input variable.

Decomposing a prediction is generally difficult.
Sensitivity Analysis

computes for each pixel:

\[ R_i = \left( \frac{\partial f}{\partial x_i} \right)^2 \]
Sensitivity Analysis

Question: If sensitivity analysis computes a decomposition of something: Then, what does it decompose?

\[ R_i = \left( \frac{\partial f}{\partial x_i} \right)^2 \]

\[ \sum_{i=1}^{d} R_i = \| \nabla f(x) \|^2 \]
Sensitivity Analysis

Sensitivity analysis explains a variation of the function, not the function value itself.

input explanation for “car”

variation = make something appear less/more a car.

\[ \sum_{i=1}^{d} R_i = \| \nabla f(x) \|^2 \]
The Taylor Expansion Approach

1. Take a linear model:
   \[ f(x) = \sum_{i=1}^{d} w_i x_i + b \]

2. First-order expansion at root point:
   \[ f(x) = \sum_{i=1}^{d} \frac{\partial f}{\partial x_i} \bigg|_{\tilde{x}} \cdot (x_i - \tilde{x}_i) \]

3. Identifying linear terms:
   \[ R_i = w_i \cdot (x_i - \tilde{x}_i) \]

   a decomposition

**Observation:** explanation depends on the root point.
The Taylor Expansion Approach

Obtained relevance scores

\[ R_i = w_i \cdot (x_i - \tilde{x}_i) \]

How to choose the root point?

- Closeness to the actual data point
- Membership to the input domain (e.g. pixel space)
- Membership to the data manifold.
Non-Linear Models

Nonlinear model

\[ f(x) = \sum_j \rho \left( \sum_{i=1}^d w_{ij} x_i + b_j \right) \]

\[ f(x) = \sum_{i=1}^d \left. \frac{\partial f}{\partial x_i} \right|_{\tilde{x}} \cdot (x_i - \tilde{x}_i) + o(xx^T) \]

second-order terms are hard to interpret and can be very large

Simple Taylor decomposition is not suitable for highly non-linear models.
Overcoming NonLinearity

Integrated Gradients [Sundararajan’17]:

\[
f(x) = \int_{\xi=\tilde{x}}^{x} \sum_{i=1}^{d} \left. \frac{\partial f}{\partial x_i} \right|_{\xi} \cdot d\xi_i
\]

\[
f(x) = \sum_{i=1}^{d} \int_{\xi=\tilde{x}}^{x} \left. \frac{\partial f}{\partial x_i} \right|_{\xi} \cdot d\xi_i
\]

- Fully decomposable
- Require computing an integral (expensive)
- Which integration path?

Overcoming NonLinearity

Special case when the origin is a root point and the gradient along the integration path is constant:

\[ f(x) = \sum_{i=1}^{d} \left( \int_{\xi=0}^{x} \frac{\partial f}{\partial x_i} \cdot d\xi_i \right) \]

\[ f(x) = \sum_{i=1}^{d} \frac{\partial f}{\partial x_i} \cdot x_i \]

gradient x input
Let’s consider a different approach ...
View the decision as a **graph computation** instead of a function evaluation, and propagate the decision backwards until the input is reached.
Layer-Wise Relevance Propagation (LRP) [Bach’15]

\[
R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k
\]
Gradient-Based vs. LRP

grad × input

LRP-$\alpha_1\beta_0$
Layer-Wise Relevance Propagation (LRP) [Bach’15]

Carefully engineered propagation rule: e.g. LRP-\(\alpha_1\beta_0\)

\[
R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k
\]

pooling received messages

neuron contribution

normalization term

available for redistribution
LRP Propagation Rules: Two Views

View 1:

\[ R_j = \sum_k \left( \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} \right) R_k \]

- \( R_k \): available for redistribution
- \( a_j w_{jk}^+ \): neuron contribution
- \( \sum_j a_j w_{jk}^+ \): normalization term
- \( \sum_k \): pooling received messages

View 2:

\[ R_j = a_j \left( \frac{\sum_k w_{jk}^+}{\sum_j a_j w_{jk}^+} \right) R_k \]

- \( R_k \): available for redistribution
- \( a_j \): neuron activation
- \( \sum_k w_{jk}^+ \): weighted sum
- \( \sum_j a_j w_{jk}^+ \): normalization term
Implementing Propagation Rules (1)

\[ R_j = a_j \sum_k w_{jk}^+ \frac{R_k}{\sum_j a_j w_{jk}^+} \]

**Element-wise operations**

- \( z_k \rightarrow \sum_j a_j w_{jk}^+ \)
- \( s_k \rightarrow R_k / z_k \)
- \( c_j \rightarrow \sum_k w_{jk}^+ s_k \)
- \( R_j \rightarrow a_j c_j \)

**Vector operations**

- \( z \rightarrow W_+^\top \cdot a \)
- \( s \rightarrow R \odot z \)
- \( c \rightarrow W_+ \cdot s \)
- \( R \rightarrow a \odot c \)
Implementing Propagation Rules (2)

Code that reuses forward and gradient computations:

```python
def lrp(layer, a, R):
    clone = layer.clone()
    clone.W = maximum(0, layer.W)
    clone.B = 0

    z = clone.forward(a)
    s = R / z
    c = clone.backward(s)

    return a * c
```

See also http://www.heatmapping.org/tutorial
How Fast is LRP?

GPU-based implementation of LRP: Check out iNNvestigate [Alber’18]
https://github.com/albermax/innvestigate
Is there an underlying mathematical framework for LRP?
**Deep Taylor Decomposition** [Montavon’17]

**Question:** Suppose that we have propagated the relevance until a given layer. How should it be propagated one layer further?

**Idea:** By performing a Taylor expansion of the relevance.

$$R_j = \sum_k \frac{a_j w_{jk}^+}{\sum_j a_j w_{jk}^+} R_k$$
Observation: Relevance at each layer is a product of the activation and an approximately locally constant term.
Deep Taylor Decomposition

Relevance neuron:

\[ R_j(a) = \max(0, \sum_i a_i w_{ij} + b_j) \cdot c_j \]

Taylor expansion:

\[ R_j(a) = \sum_i \left. \frac{\partial R_j}{\partial a_i} \right|_{\tilde{a}^{(j)}} \cdot (a_i - \tilde{a}_i^{(j)}) \]

Redistribution:

\[ R_{i \leftarrow j} = \frac{(a_i - \tilde{a}_i^{(j)}) w_{ij}}{\sum_i (a_i - \tilde{a}_i^{(j)}) w_{ij}} R_j \]
Choosing the Root Point

\[ R_{i \leftarrow j} = \frac{(a_i - \tilde{a}_i^{(j)})w_{ij}}{\sum_i (a_i - \tilde{a}_i^{(j)})w_{ij}} R_j \]  

(Deep Taylor generic)

---

### Choice of root point

<table>
<thead>
<tr>
<th>1. nearest root</th>
<th>( \tilde{a}^{(j)} = a - t \cdot w_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. rescaled excitations</td>
<td>( \tilde{a}^{(j)} = a - t \cdot a \odot 1_{w_j &gt; 0} )</td>
</tr>
</tbody>
</table>

(same as LRP-\( \alpha_1 \beta_0 \))

\[ R_{i \leftarrow j} = \frac{a_i w_{ij}^+}{\sum_i a_i w_{ij}^+} R_j \]
Choosing the Root Point

\[
R_{i \leftarrow j} = \frac{(x_i - \tilde{x}_i^{(j)})w_{ij}}{\sum_i(x_i - \tilde{x}_i^{(j)})w_{ij}} R_j \quad \text{(Deep Taylor generic)}
\]

**Pixels domain:**

\[
[l, h]^{3 \times d}
\]

Choice of root point

\[
\tilde{x}^{(j)} = x - t \cdot (x - l \odot 1_{w_j > 0} - h \odot 1_{w_j < 0})
\]

Resulting propagation rule

\[
R_{i \leftarrow j} = \frac{x_{ij}w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_i x_{ij}w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-} R_j
\]
Choosing the Root Point

Choice of root point

Resulting propagation rule

\[
R_{i \leftarrow j} = \frac{(x_i - \tilde{x}_i^{(j)})w_{ij}}{\sum_i(x_i - \tilde{x}_i^{(j)})w_{ij}} \cdot R_j \quad \text{(Deep Taylor generic)}
\]
DTD View on Explaining a ConvNet [Montavon’17]
One-class SVM rewritten as a min-pooling over distances:

Deep Taylor decomposition:

\[
R_i = \sum_j \frac{(x_i - u_{ij})^2}{\|x - u_j\|^2_2} (R_j - D_j^+) \quad \text{and} \quad R_j = (a_j + \varepsilon_j) \cdot \frac{\exp(-a_j)}{\sum_j \exp(-a_j)}
\]
DTD-OCSVM on MNIST

dataset

outlier digits

Outlier Model $\alpha(x)$ Explanation

pixel-wise explanation why they are outliers
DTD-OCSVM on Images

input

explanation for outlierness

□ → patch size
Conclusion for Part 2

1. Explaining deep neural networks is non-trivial. Simple gradient-based methods either do not ask the right question, or are difficult to scale to deep models.

2. Propagation-based approaches (e.g. LRP) seem to work better on complex DNN models. (This will be validated in Part 3).

3. Deep Taylor Decomposition provides a theoretical framework for understanding and deriving LRP-type explanation procedures.