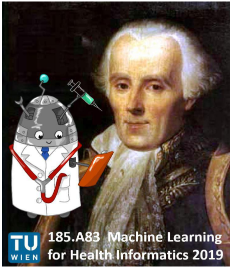
Layer-wise Relevance Propagation (LRP)
Machine Learning for Health Informatics

Anna Saranti

Holzinger Group hci-kdd.org

26.03.2019



Outline

Taylor Decomposition

- Taylor expansion of a function $f(x)$ at point a:
  $f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$
- Classification (output) $f(\mathbf{x})$ of input $\mathbf{x}$ (1st order only):

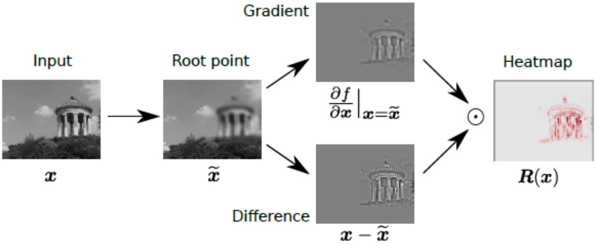
$$f(\mathbf{x}) = f(\tilde{\mathbf{x}}) + \left(\left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}=\tilde{\mathbf{x}}} \right)^T (\mathbf{x} - \tilde{\mathbf{x}}) + \epsilon$$

$$0 + \sum_p \underbrace{\left. \frac{\partial f}{x_p} \right|_{\mathbf{x}=\tilde{\mathbf{x}}} (x_p - \tilde{x}_p)}_{R_p(\mathbf{x})} + \epsilon$$

  where p is the index of the pixel.
- Find a neighbouring point $\tilde{\mathbf{x}}$, for which $f(\tilde{\mathbf{x}}) = 0$
- $\sum_j R_j = \left(\left. \frac{\partial(\sum_j R_j)}{\partial\{x_i\}} \right|_{\partial\{\tilde{x}_i\}} \right)^T (\{x_i\} - \{\tilde{x}_i\}) + \epsilon =$
  $\sum_i \sum_j \left. \frac{\partial R_j}{\partial x_i} \right|_{\partial\{\tilde{x}_i\}} (x_i - \tilde{x}_i) + \epsilon$

Pixel-wise decomposition of a function

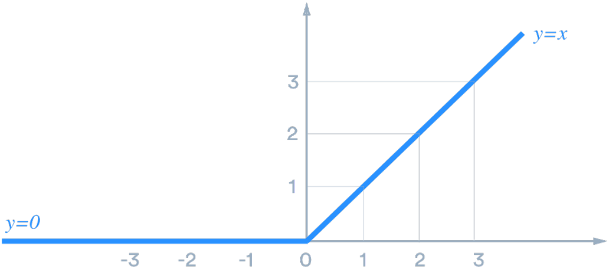
- Goal: redistribute the neural network output onto the input variables; the relevance R_j to lower-level relevances $\{R_i\}$
- How to choose the neighbouring point $\tilde{x}$?
- Similar image, object not recognizable from the classifier - hence the output $f(\tilde{\mathbf{x}}) = 0$



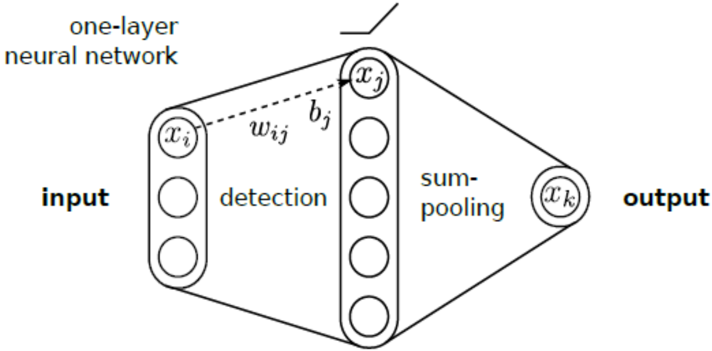
Properties

- Conservation: $\forall \mathbf{x} : f(\mathbf{x}) = \sum_p R_p(\mathbf{x})$
  $\sum_j R_j = \sum_i R_i$ (i and j are layers)
- Positivity: $\forall \mathbf{x}, p : R_p(\mathbf{x}) \geq 0$

- Rectified Linear Unit:



Example (1/2)



- $x_j = max(0, \sum_i x_i w_{ij} + b_j)$ (ReLU nonlinearity)
- $x_k = \sum_j x_j$ (Sum pooling)

Example (2/2)

R_k of output layer: Total relevance that must be backpropagated:

- $R_k = x_k = \sum_j x_j$

R_j of hidden layer: Taylor decomposition on $\{\tilde{x}_j\} = 0$:

- $R_j = R_k(\tilde{\mathbf{x}}) + \frac{\partial R_k}{\partial x_j}\Big|_{\{\tilde{x}_j\}} \cdot (x_j - \tilde{x}_j) = x_j = max(0, \sum_i x_i w_{ij} + b_j)$
- For which $\tilde{\mathbf{x}}$ is $R_k(\tilde{\mathbf{x}}) = 0$?
  Since ReLU ensures that $\{\forall j : \tilde{x}_j \geq 0\}$ and
  $\frac{\partial R_k}{\partial x_j} = \frac{\partial \sum_j x_j}{\partial x_j} = 1$

R_i of input layer:

- $R_i = \sum_j \frac{\partial R_j}{\partial x_i}\Big|_{\{\tilde{x}_i\}^{(j)}} \cdot (x_i - \tilde{x}_i^{(j)})$
- $R_i = \sum_j \frac{w_{ij}^2}{\sum_{i'} w_{i'j}^2} R_j$

Task (1/2)

The task contains two parts

1. Numerical task
   - Use the equations above to compute numerically the relevance of all layers of the network depicted in the figure.
   - Use your own weight values (w_{ij}), but think on weighting schemes that are typically used in neural networks. See `https://keras.io/initializers/`
   - Verify that the conservation and positivity rules properties apply.
   - Provide descriptions of the interpretations
2. Programmatic task
   - Install Python $3.5.+$ and the relevant libraries.
   - Provide descriptions of the interpretations of the relevance images with respect to the input images as well as their differences

Task (2/2)

1. Python libraries
   - https://www.tensorflow.org/, https://mxnet.apache.org/
   - https://keras.io/
   - https://github.com/albermax/innvestigate
   - Run the examples/readme_code_snippet.py with any of the .jpg figures in the examples/images folder (line 37)
   - Adapt line 54 to select a different analyzer
   - Python IDE: https://www.jetbrains.com/pycharm/

Literature

- Montavon, Grégoire, et al. "Explaining nonlinear classification decisions with deep taylor decomposition." Pattern Recognition 65 (2017): 211-222.