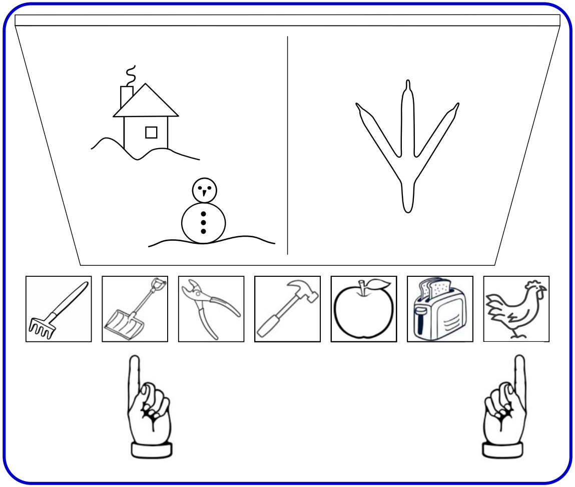
Post Training Analysis

Deep Learning

Explainability

- After training, analyze the network.
- One purpose of the analysis is to explain network decision making.
- In many applications (e.g., medicine) users need to understand the decisions in order to have confidence in them.
- The objective of explainability is to provide a narrative that gives the user confidence in the decisions.
- We will discuss techniques that explain how deep networks are making their decisions.

How good are human explanations?



Thoughts on explainability

- If deep networks take on complex tasks, their explanations may also be complex.
- We will discuss methods that provide insights into deep network operation.
- They don't provide complete explanations.
- We use explainability in a technical way that does not include all meanings of its everyday usage.

Definitions of explain

- Merriam Webster's – to give the reason for or cause of.
- Related words – clarify, explicate, illustrate, interpret.
- How can we apply this concept to deep learning and find a quantitative measure?
- There will never be "ground truth", in that we will never be certain our explanation is correct.
- Explainability methods are mainly judged on their ability to provide a convincing narrative.

Good definition, 1977

Weight of evidence

$$W(H : E) = \ln \frac{P(E|H)}{P(E|\bar{H})}$$

- H is an hypothesis, and E is evidence for that hypothesis.
- $P(E|H)$ is the conditional probability of E given H.
- $\bar{H}$ is the complement of H.
- If E was more likely to occur when H was true, the weight of evidence would be positive.

Example

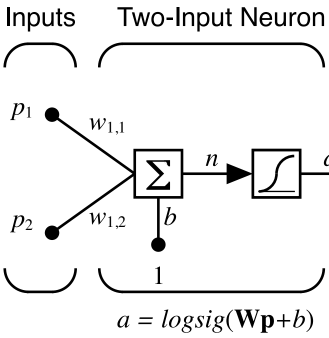
One layer logsig network.

$$
\begin{aligned}
n_1 &= w_{1,1}p_1 + w_{1,2}p_2 \\
a_1 &= \frac{1}{1+e^{-n_1}}
\end{aligned}
$$

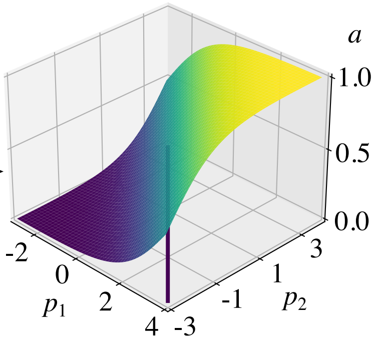
$$
\begin{aligned}
W(H:E) &= \ln \frac{P(E|H)}{P(E|\bar{H})} = \ln \frac{\frac{P(H|E)P(E)}{P(H)}}{\frac{P(\bar{H}|E)P(E)}{P(\bar{H})}} = \ln \frac{P(H|E)P(\bar{H})}{P(\bar{H}|E)P(H)} \\
&= \ln \frac{P(H|E)}{P(\bar{H}|E)} = \ln \frac{a_1}{1-a_1} = \ln \frac{\frac{1}{1+e^{-n_1}}}{\frac{e^{-n_1}}{1+e^{-n_1}}} = \ln e^{n_1} = n_1 \\
&= \boxed{w_{1,1}p_1} + w_{1,2}p_2
\end{aligned}
$$

first input contribution

Illustrative example: single neuron



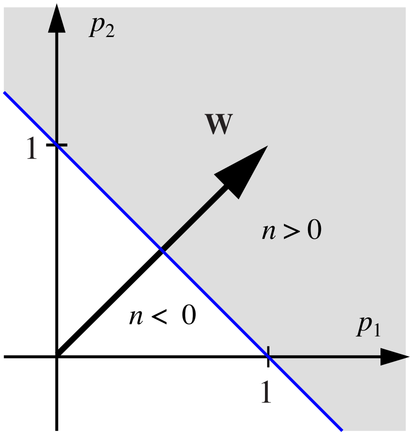
One Layer Network



Network Response

$$a = f(n) = f(\mathbf{W}\mathbf{p} + b) = logsig(1p_1 + 1p_2 - 1)$$

Global explanation



Linear Decision Boundary

Local explanation

$$n = 1p_1 + 1p_2 - 1$$
$$a = logsig(n)$$

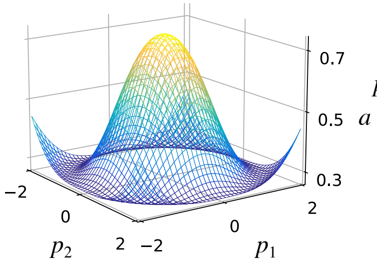
- The global explanation is the weight (decision boundary).
- The weights have equal magnitude, so globally both inputs are equally important.
- For a particular example, the weights times the inputs are the local explanations.

$$\mathbf{p} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

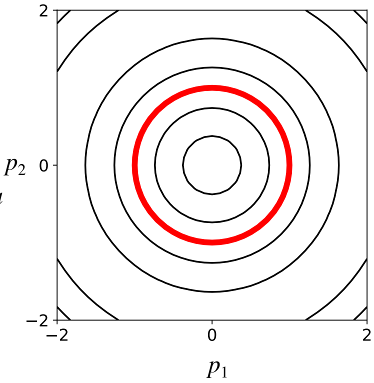
$$n = \overset{\text{main contribution}}{\boxed{1 \times 2}} + 1 \times 0 - 1$$

Example: radial net

Global Explanation



Network Response Surface



Contour Plot and Decision Boundary

Linearizing the network response

Taylor Series Expansion

$$a(\mathbf{p}) \cong a(\mathbf{p}^*) + \nabla a(\mathbf{p})^T\big|_{\mathbf{p}^*} (\mathbf{p} - \mathbf{p}^*)$$

- $\mathbf{p}^*$ is the input where the expansion takes place.
- $\nabla a(\mathbf{p})$ is the gradient.

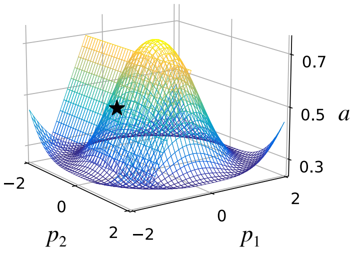
$$\nabla a(\mathbf{p}) = \begin{bmatrix} \frac{\partial a(\mathbf{p})}{\partial p_1} & \frac{\partial a(\mathbf{p})}{\partial p_2} & \dots & \frac{\partial a(\mathbf{p})}{\partial p_R} \end{bmatrix}^T$$

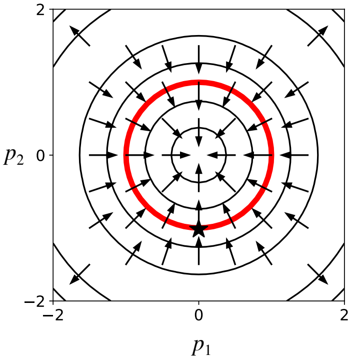
- The gradient is the weight in the approximate linear network.

Linearize the radial network

$$\mathbf{p}^* = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$a(\mathbf{p}) \cong 0.5 + \begin{bmatrix} 0 & 0.4 \end{bmatrix} \left(\mathbf{p} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 0 & 0.4 \end{bmatrix} \mathbf{p} + 0.9$$





Notation and General Explainability Definitions

- We use Good/Poulin definition of contribution, but extended to multilayer networks.
- For a single-layer network, it is equal to the product of the input times the weight.
- We use two methods to extend this approach to multilayer networks.
- First, linearize the network about the relevant input vector.
- Second, backpropagate the contributions through the network.

Desired properties of contributions

1. Continuity: If $a_c^M(\mathbf{p})$ is continuous, then $\mathbf{c}(\mathbf{p})$ is continuous.
2. Implementation Invariance: If two networks are functionally equivalent (same input produces same output), then the computed contributions are the same.
3. Sensitivity
   1. If an input and a baseline differ in one feature and have different network outputs, a nonzero contribution is assigned to that feature.
   2. If the network output does not depend on a feature, the contribution of that feature is zero.
4. Completeness: The sum of all contributions is equal to the network output (or the difference between the output and a baseline output).

Derivative of Network Output with Respect to Input

$$c^{sal}(p_i) = \frac{\partial n_c^M(\mathbf{p})}{\partial p_i} \times p_i$$

$$\mathbf{s}^m = \dot{\mathbf{F}}^m(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1}$$

$$\mathbf{s}^M = \frac{\partial n_c^M}{\partial \mathbf{n}^M} = \boldsymbol{\epsilon}_c$$

$$\mathbf{c}^{sal}(\mathbf{p}) = \frac{\partial n_c^M}{\partial \mathbf{p}} \circ \mathbf{p} = \left(\frac{\partial (\mathbf{n}^1)^T}{\partial \mathbf{p}} \frac{\partial n_c^M}{\partial \mathbf{n}^1} \right) \circ \mathbf{p} = \left((\mathbf{W}^1)^T \mathbf{s}^1 \right) \circ \mathbf{p}$$

Integrated gradient

- The gradient can be zero at $\mathbf{p}$, even when the network is confident.
- Average the gradient over a line between the current input $\mathbf{p}$ and a baseline input $\overline{\mathbf{p}}$ to get a more consistent contribution.

$$\mathbf{c}^{ig}(\mathbf{p}) = \left[\frac{1}{T} \sum_{t=0}^{T-1} \frac{\partial n_c^M(\tilde{\mathbf{p}})}{\partial \tilde{\mathbf{p}}} \bigg|_{\tilde{\mathbf{p}}=\mathbf{p}^t} \right] \circ (\mathbf{p} - \overline{\mathbf{p}})$$

$$\mathbf{p}^t = \overline{\mathbf{p}} + \frac{t}{T-1} (\mathbf{p} - \overline{\mathbf{p}})$$

Contribution propagation

Compute contribution at last layer and then backpropagate it.

$$\mathbf{c}(\mathbf{a}^M) = n_c^M \boldsymbol{\epsilon}_c$$

$$\mathbf{c}(\mathbf{a}^m) = \mathbf{C}\left(\mathbf{a}^m|\mathbf{a}^{m+1}\right)\mathbf{c}(\mathbf{a}^{m+1}), \quad m = M-1, M-2, \cdots, 0$$

$$\left[\mathbf{C}\left(\mathbf{a}^m|\mathbf{a}^{m+1}\right)\right]_{i,j} = c\left(a_i^m|a_j^{m+1}\right)$$

To maintain total contribution.

$$\sum_{i=1}^{S^m} c(a_i^m) = n_c^M, \text{ if } \sum_{i=1}^{S^m} c\left(a_i^m|a_j^{m+1}\right) = 1$$

Layerwise Relevance Propagation (LRP)

LRP-0

$$c^{lrp-0}\left(a_i^m | a_j^{m+1}\right) = \frac{w_{j,i}^{m+1} a_i^m}{\sum_{k=1}^{S^m} w_{j,k}^{m+1} a_k^m}$$

LRP-ϵ

$$c^{lrp-\epsilon}\left(a_i^m | a_j^{m+1}\right) = \frac{w_{j,i}^{m+1} a_i^m}{\epsilon + \sum_{k=1}^{S^m} w_{j,k}^{m+1} a_k^m}$$

LRP-γ

$$c^{lrp-\gamma}\left(a_i^m | a_j^{m+1}\right) = \frac{\left(w_{j,i}^{m+1} + \gamma\left(w_{j,i}^{m+1}\right)^{+}\right) a_i^m}{\sum_{k=1}^{S^m}\left(w_{j,k}^{m+1} + \gamma\left(w_{j,k}^{m+1}\right)^{+}\right) a_k^m}$$

DeepLIFT

DeepLift measures contributions from a baseline $\overline{\mathbf{p}}$.

$$\Delta n_c^M = n_c^M(\mathbf{p}) - n_c^M(\overline{\mathbf{p}}) = n_c^M - \overline{n}_c^M$$

Contributions are defined in terms of changes.

$$c^{dl}\left(\Delta a_i^m\right) = c^{dl}\left(\Delta a_i^m | \Delta n_c^M\right) = c\left(a_i^m\right)$$

$$c^{dl}\left(\Delta a_i^m | \Delta a_j^{m+1}\right) = c\left(a_i^m | a_j^{m+1}\right) \Delta a_j^{m+1}$$

Modified multipliers are divided by change.

$$q\left(\Delta a_i^m | \Delta a_j^{m+1}\right) = \frac{c^{dl}\left(\Delta a_i^m | \Delta a_j^{m+1}\right)}{\Delta a_i^m} = c\left(a_i^m | a_j^{m+1}\right) \frac{\Delta a_j^{m+1}}{\Delta a_i^m}$$

$$q\left(\Delta a_i^m\right) = \frac{c^{dl}\left(\Delta a_i^m\right)}{\Delta a_i^m} = c\left(a_i^m\right) \frac{1}{\Delta a_i^m}$$

DeepLIFT propagation equations

$$q\left(\Delta a_c^M\right) = 1$$

$$q\left(\Delta a_i^m\right) = \sum_{j=1}^{S^{m+1}} q\left(\Delta a_i^m | \Delta n_j^{m+1}\right) q\left(\Delta n_j^{m+1} | \Delta a_j^{m+1}\right) q\left(\Delta a_j^{m+1}\right)$$

Rescale Rule

$$q\left(\Delta n_j^{m+1} | \Delta a_j^{m+1}\right) = \frac{\Delta a_j^{m+1}}{\Delta n_j^{m+1}}$$

Linear Rule

$$q\left(\Delta a_i^m | \Delta n_j^{m+1}\right) = w_{j,i}^{m+1}$$

DeepLIFT Contributions

$$c^{dl}\left(\Delta p_i\right) = q\left(\Delta p_i\right) \Delta p_i$$