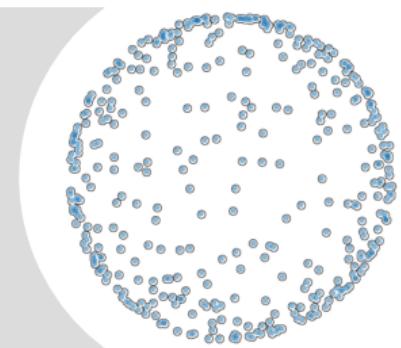


WiSe 2024/25

Machine Learning for Data Science

Lecture by G. Montavon



Lecture 12a

Explainable AI

Outline

Intro to Explainable AI

- ▶ Why explainable AI
- ▶ Types of explanations
- ▶ Uses of explanations

Explainable AI Techniques

- ▶ Activation maximization
- ▶ Attribution via Shapley Values
- ▶ Attribution via LRP

Application to a Data Science Problem

- ▶ Finding influential proteins from proteomics data.

What insights ML models give us?

PCA/Clustering/etc:

- ▶ Provides a decomposition of the variance in data into different components or between what is explained/unexplained by clusters.
- ▶ Provide prototypes (direction in feature space, cluster centroids, that describe the data).

Linear Regression / Discriminant

- ▶ Demonstrates that y is predictable up to some accuracy from \boldsymbol{x} .
- ▶ Provides directions in feature space that maximally correlate with the other modality (e.g. real-valued target or class).

Nonlinear Models (Kernels / Deep Networks):

- ▶ Demonstrates that y is predictable up to some accuracy from \boldsymbol{x} .

Question: Can we get more insights from these models?

Types of Explanation

Example 1: Synthesize an input pattern that most strongly activates the output of the ML model (e.g. associated to a particular class).



Image source: Nguyen et al. (2016) Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks

Types of Explanation

Example 2: Highlight features that have contributed for a given data point to the ML prediction.

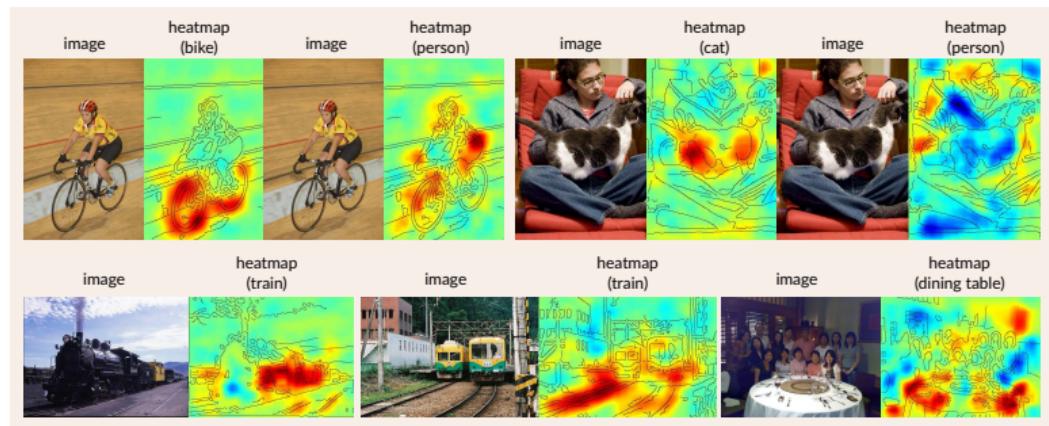
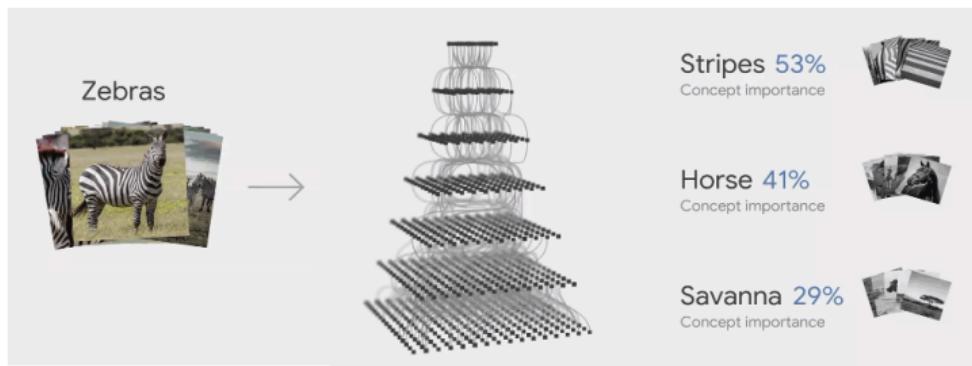


Image source: Lapuschkin et al. (2016) Analyzing Classifiers: Fisher Vectors and Deep Neural Networks

Types of Explanation

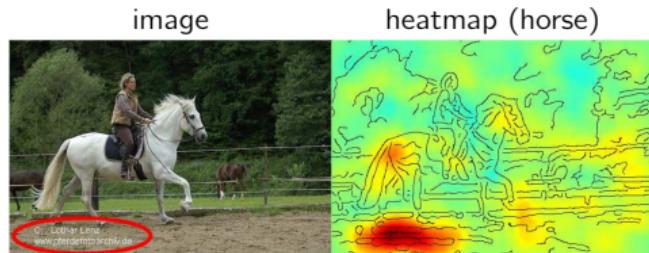
Example 3: Concept activation vectors (TCAV). Highlight the mid-level concepts that explain, for a given data point, the ML prediction.



Source: Google Keynote'19 (URL: <https://www.youtube.com/watch?v=lyRPyRKH08M&t=2279s>)

Uses of an Explanation

Example: The classifier is right for the wrong reasons



average precision of the Fisher Vector model on the PascalVOC dataset

aer	bic	bir	boa	bot
79.08	66.44	45.90	70.88	27.64
bus	car	cat	cha	cow
69.67	80.96	59.92	51.92	47.60
din	dog	hor	mot	per
58.06	42.28	80.45	69.34	85.10
pot	she	sof	tra	tvm
28.62	49.58	49.31	82.71	54.33

- ▶ In this example, the classifier accurately predicts the horse class, but based on the wrong features (some copyright tag in the corner).
- ▶ This incorrect decision strategy cannot be detected by just looking at the test error.

cf. Lapuschkin et al. (2019) Unmasking Clever Hans Predictors and Assessing What Machines Really Learn. Nature Communications

Uses of an Explanation

Learn something about the data

(or about the system that produced the data)

- ▶ Step 1: Train a ML model that predicts well the data.
- ▶ Step 2: Apply XAI to the trained ML model to produce explanations of the ML decision strategy.
- ▶ Step 3: Based on the XAI explanations, the users can compare their reasoning with that of the ML model, and can potentially refine their own domain knowledge.

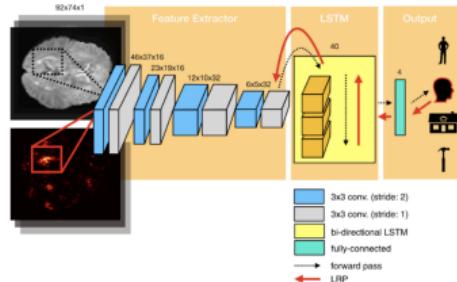


Image source: Thomas et al. (2019) Analyzing Neuroimaging Data Through Recurrent Deep Learning Models

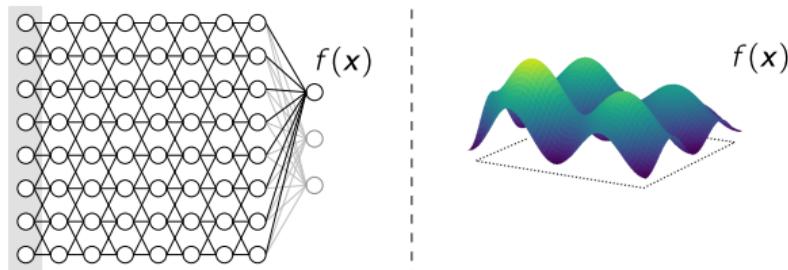
Part 1

Activation Maximization

Activation Maximization

Assume a trained a ML model (e.g. a neural network), and we would like to understand what concept is associated to some particular output neuron of the ML model, e.g. the output neuron that codes for the class 'cat'. Activation maximization proceeds in two steps:

- ▶ **Step 1:** Think of the ML model as a function of the input



- ▶ **Step 2:** Explain the function f by generating a maximally activating input pattern:

$$\mathbf{x}^* = \arg \max_{\mathbf{x}} f(\mathbf{x})$$

Activation Maximization

Problem:

- ▶ In most cases $f(\mathbf{x})$ does not have a single point corresponding to the maximum. E.g. in linear models, $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, we can keep moving the point \mathbf{x} further along the direction \mathbf{w} , and the output continues to grow).

Solution:

- ▶ Apply a preference for 'regular' regions of the input domain, i.e.

$$\mathbf{x}^* = \arg \max_{\mathbf{x}} f(\mathbf{x}) - \Omega(\mathbf{x})$$

In practice, the preference can be for data points with small norm (i.e. we set $\Omega(\mathbf{x}) = \lambda \|\mathbf{x}\|^2$ so that points with large norm are penalized.)

Activation Maximization: Examples

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b \text{ and } \Omega(\mathbf{x}) = \lambda \|\mathbf{x}\|^2$$

$$f(\mathbf{x}) = \max(x_1, x_2) \text{ and } \Omega(\mathbf{x}) = \lambda \|\mathbf{x}\|^2$$

$$\underset{\mathbf{x}}{\operatorname{argmax}} \underbrace{f(\mathbf{x}) - \Omega(\mathbf{x})}_{J(\mathbf{x})}$$

$$J(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b - \lambda \|\mathbf{x}\|^2$$

$$\frac{\partial J}{\partial \mathbf{x}} = \mathbf{w} - 2\lambda \mathbf{x} \stackrel{!}{=} 0$$

$$\mathbf{x}^* = \frac{\mathbf{w}}{2\lambda}$$



$$\textcircled{1} \quad x_1 > x_2$$

$$J(\mathbf{x}) = x_1 - \lambda \|\mathbf{x}\|^2$$

$$\frac{\partial J}{\partial \mathbf{x}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} - 2\lambda \mathbf{x} \stackrel{!}{=} 0$$

$$\mathbf{x}^* = \begin{pmatrix} 1 \\ 0 \end{pmatrix} / 2\lambda$$

$$\textcircled{2} \quad x_2 > x_1$$

$$\mathbf{x}^* = \begin{pmatrix} 0 \\ 1 \end{pmatrix} / 2\lambda$$

Activation Maximization: Probability View

Assume the model produces a log-probability for class ω_c :

$$f(\mathbf{x}) = \log p(\omega_c | \mathbf{x})$$

The input \mathbf{x}^* that maximizes this function can be interpreted as the point where the classifier is the most *sure* about class ω_c .

Choose the regularizer $-\Omega(\mathbf{x}) = \log p(\mathbf{x})$, i.e. favor points that are *likely*.

The optimization problem becomes:

$$\begin{aligned}\mathbf{x}^* &= \arg \max_{\mathbf{x}} \log p(\omega_c | \mathbf{x}) + \log p(\mathbf{x}) \\ &= \arg \max_{\mathbf{x}} \log p(\mathbf{x} | \omega_c)\end{aligned}$$

where \mathbf{x}^* can now be interpreted as the most *typical* input for class ω_c .

Activation Maximization with Different Regularizers

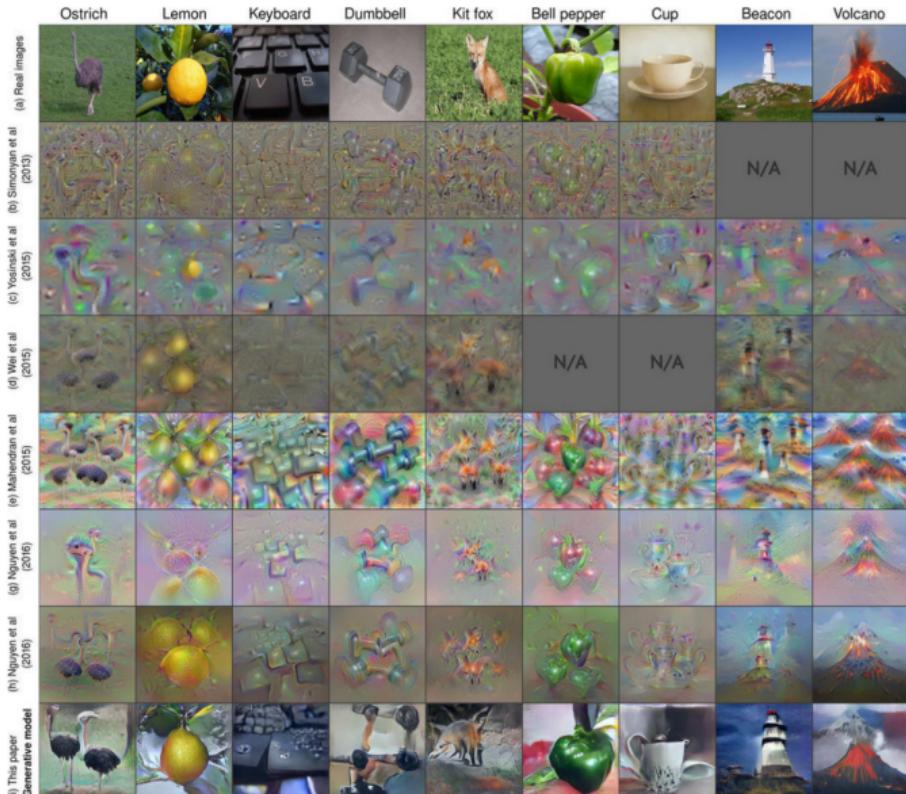


Image source: Nguyen et al. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks (2016)

Part 2

Attribution

Attribution of a Prediction to Input Features

input

x_1

x_2

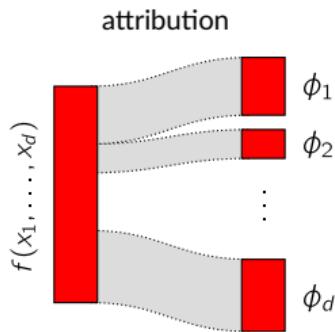
:

x_d

ML
blackbox

prediction

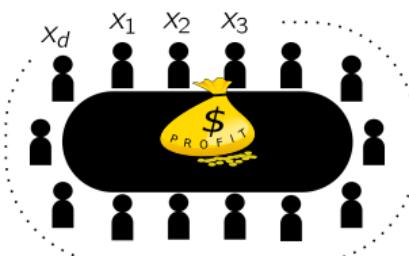
$f(x_1, \dots, x_d)$



1. The data point $\mathbf{x} \in \mathbb{R}^d$ is fed to the ML model and we get a prediction $f(\mathbf{x}) \in \mathbb{R}$.
2. We explain the prediction by identifying the additive contribution of each input feature.
3. Important property of attribution: **conservation** ($\sum_{i=1}^d \phi_i = f(\mathbf{x})$).

Attribution: Shapley Values [5, 6, 3]

- ▶ Framework originally proposed in the context of game theory (Shapley 1951) for assigning payoffs in a cooperative game, and recently applied to ML models.
- ▶ Each input variable is viewed as a player, and the function output as the profit realized by the cooperating players.



The Shapley values ϕ_1, \dots, ϕ_d measuring the contribution of each feature are:

$$\phi_i = \sum_{S: i \notin S} \frac{|S|!(d-|S|-1)!}{d!} [f(\mathbf{x}_{S \cup \{i\}}) - f(\mathbf{x}_S)]$$

where $(\mathbf{x}_S)_S$ are all possible subsets of features contained in the input \mathbf{x} .

Attribution: Shapley Values

Recall: $\phi_i = \sum_{S: i \notin S} \underbrace{\frac{|S|!(d-|S|-1)!}{d!}}_{\alpha_S} \underbrace{[f(\mathbf{x}_{S \cup \{i\}}) - f(\mathbf{x}_S)]}_{\Delta_S}$

Worked-through example: Consider the function $f(\mathbf{x}) = x_1 \cdot (x_2 + x_3)$. Calculate the contribution of each feature to the prediction $f(\mathbf{1})$.

ϕ_1	S	α_S	Δ_S	ϕ_2	S	α_S	Δ_S	ϕ_3
	$\{\emptyset\}$	$1/3$	0		$\{\emptyset\}$	$1/1$	0	
	$\{2\}$	$1/6$	1		$\{1\}$	$1/6$	1	
	$\{3\}$	$1/6$	1		$\{3\}$	$1/6$	0	
	$\{2, 3\}$	$1/3$	2		$\{1, 3\}$	$1/3$	1	
			\downarrow					
			1					
							\downarrow	
							$1/2$	$1/2$

Attribution: Shapley Values

Problem:

- ▶ Shapley values require an exponentially large number of evaluations of the function f (\rightarrow practically infeasible).

Observation:

- ▶ It has been shown that the Shapley value can also be written as:

$$\phi_i = \frac{1}{d!} \sum_{S \in \mathcal{P}} [f(\mathbf{x}_{\text{preceq}(S,i)}) - f(\mathbf{x}_{\text{prec}(S,i)})]$$

where \mathcal{P} denotes the set of all possible permutations of input features, e.g. for $d = 4$, we have

$$\mathcal{P} = \{(1, 2, 3, 4), (1, 2, 4, 3), (1, 3, 2, 4), \dots\},$$

and where $\text{prec}(S, i)$ is the set of features that precede feature i in the order of S , and $\text{preceq}(S, i)$ is the same features plus feature i , e.g.

$$\text{prec}((3, 1, 2, 4), 2) = (3, 1)$$

$$\text{preceq}((3, 1, 2, 4), 2) = (3, 1, 2)$$

Attribution: Shapley Values

Shapley Value Sampling:

- ▶ Approximate the Shapley value by replacing the exact form:

$$\phi_i = \frac{1}{d!} \sum_{S \in \mathcal{P}} [f(\mathbf{x}_{\text{preceq}(S,i)}) - f(\mathbf{x}_{\text{prec}(S,i)})]$$

by the sampling approximation:

$$\phi_i = \frac{1}{T} \sum_{S \in \{S^{(1)}, \dots, S^{(T)}\}} [f(\mathbf{x}_{\text{preceq}(S,i)}) - f(\mathbf{x}_{\text{prec}(S,i)})]$$

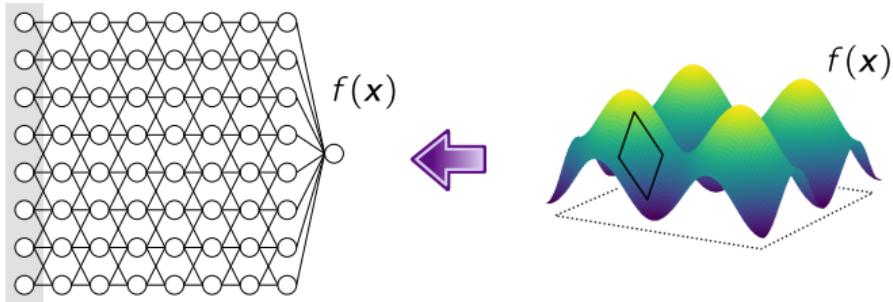
where $S^{(1)}, \dots, S^{(T)} \sim \text{uniform}(\mathcal{P})$.

- ▶ Converge to the true Shapley Value in the limit of $T \rightarrow \infty$.
- ▶ The larger T , the more precise the evaluation but also the more computationally expensive.

Example:

- ▶ Applying $T = 10$ on a function with $d = 25$ input dimensions, we need to evaluate 500 times the function f .
- ▶ No longer exponential, but still computationally expensive for large d .

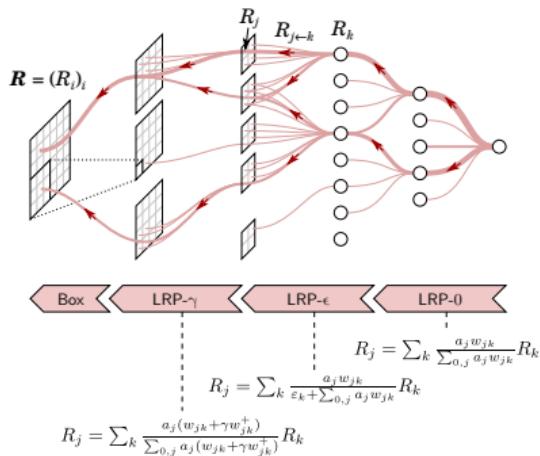
From Function-Based to Propagation-Based



Questions:

- ▶ Can we speedup the process of explanation by using the structure of the network (e.g. by running a special backpropagation pass)?

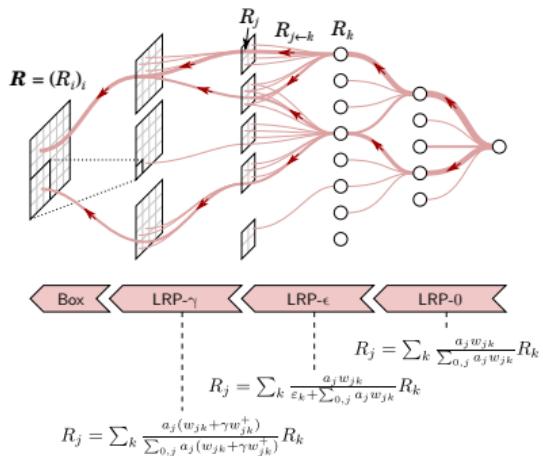
Layer-wise Relevance Propagation (LRP) [1, 4]



Ideas:

- ▶ Use the structure of the neural network to robustly compute relevance scores for the input features.
- ▶ Propagate the output of the network backwards by means of propagation rules.
- ▶ Propagation rules can be tuned for explanation quality. E.g. sensitive in top-layers, robust in lower layers.

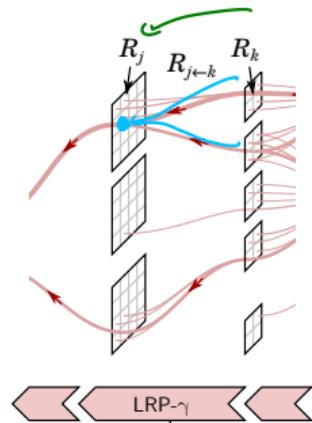
Layer-wise Relevance Propagation (LRP) [1, 4]



Some notation:

- ▶ j and k : neurons from successive layers
- ▶ w_{jk} : weight connecting neuron j to neuron k
- ▶ w_{0k} : bias for neuron k .
- ▶ $\sum_{0,j}$ sum over all input neurons j of neuron k and the bias.
- ▶ ReLU neuron:
 $a_k = \max(0, \sum_{0,j} a_j w_{jk})$.

Dissecting an LRP Propagation Rule



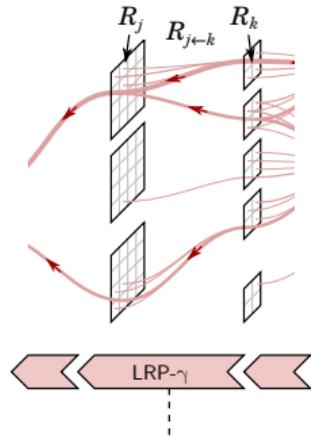
Example: LRP- γ [4]

$$R_j = \sum_k \frac{a_j(w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j(w_{jk} + \gamma w_{jk}^+)} R_k$$

- ▶ $a_j(w_{jk} + \gamma w_{jk}^+)$: Contribution of neuron a_j to the activation a_k .
- ▶ R_k ‘Relevance’ of neuron k available for redistribution.
- ▶ $\sum_{0,j} a_j(w_{jk} + \gamma w_{jk}^+)$ Normalization term that implements conservation.
- ▶ \sum_k : Pool all ‘relevance’ received by neuron j from the layer above.

$$R_j = \sum_k \frac{a_j(w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j(w_{jk} + \gamma w_{jk}^+)} R_k$$

Dissecting an LRP Propagation Rule (2nd view)



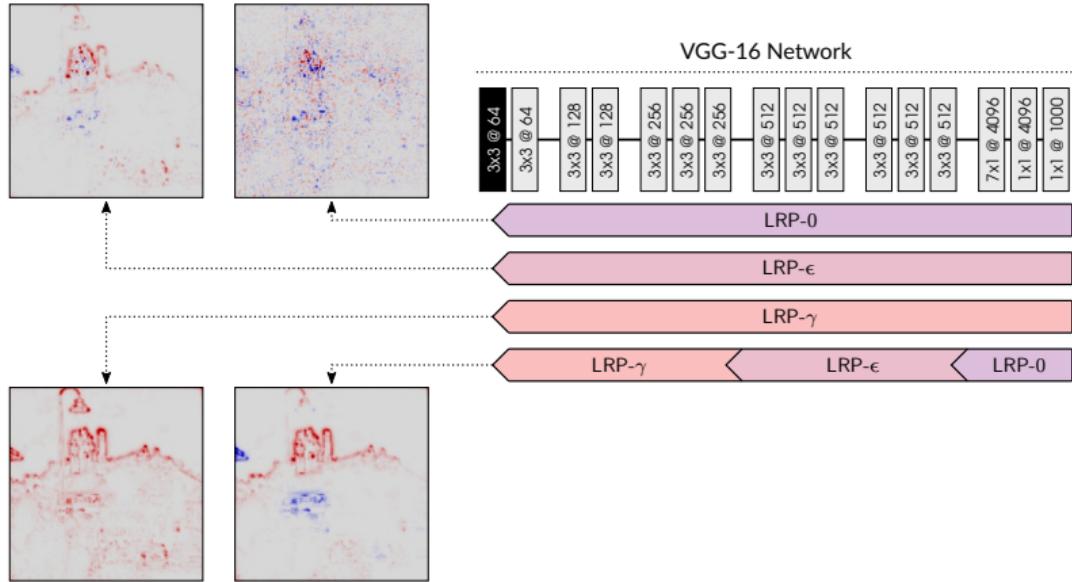
Example: LRP- γ [4]

$$R_j = a_j \cdot \left(\sum_k \frac{(w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)} R_k \right)$$

- ▶ a_j : Activation of neuron j .
- ▶ $(\sum_k \dots)$: Sensitivity of neural network output to a_j .

$$R_j = \sum_k \frac{a_j (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)} R_k$$

Effect of LRP Rules on Explanation



LRP rules must be chosen carefully to deliver best explanation quality. Generally, LRP rules are set different at each layer (cf. [4] for heuristics).

Layer-Wise Relevance Propagation

Advantages

- ▶ *Good explanation quality* on deep networks.
- ▶ *Fast* (in the order of a single forward/backward pass).
- ▶ *Flexible* (the multiple hyperparameters can be tuned to match the user needs).

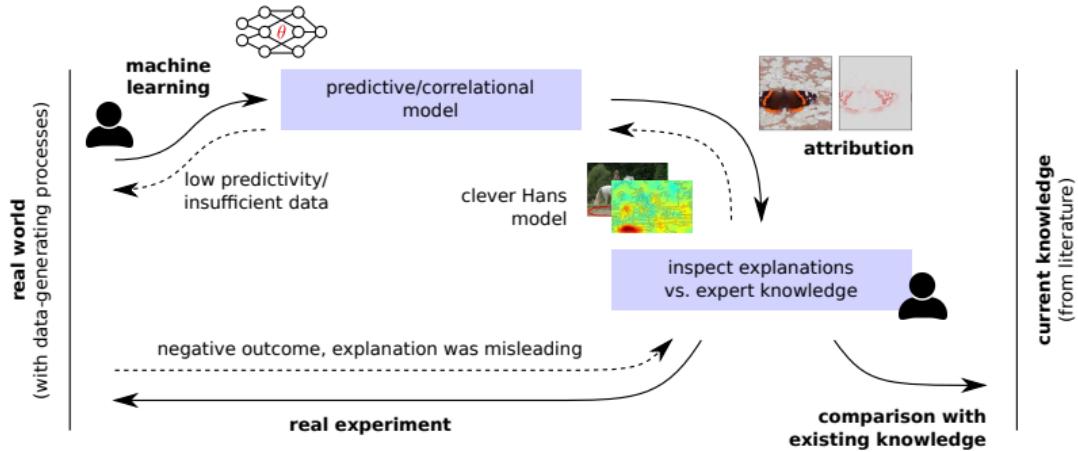
Limitations

- ▶ Need to choose the hyperparameters (\Rightarrow not unique).
- ▶ The LRP propagation strategy must be adapted to each new architecture.
- ▶ LRP assumes that the model has sufficiently disentangled the function to explain (i.e. it works for many neural networks but not for all models).

Part 3

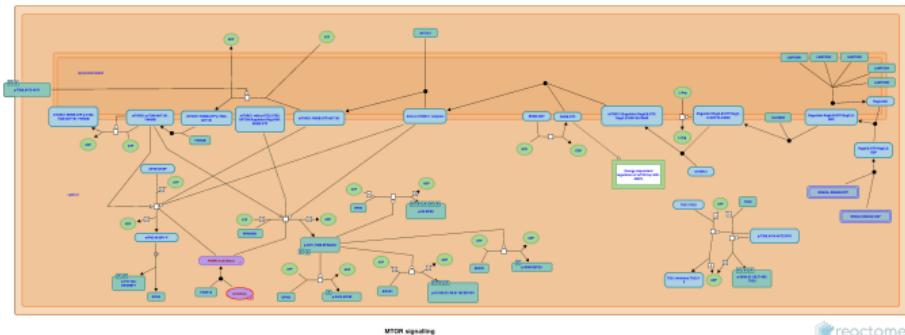
Finding Influential Proteins [2]

Explainable AI for Data Science



Example: Discovering Influential Proteins

Example: MTOR signaling network (from reactome.org)



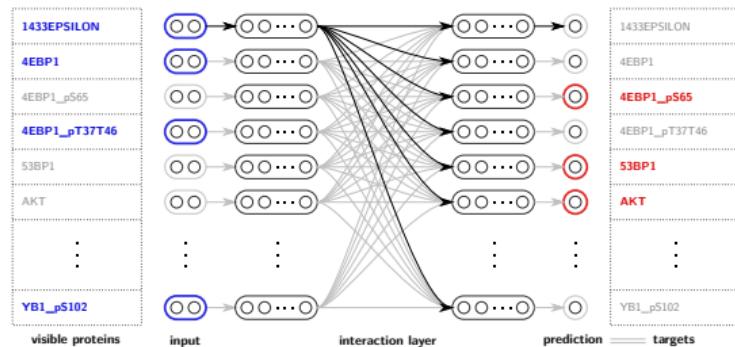
Question:

- ▶ Can we use ML/XAI to infer these networks (or aspects of them) directly from the data?

Finding Influential Proteins with ML/XAI

Step 1: From Data to ML

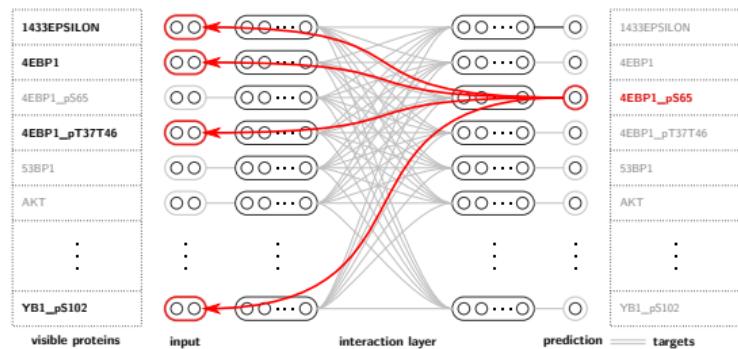
- ▶ Assemble a dataset.
- ▶ Build a ML model (neural network) that predicts proteins from other proteins with best possible accuracy.



Finding Influential Proteins with ML/XAI

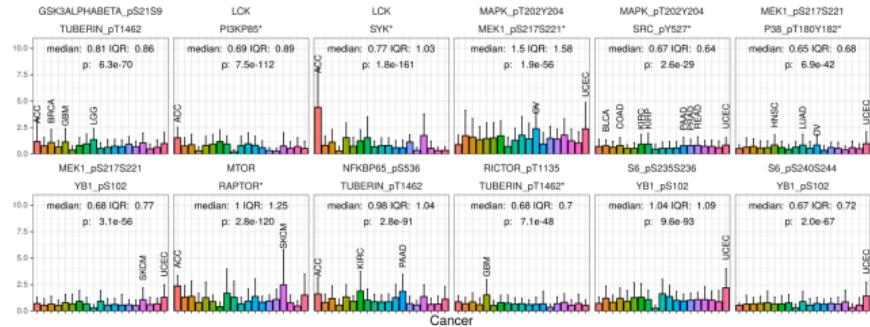
Step 2: From ML to XAI

- ▶ Apply Explainable AI (here the LRP attribution technique) to identify to what extent proteins contribute to the expression of other proteins.

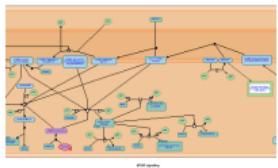


Finding Influential Proteins with ML/XAI

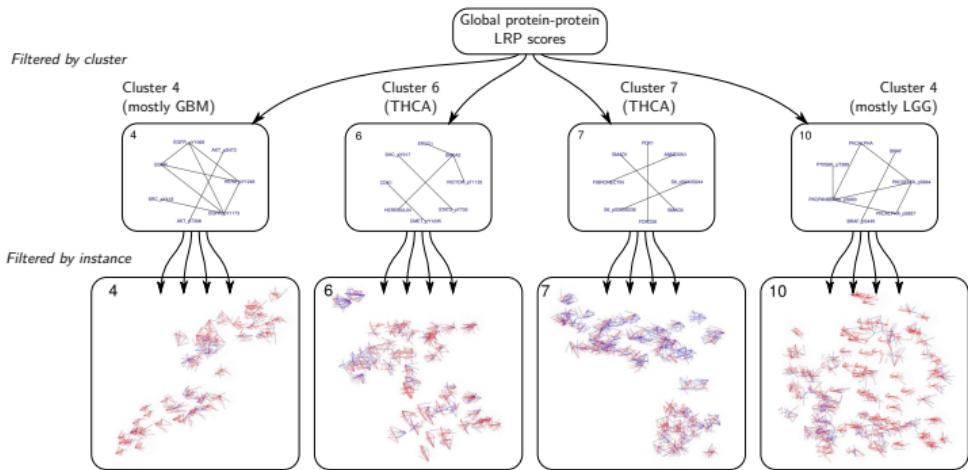
Excerpt of top- k protein influences via XAI (LRP attribution):



- ▶ Generally consistent with existing knowledge, e.g. highlights mTOR pathway; correlates with entries in the reactome knowledgebase (<https://reactome.org/>).
- ▶ Provides *cancer-specific* (or even *instance-specific*) view of protein influences.



Finding Influential Proteins with ML/XAI (Keyl et al. 2022)



- ▶ Can further enhance existing knowledge with *case-specific* protein influences.

Summary

Summary

- ▶ Explainable AI is an important addition to classical ML models (e.g. for validating a ML model or extracting knowledge from it).
- ▶ Many XAI methods have been developed, each of them, with their strengths and limitations:
 - ▶ *Activation maximization* can be used to understand what a ML model has learned, but is unsuitable for explaining an individual prediction $f(\mathbf{x})$.
 - ▶ *Shapley value sampling* has strong theoretical foundations, but is computationally demanding for high-dimensional input data.
 - ▶ *LRP* yields fast explanations of the ML model, but requires to carefully choose the propagation rules.

References

-  S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek.
On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation.
PLoS ONE, 10(7):e0130140, 07 2015.
-  P. Keyl, M. Bockmayr, D. Heim, G. Dernbach, G. Montavon, K.-R. Müller, and F. Klauschen.
Patient-level proteomic network prediction by explainable artificial intelligence.
npj Precision Oncology, 6(1), June 2022.
-  S. M. Lundberg and S. Lee.
A unified approach to interpreting model predictions.
In *NIPS*, pages 4765–4774, 2017.
-  G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller.
Layer-wise relevance propagation: An overview.
In *Explainable AI*, volume 11700 of *Lecture Notes in Computer Science*, pages 193–209. Springer, 2019.
-  L. S. Shapley.
A value for n-person games.
In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, Princeton, 1953.
-  E. Strumbelj and I. Kononenko.
An efficient explanation of individual classifications using game theory.
J. Mach. Learn. Res., 11:1–18, 2010.