REDS

An overview of machine learning interpretability

Authors :
Abdelraouf Keskes
Aurélia Dolo

Mentor :
Mme Laure Soulier

January 21, 2020

Contents

1 Introduction

Nowadays , the usage of AI and machine learning models is likely to become more ubiquitous in our daily life .It is also embracing the economy automation and data-driven decision-making. While these predictive systems can be quite accurate, they are usually treated as unbreakable black boxes that produce only numeric predictions with no accompanying explanations . Thus , one of the most aspiring , challenging and open hard tasks in Machine learning and especially in Deep learning is the 'Interpretability' also called the 'Explainability' . as Yann LeCun describes it in his podcast interview with Lex Friedman when he said that machine learning is the science of **"sloppiness"** which means that we generally start from ideas and intuitions .Then , we go through an experimental process pretending an empirical generalization without any solid proof , guaranty or **explainability**.

The latter raises some appealing questions around this area :

- Why we should trust our model ?
- are our usual metrics (accuracy, F1-score, ...) "reliable" and "sufficient" to evaluate ML Models in a real world problem involving budget and time investment ?
- do all applications have the same interpretability needs?

Consequently , if we look forward to improve ML models we need to formalize these notions and make them evidence-based .

In this report we propose an insightful overview over "Machine learning Interpretability" in order to answer partially these questions.We will also focus on and dive deeply into the most relevant techniques used in this fast-paced growing domain which could be one day a milestone by building a trustful and faithful bridge between academia and industry in AI. We will also propose a novel method tackling this task to go beyond what's available in the literature .

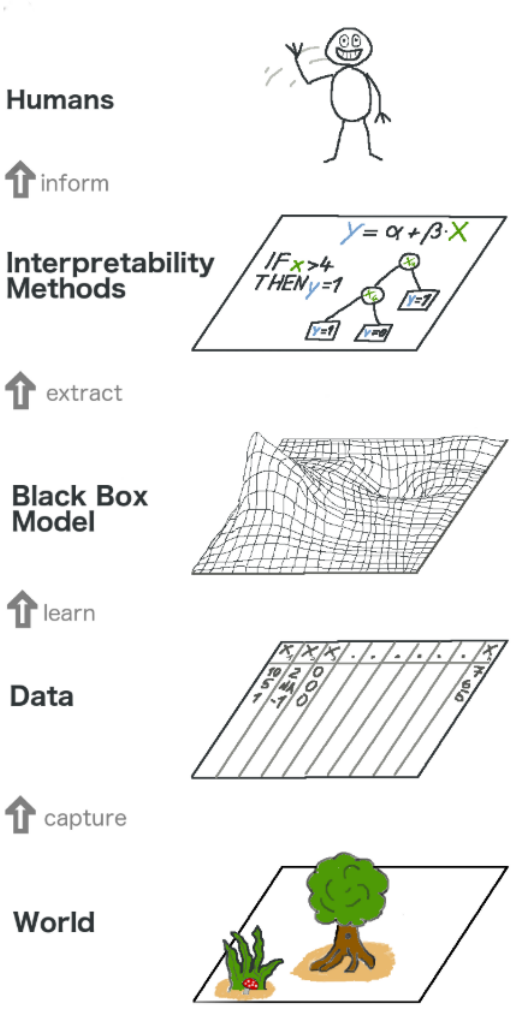
But before exposing state-of-the-art interpretability models and discussing them , we need to give some definitions and fundamentals around this domain .

2 Definition of ML interpretability

It is the ability to explain and present in an understandable terms to a human , whatever his background is , how the ML models are taking their decisions and why they predict a target instead of another with different input instances . In a concise way it should be defined as :

When we can stop asking why ?

here is a bigger picture of what ML interpretability / Explainability is :



Source : Cristoph Molnar Book (https://christophm.github.io/interpretable-ml-book/)

3 Do all systems and Models needs ML interpretability ?

The answer is No for both systems and models .

1. **Systems** :

   There are several applications such as Ad servers, postal code sorting and many other applications where an explanation is not required because of 2 main reasons :

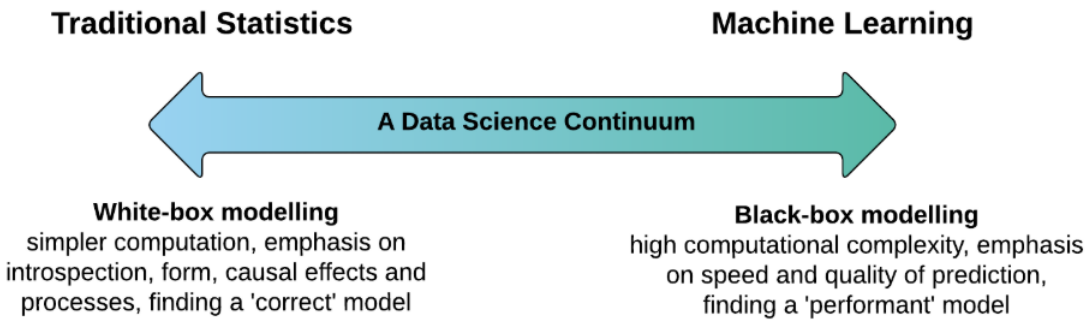
   - These subjects have been studied very well for years and leads to operational real world applications that are trustful and works very well , even though the system is not perfect .
   - There is no negative consequences if the system misses some outputs .

2. **Models** :

   We distinguish two types of machine learning models :

   - **White box Models** : which are generally weak models in terms of expressiveness and obviously performances . They are essentially Linear Models (such as linear regression) and Decision Trees which are easier to understand and interpret , but they have less complexity to capture some complex regularities behind the data .
   - **Black box Models** : We all know that stronger models which work better according to our metrics like Neural nets , GDBT , … are “Black-box” models where we don’t have any idea on how they’re taking decisions and predictions.Additionally , it is extremely hard to debug these models to improve results .

   From these two types , we could speculate that in real world problems , we try to find a trade off between ”performance” and ”interpretability” . The following figure picked up from Applied.AI illustrates the spectrum encountered by data scientists and companies working with these kind of technologies .



Source (Applied AI : Bayesian inference blog)

Note :

Note that even White Box Models could be easily transformed to Black Box models. For instance , decision trees could have a huge depth and it is almost impossible for a human to understand a graphical tree with hundreds of nodes .Linear models also could suffer from this issue with hundreds/thousands of weights weighting different features (which could be fixed by cropping and fetching only the relevant K weights where K is a hyper-parameter).

4 The main goals of ML interpretability ?

The following is a nutshell list of the main challenges and goals behind this growing domain :

- **Building low-risk application** :

  Deploying trustful application aiming to minimize drastically the error risk especially in senstive areas where the error is almost intolerable from a financial, operational and strategical perspective .

- **Debugging ML Models** :

  Most of data scientists , machine learning engineers or even researchers struggle a lot while discovering and trying to remedy models weaknesses and internal mechanisms and ML interpretability will natively help them debugging their models and have a deeper understanding of what's happening inside .

- **building accurate and trustful ML models** :

  if we could understand why the model tends to choose a class rather than another , we could enhance the model and achieve significant improvements .

  for example in image classification (Husky vs Wolf) , if we yield to the following explanation :

(a) Husky classified as wolf (b) Explanation

We could easily realize that our model is taking the snow a strong discriminant factor to predict a 'Husky' class (we build a snow detector rather than a husky/wolf classifier) , then we could remedy by importing new various training images for the wolf including snow , and vice-versa for the husky class .

- **AI fairness** :

  from a fair and ethical perspective , we could say that interpretability is a strong chunk in the current fairness/ethical questions .Because , If we could discover the patterns used by the model to predict we could devise and figure out the raised biases coming from the society and the data itself , in order to remediate the latter . Especially in human-centered sensitive domains such as healthcare , criminal justice and so forth accuracy is not no longer enough and people need explanations .
  Additionally , the problem of interpretability is exacerbated by the fact that there are different types of users of machine learning systems and these users may have different needs in different scenarios . The approach that works best for a regulator who wants to understand why a particular person was denied a loan may be different from the approach that works best for a CEO using a model to make a high-stakes decision.

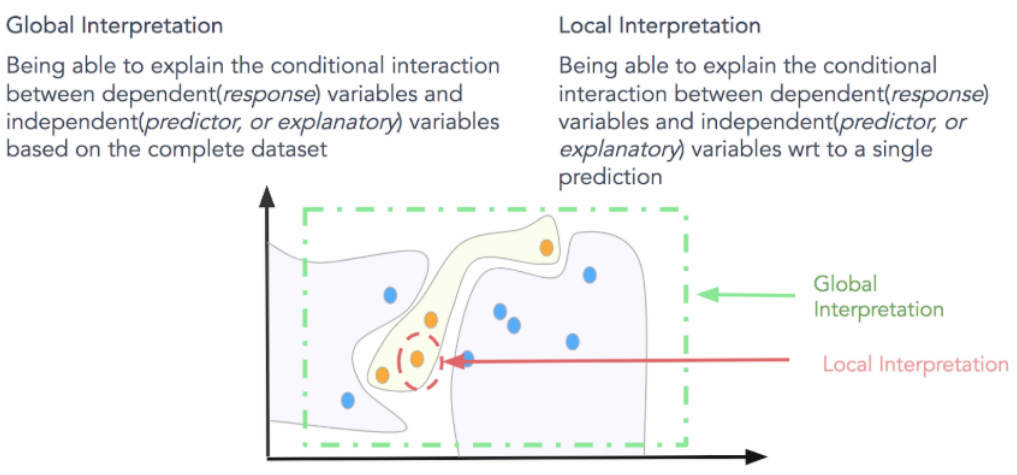
5 The scope ML interpretability ?

Basically , we have two different scopes :

- **Local** : when we answer the question : "Why did the model make a **certain prediction** for **a certain instance** ?" Local interpretations promote understanding small regions of our model , We assume that these small regions are more likely to be linear,monotonic,... which could be easily approximated .

  the following figures depicts the Local / Global scope :

Source : (H2O.AI)



Source : (Datascience.com)

- **Global** : when we answer the question : "How does the trained model make predictions ?" , global interpretations focus on the model as a whole chunk and its behavior, before deploying it "into the wild" .

6 Model-Agnostic and Model-Speci c Interpretability ?

Another important way to classify model interpretability techniques is whether they are model agnostic, meaning they can be applied to different types of machine learning algorithms and consider them as a black-box , or model specific, meaning techniques that are applicable only for a single type or class of algorithm. For instance, the LIME technique (which will explain later) is model agnostic and can be used to interpret nearly any set of machine learning inputs and machine learning predictions. On the other hand, the technique known as treeinterpreter is model specific and can be applied only to decision tree based models. Although model-agnostic interpretability techniques are convenient, and in some ways ideal, they often rely on surrogate (approximative) models or other approximations that can degrade the accuracy of the explanations they provide. Model specific interpretation techniques tend to use the predicting model to be interpreted **directly**, leading to potentially more accurate explanations.

7 Models

the following is a non-exhaustive list of the most relevant models in machine learning interpretability :

7.1 LIME

LIME which stands for ”Local Interpretable Model-Agnostic Explanations” is an algorithm that can explain the predictions of **any classifier or regressor** (Model-agnostic) in a faithful way, by approximating it **locally** with an **interpretable model** (such as linear models , decision trees , ...).Thus , it helps us to trust a given prediction , and maybe it could lead to take decisions based on the latter , For instance , in health care a trustful prediction on a patient leads to better decision from the doctor or in finance when we have to take strategical decisions .

7.1.1 How do we explain a prediction

by ”explaining a prediction ” , we mean giving a human understandable ”textual or a visual artifacts” which must provide a qualitative understanding of the relationship between the original instance’s components (Ex : sneeze , weight , headache , ...) and the prediction , the following figure depicts it more clearly

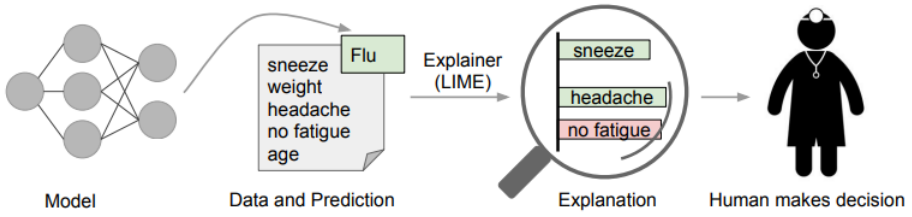


Figure 1: Explaining individual predictions. A model predicts that a patient has the flu, and LIME highlights the symptoms in the patient’s history that led to the prediction. Sneeze and headache are portrayed as contributing to the “flu” prediction, while “no fatigue” is evidence against it. With these, a doctor can make an informed decision about whether to trust the model’s prediction.

Source : LIME paper

7.1.2 Data representation

before delving deeply into the theory of the model we need to clarify the data representation , We will have two different data representation :

- **Learning representation** : It is our usual data features that we pass to our ML model which are generally complex and don’t provide some explainability , such as : word embeddings in NLP tasks .
- **Interpretable representation** : a new representation which must be understandable to humans, regardless of the actual features used by the ML model , for example

in text classification it is a binary vector indicating the presence or absence of a word . This the used representation for giving insights on interpretability .

7.1.3 Theory

Let's formalize LIME's theory mathematically :

- we denote one example from our dataset as $\boldsymbol{x} \in \boldsymbol{R}^{\boldsymbol{d}}$ and its interpretable representation as a binary vector denoted $\boldsymbol{x'} \in \boldsymbol{\{0,1\}}^{\boldsymbol{d'}}$.
- we define $\boldsymbol{f}$ as our ML model that we want to explain whose domain is $\boldsymbol{f : R^d \mapsto R}$ .For instance , in a classification problem $\boldsymbol{f(x)}$ is the probability (or a binary indicator) that $\boldsymbol{x}$ belongs to a certain class.
- we define an explanation as a model denoted $\boldsymbol{g} \in \boldsymbol{G}$, where $\boldsymbol{G}$ is a class of interpretable models such as : Linear Models , Decision Trees , ... We also denote the complexity of our model $\boldsymbol{g}$ as $\boldsymbol{\Omega(g)}$ (tree depth for Decision trees , non zeros weights for Linear models ...) . Note that the domain of $\boldsymbol{g}$ is $\boldsymbol{g : \{0,1\}^d \mapsto R}$.
- to define the locality around an instance $\boldsymbol{x}$, We further use $\boldsymbol{\pi_x(z)}$ as a proximity measure from an instance $\boldsymbol{z}$ to $\boldsymbol{x}$ .In the Lime paper they used a gaussian kernel (aka RBF) as a proximity measure between two points $\boldsymbol{\exp\left(-\frac{1}{\sigma^2} D(x,z)^2\right)}$, where $\boldsymbol{D}$ is a distance measure (e.g. cosine distance for text, $L2$ distance for images).
- Finally , let $\boldsymbol{\mathcal{L}(f,g,\pi_x)}$ be a measure of how unfaithful $\boldsymbol{g}$ is in approximating $\boldsymbol{f}$ in the locality defined by $\boldsymbol{\pi_x}$, as as a consequence , our goal now is to minimize this unfaithfulness to make g as close as possible to f .

With all these fundamental notations we could formalize our goal as :

$$\boldsymbol{\xi(x) = \underset{g \in G}{\arg\min} \left[\mathcal{L}(f,g,\pi_x) + \Omega(g)\right]}$$

By sampling uniformly around our represented instance $\boldsymbol{x'}$, we generate a small dataset where each generated instance is denoted $\boldsymbol{z' \in \{0,1\}^{d'}}$. Then, we recover the learning representation $\boldsymbol{z \in R^d}$ from our $\boldsymbol{z}$'.Afterwards,we could calculate $\boldsymbol{\pi_x(z)}$ and also obtain $\boldsymbol{f(z)}$ which is the label of the perturbed sampled point z according to our ML model . Note that the number of sampled instances $\boldsymbol{N}$ is also selected uniformly .

For the rest of the paper they decided to consider $\boldsymbol{g}$ as a Linear Model which means that $\boldsymbol{g(z') = w_g . z'}$, Therefore our $\boldsymbol{\mathcal{L}}$ becomes a weighted MSE loss as following :

$$\boldsymbol{\mathcal{L}(f,g,\pi_x) = \sum_{z,z'} \pi_x(z)(f(z) - g(z'))^2}$$

Generally , in the Interpretable representation we want to list TOP-K elements , for example in text classification we will not display the list of all vocabulary words with a weight of contribution for each vocabulary word , to make it human readable we will seek K most relevant words and display them . Therefore , before optimizing our Loss function we will need to select K features and for this we will use LASSO feature selection and then learning the weights W_g , this 2 learning steps will be denoted as K-LASSO .

The final algorithm is :

Algorithm 1 Sparse Linear Explanations using LIME

Require: Classifier f, Number of samples N
Require: Instance x, and its interpretable version x'
Require: Similarity kernel π_x, Length of explanation K
  $\mathcal{Z} \leftarrow \{\}$
  for $i \in \{1, 2, 3, ..., N\}$ **do**
    $z'_i \leftarrow sample\_around(x')$
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z_i), \pi_x(z_i) \rangle$
  end for
  $w \leftarrow$ K-Lasso$(\mathcal{Z}, K)$ ▷ with z'_i as features, $f(z)$ as target
  return w

7.1.4 Extension : SP-LIME

It is an extension of LIME to give a global interpretation .The idea is to select B representative and non redundant instances from X and apply a LIME on each of them to get the global behavior of the model . So , our main problem is to select these B examples, we will explain the approach point by point :

- given the interpretability representation of n instances We define W as the matrix that represents the local importance of the interpretable components for each instance whose shape is $n \times d$.

- we will use feature importance to describe the **global** importance of a component j in the interpretability space , it is defined as : $\boldsymbol{I_j = \sqrt{\sum_i W_{i,j}}}$

- we want to cover all features with no redundancy as the following example :

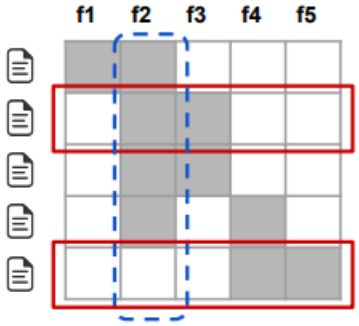


Figure 5: Toy example $\mathcal{W}$. Rows represent instances (documents) and columns represent features (words). Feature f2 (dotted blue) has the highest importance. Rows 2 and 5 (in red) would be selected by the pick procedure, covering all but feature f1.

We define the coverage as the set function C that, given the matrix W , the importance vector of size d' as I , and a set of instances as V , as the following :

$$\boldsymbol{C(V, W, I) = \sum_{j=1}^{d'} 1_{(\exists i \in V, W_{ij} > 0)} I_j}$$

and our goal as :

$$\boldsymbol{Pick(W, I) = \arg\max_{V, |V| \leq B} C(V, W, I)}$$

- The problem is NP-Hard . Let $C(V \cup \{i\}, W, I) - C(V, W, I)$ be the marginal coverage gain of adding an instance i , a greedy algorithm that iteratively adds the instance with the highest marginal coverage gain will ensure an approximation guarantee to the optimum , this pick is called **submodular pick** hence the name of the algorithm .

the SP-LIME algorithm is therefore as following :

Algorithm 2 Submodular pick (SP) algorithm

Require: Instances X, Budget B
  for all $x_i \in X$ **do**
    $\mathcal{W}_i \leftarrow$ **explain**(x_i, x_i') ▷ Using Algorithm 1
  end for
  for $j \in \{1 \dots d'\}$ **do**
    $I_j \leftarrow \sqrt{\sum_{i=1}^{n} |\mathcal{W}_{ij}|}$ ▷ Compute feature importances
  end for
  $V \leftarrow \{\}$
  while $|V| < B$ **do** ▷ Greedy optimization of Eq (4)
    $V \leftarrow V \cup \text{argmax}_i \, c(V \cup \{i\}, \mathcal{W}, I)$
  end while
  return V

Note that they use **sum pooling** to estimate the feature importance

7.1.5 Advantages

1. it works for tabular data, text and images.
2. gives us a good idea of how reliable the interpretable model is in explaining the black box predictions in the neighborhood of the data instance of interest

7.1.6 Limitations

1. the kernel function is based on heuristics with no theoretical proofs , thus it should be really devised before selecting it for a certain probelem .
2. Sampling problem : ignoring the correlation between features could lead unrealistic data points .

7.2 Layer-wise Relevance Propagation (LRP) for Neural Networks

7.2.1 Idea

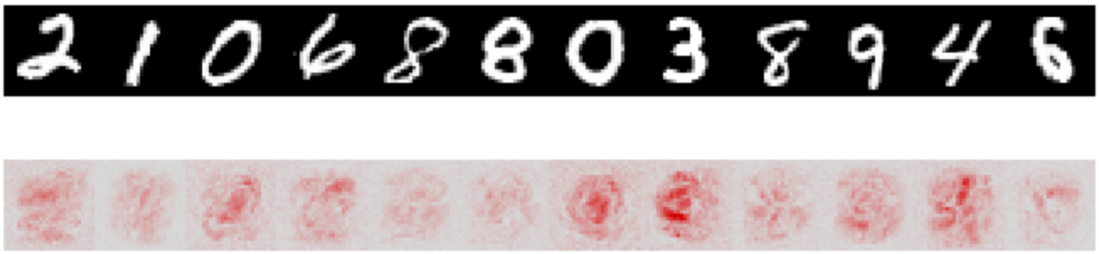
this method is dedicated to explain neural networks which are the most striking "black box" models example . In the following we consider neural networks consisting of layers of neurons. The output $\boldsymbol{xj}$ of a neuron $\boldsymbol{j}$ is a non-linear activation function $\boldsymbol{g}$ as given by :

$$\boldsymbol{x_j = g(\sum_i w_{ij} * x_i + b)}$$

Given an image $\boldsymbol{x}$ and a classifier $\boldsymbol{f}$ the aim of layer-wise relevance propagation is to assign each pixel $\boldsymbol{p}$ of $\boldsymbol{x}$ a pixel-wise relevance score $R_p^{(1)}$ (1 is for input layer **"1"** representing pixels for an image) such that :

$$\boldsymbol{f(x) \approx \sum_p R_p^{(1)}}$$

Pixels p with $\boldsymbol{R_p^{(1)} < 0}$ contain evidence against the presence of a class, while , $\boldsymbol{R_p^{(1)} > 0}$ is considered as evidence for the presence of a class. These pixel-wise relevance scores can be visualized as an image called **heatmap** as the following figure shows it :



Source : (heatmapping.org/tutorial/)

Relevant pixels are highlighted in red . we could see that each digit was captured and drawn with relevant pixels , including some background pixels . For instance , we observe two red horizontal bars next to the digit "3", highlighting the fact that if those pixels would be different, the digit 3 would likely turn into a "8"

Another figure which illustrates more what LRP does to explain predictions

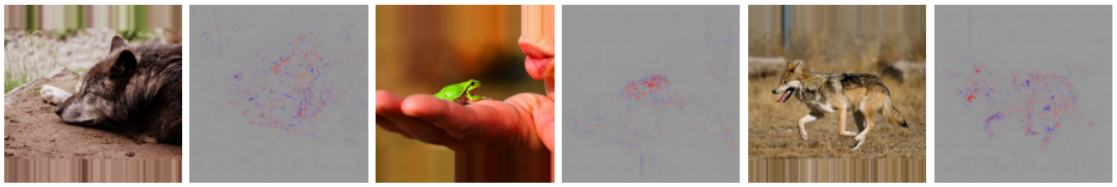


Fig. 1. Pixel-wise decompositions for classes wolf, frog and wolf using a neural network pretrained for the 1000 classes of the ILSVRC challenge.

Source : (LRP with Local Renormalization Layers paper)

7.2.2 Theory

We could see relevant pixels capturing the head of the wolf and the frog . We assume that we know the relevance $\boldsymbol{R_j^{(l+1)}}$ of a neuron $\boldsymbol{j}$ at network layer $\boldsymbol{(l+1)}$ for the classification

decision $\boldsymbol{f(x)}$, then we like to decompose this relevance into messages $\boldsymbol{R_{i\leftarrow j}^{(l,l+1)}}$ sent to those neurons $\boldsymbol{i}$ at the layer $\boldsymbol{l}$ which provide inputs to neuron $\boldsymbol{j}$ such that the following equation holds :

$$\boldsymbol{R_j^{(l+1)} = \sum_{i\in(l)} R_{i\leftarrow j}^{(l,l+1)}}$$

We can then define the relevance of a neuron $\boldsymbol{i}$ at layer $\boldsymbol{l}$ by summing all messages from neurons at layer $\boldsymbol{l+1}$ as :

$$\boldsymbol{R_i^{(l)} = \sum_{j\in(l+1)} R_{i\leftarrow j}^{(l,l+1)}}$$

the previous 2 equations define the propagation of relevance from layer $l+1$ to layer l

Note that :

- The relevance of the output neuron k at layer L is $\boldsymbol{R_k^{(L)} = f(x)}$
- The pixel-wise score are the resulting relevance of the input neuron d as $\boldsymbol{R_d^{(1)}}$

the following figure gives an intuitive visualization of what these formulas and explanations are talking about :

Pixel-Wise Explanations of Non-Linear Whole Image Classifier Decisions

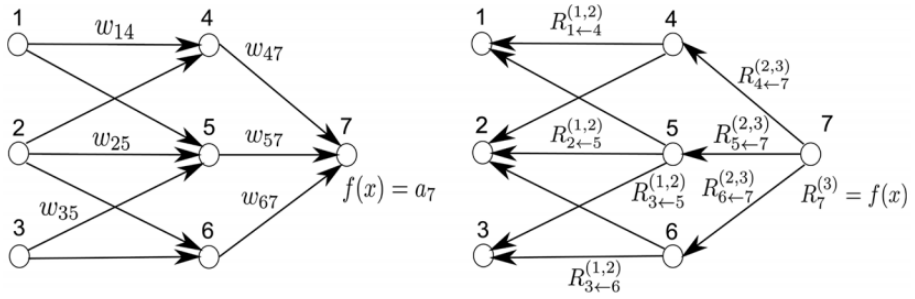


Fig 2. Left: A neural network-shaped classifier during prediction time. w_{ij} are connection weights. a_i is the activation of neuron i. Right: The neural network-shaped classifier during layer-wise relevance computation time. $R_i^{(l)}$ is the relevance of neuron i which is to be computed. In order to facilitate the computation of $R_i^{(l)}$ we introduce messages $R_{i\leftarrow j}^{(l,l+1)}$. $R_{i\leftarrow j}^{(l,l+1)}$ are messages which need to be computed such that the layer-wise relevance in Eq (2) is conserved. The messages are sent from a neuron i to its input neurons j via the connections used for classification, e.g. 2 is an input neuron for neurons 4, 5, 6. Neuron 3 is an input neuron for 5, 6. Neurons 4, 5, 6 are the input for neuron 7.

doi:10.1371/journal.pone.0130140.g002

Source : (LRP PLOS ONE journal)

to compute the messages $\boldsymbol{R_{i\leftarrow j}^{(l,l+1)}}$ the they established two formulas :

1. The first formula called **ϵ-rule** which is given by :

$$\boldsymbol{R_{i\leftarrow j}^{(l,l+1)} = \frac{z_{ij}}{z_j + \epsilon * sign(z_j)} R_j^{(l+1)}}$$

with $\boldsymbol{z_{ij} = w_{ij} * xi}$, $\boldsymbol{z_j = \sum_{k:w_{kj}\neq 0} z_{kj}}$ and $\boldsymbol{\epsilon}$ is a small number for stabilization and to avoid degeneration when z_j is very close to zero .

2. The second formula called β-rule is given by :

$$\boldsymbol{R_{i\leftarrow j}^{(l,l+1)} = ((1+\beta) * \frac{z_{ij}^+}{z_j^+} - \beta * \frac{z_{ij}^-}{z_i^-}) * R_j^{(l+1)}}$$

where the positive and negative weighted activations are treated separately. The variable β controls how much inhibition/contribution is incorporated in the relevance redistribution

Obviously , we have the following structure : $\boldsymbol{R_{i\leftarrow j}^{(l,l+1)} = v_{ij} * R_j^{(l+1)}}$ with $\boldsymbol{\sum_i v_{ij} = 1}$

The meaningfulness of the resulting pixel-wise decomposition for the input layer comes from the fact that the terms v_{ij} are derived from the weighted activations $wij * xi$ of the input neurons. Note that layer-wise relevance propagation does not use gradients in contrast to backpropagation during the training phase .

Other rules :

Name	Formula	Usage	DTD
LRP-0 [7]	$R_j = \sum_k \frac{a_j w_{jk}}{\sum_{0,j} a_j w_{jk}} R_k$	upper layers	✓
LRP-ϵ [7]	$R_j = \sum_k \frac{a_j w_{jk}}{\epsilon + \sum_{0,j} a_j w_{jk}} R_k$	middle layers	✓
LRP-γ	$R_j = \sum_k \frac{a_j (w_{jk} + \gamma w_{jk}^+)}{\sum_{0,j} a_j (w_{jk} + \gamma w_{jk}^+)} R_k$	lower layers	✓
LRP-$\alpha\beta$ [7]	$R_j = \sum_k \left(\alpha \frac{(a_j w_{jk})^+}{\sum_{0,j} (a_j w_{jk})^+} - \beta \frac{(a_j w_{jk})^-}{\sum_{0,j} (a_j w_{jk})^-}\right) R_k$	lower layers	×⋆
flat [30]	$R_j = \sum_k \frac{1}{\sum_j 1} R_k$	lower layers	×
w^2-rule [36]	$R_i = \sum_j \frac{w_{ij}^2}{\sum_i w_{ij}^2} R_j$	first layer ($\mathbb{R}^d$)	✓
$z^{\mathcal{B}}$-rule [36]	$R_i = \sum_j \frac{x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-}{\sum_i x_i w_{ij} - l_i w_{ij}^+ - h_i w_{ij}^-} R_j$	first layer (pixels)	✓

(⋆ DTD interpretation only for the case $\alpha = 1, \beta = 0$.)

7.2.3 Advantages

1. Propagation rules can be implemented efficiently and modularly in most modern neural network frameworks .

2. LRP rules can be set in a way that high explanation quality is obtained even for complex models

3. LRP is effective when applied to various data types (images, text, audio, video, EEG/fMRI signals) and neural architectures (ConvNets, LSTMs) .

7.2.4 Limitations

1. Model-specific : LRP is dedicated only for Deep neural nets

7.3 DeepLIFT (Deep Learning Important FeaTures)

7.3.1 Idea

It is a method which try to fix several problems of related work such as:

1. Perturbation-based approaches and gradient-based approaches failure to model saturation . Note that in this report we did not tackle yet (gradient-based methods) maybe in further versions we will discuss them with details , The saturation problem implies that perturbations sampling will have no effect and the gradient will be 0 .The following figure illustrates it :

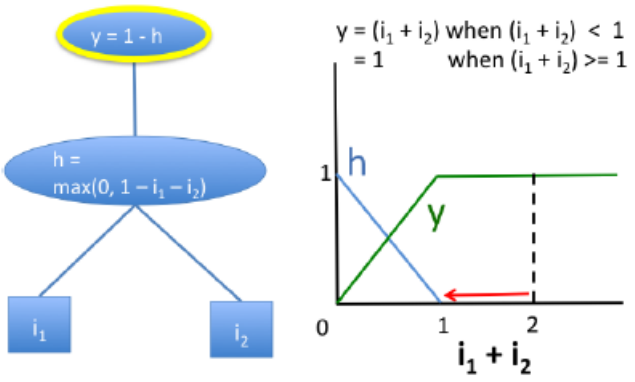


Figure 1. **Perturbation-based approaches and gradient-based approaches fail to model saturation.** Illustrated is a simple network exhibiting saturation in the signal from its inputs. At the point where $i_1 = 1$ and $i_2 = 1$, perturbing either i_1 or i_2 to 0 will not produce a change in the output. Note that the gradient of the output w.r.t the inputs is also zero when $i_1 + i_2 > 1$.

Source : (DeepLIFT paper)

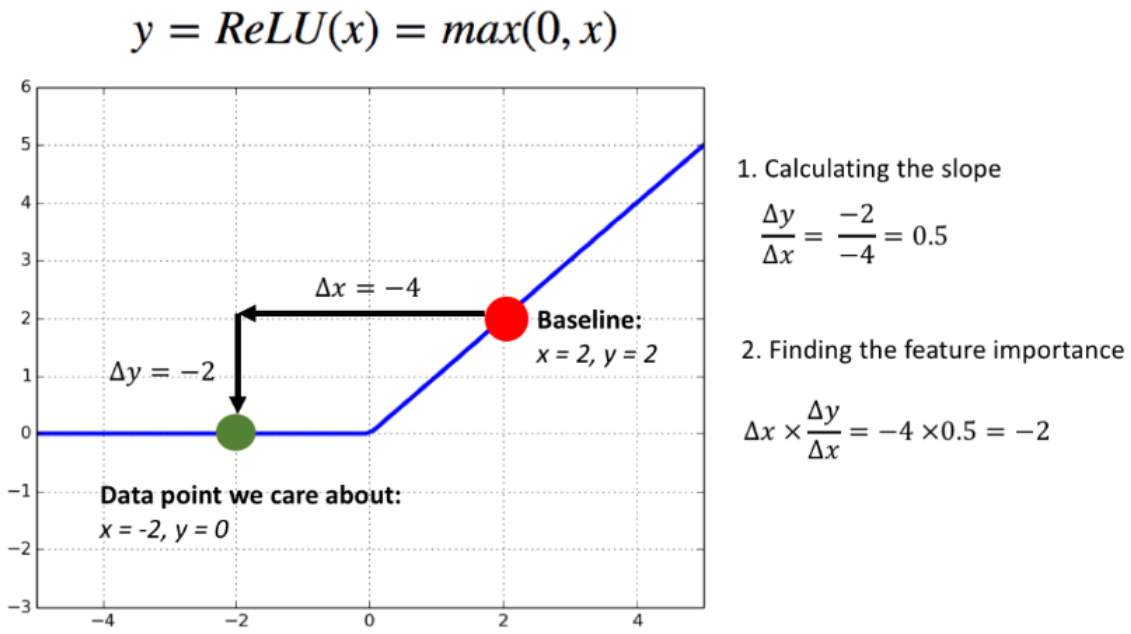
2. We have also the problem of the ReLU function (an ubiquitous activation function in nowadays Neural nets architectures which zero outs negative signals) .The latter lead to an issue while backpropagating gradient to extract feature importance , thus we could not tease out some features that impacts in our explanations . Discontinuities in the gradients can also cause undesirable artifacts .

DeepLIFT recognizes that what we care about is not the gradient, which describes how y changes as x changes at the point x, **but the slope, which describes how y changes as x differs from the reference(baseline).**

$$x_i \times \frac{\partial y}{\partial x_i} \Rightarrow x_i \times \frac{\Delta y}{\Delta x_i}$$

7.3.2 Theory

the following figure will give you a more insightful and clear vision about it :



Source : (Gabriel Tseng Medium Blogs)

$ReLU(x = 2) = 2$, and $ReLU(x = -2) = 0$, so my input feature $x = -2$ has changed the output of my model by 2 compared to the baseline. This change in the output of my model has to be attributed to the change in x, since its the only input feature to this model, but the gradient of $ReLU(x)$ at the point x = -2 is 0! This tells me the contribution of x to the output is 0, which is obviously not informative at all and in a certain way a contradiction.

Therefore as you notice it we are going to use "the slope" to backpropagate (instead of gradients) for explainability (not for learning) which is defined as :

$$slope = \frac{y - y_{ref}}{x - x_{ref}} = \frac{\Delta y}{\Delta x}$$

they call this slope **" a multiplier"** denoted by **m** .We obviously and intuitively could apply the chain rule and the backpropagation for both gradients (an infinitesimal contribution by definition of the derivative) and slopes called multipliers in their paper .Therefore, you can now backpropagate along these multipliers to find the slope of the input with respect to the output of the model, allowing feature importances to be easily defined.

$$FeatureImportance(i) = x_i \times \frac{\partial y}{\partial x_i} \Rightarrow (x_i - x_i^{ref}) \times \frac{\Delta y}{\Delta x_i}$$

Note that here we use the 'partial slope' of y, similar to calculating the partial derivative.

7.3.3 Choosing a reference/baseline ?

'

- **input** : The 'reference' input represents some default or 'neutral' input that is chosen according to what is appropriate for the problem at hand , therefore it is very critical and it depends on domain knowledge , we could ask ourselves "what am I interested in measuring differences against?" which could help to define this input reference .We have also some practical examples that work very well such as : MNIST (references are all zeros) , Genomics (references are the frequencies of ACGT) and so forth , we denote them as $\boldsymbol{x_1^0, x_2^0, ..., x_n^0}$.
- **intermediate neuron** : activation of neuron when network is supplied a reference input .
- **output** : we can calculate the reference activation y^0 of the output as :

$$\boldsymbol{y^0 = f(x_1^0, x_2^0, ..., x_n^0)}$$

In other words , references for all neurons can be found by choosing a reference input and propagating activations through the net.

7.3.4 Advantages

1. it remediates to the saturation problem.
2. it only requires one backward pass of the model to calculate feature importance values.

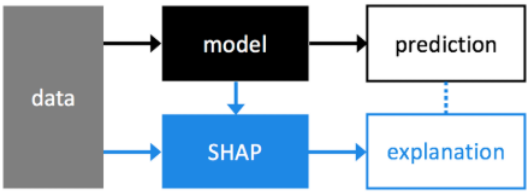
7.3.5 Limitations

1. picking the reference/baseline example is extremely difficult and required domain expertise except for trivial cases such as images, ...
2. We need to redefine all deep learning frameworks to include these slope backpropation property
3. it is suitable for Neural Nets and all gradient-based models which could be replaced by the slope to a reference , thus it does not treat the model as a black box .

7.4 SHAP (SHapley Additive exPlanations)

7.4.1 Idea

SHAP is a unified approach to explain the output of any machine learning model. It connects cooperative game theory with local explanations, uniting several previous methods and representing the only possible consistent and locally accurate additive feature attribution method based on expectations .



The idea of SHAP is that for a particular feature, the prediction of a model for a specific data point is compared when it can see the feature, and when it can't .Then, the magnitude of this difference tells us how important that feature is to the model's prediction.

7.4.2 Shapley Values : from cooperative game theory to machine learning

The Shapley value, coined by Shapley (1953) , is a method for assigning payouts to players depending on their contribution to the total payout. Players cooperate in a coalition and receive a certain profit from this cooperation. In a machine learning context we introduce :

- The "game" is the prediction task for a single instance of the dataset .
- The "gain" is the actual prediction for this instance minus the average prediction for all instances .

- ”the players” are features values of the corresponding instance that collaborate to receive the gain . A player can be an individual feature value, e.g. for tabular data. A player can also be a group of feature values. For example to explain an image, pixels can be grouped to super pixels and the prediction distributed among them .

Let’s take the following apartment example to illustrate these key points :

The “game” is the prediction task for this instance [$50m^2$, 2nd Floor , park-nearby , cat-banned] , these 4 features worked together to achieve the prediction of €300,000. Our goal is to explain the difference between the actual prediction (€300,000) and the average prediction (€310,000): a difference of -€10,000 .

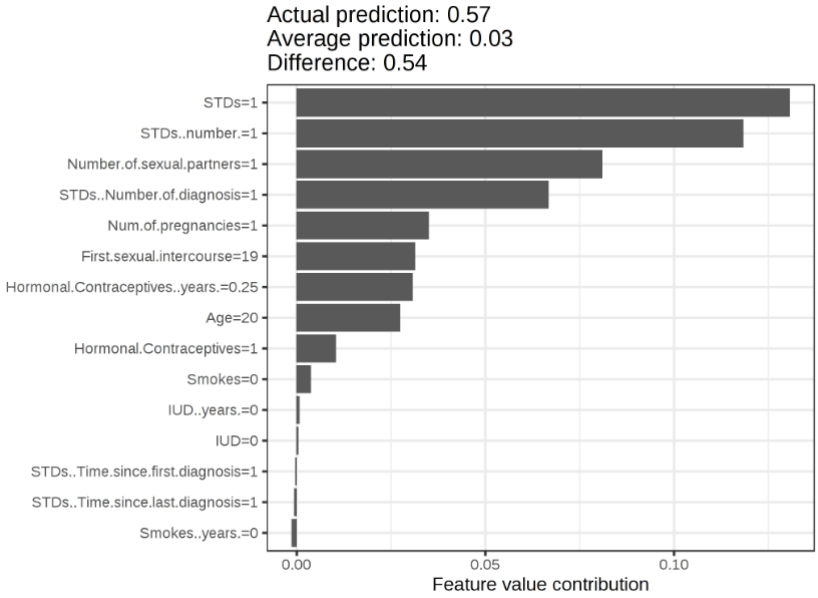
The answer could be: The park-nearby contributed €30,000; size-50 contributed €10,000; floor-2nd contributed €0; cat-banned contributed -€50,000. The contributions add up to -€10,000, the final prediction minus the average predicted apartment price.

the Shapley value : is the average marginal contribution of a feature value across all possible coalitions. For instance , The following list shows all coalitions of feature values that are needed to determine the Shapley value for ”cat-banned” feature value :

1. No feature values
2. park-nearby
3. size-50
4. floor-2nd
5. park-nearby size-50
6. park-nearby floor-2nd
7. size-50 floor-2nd
8. park-nearby size-50 floor-2nd

For each of these coalitions we compute the predicted apartment price with and without the feature value cat-banned and take the difference to get the marginal contribution. The

Shapley value is the (weighted) average of marginal contributions the following example depicts the Shapley values to analyze the predictions of a random forest model predicting cervical cancer :



Source (christoph molnar book : chapter "Shapley values")

Note that The Shapley value **is the average contribution of a feature value to the prediction in different coalitions**. The Shapley value is **NOT** the difference in prediction when we would remove the feature from the model

The Shapley value of a feature value is its contribution to the payout, weighted and summed over all possible feature value combinations:

$$\phi_j(val) = \sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_j\}} \frac{|S|!\,(p-|S|-1)!}{p!} \left(val\left(S \cup \{x_j\}\right) - val(S)\right)$$

where S is a subset of the features used in the model, x is the vector of feature values of the instance to be explained and p the number of features.$val_x(S)$ is the prediction for feature values in set S that are **marginalized over features that are not included in set S** .

to clarify and to not get confused by the many uses of the word "value":

- The feature value is the numerical or categorical value of a feature and instance .
- the Shapley value is the feature contribution to the prediction

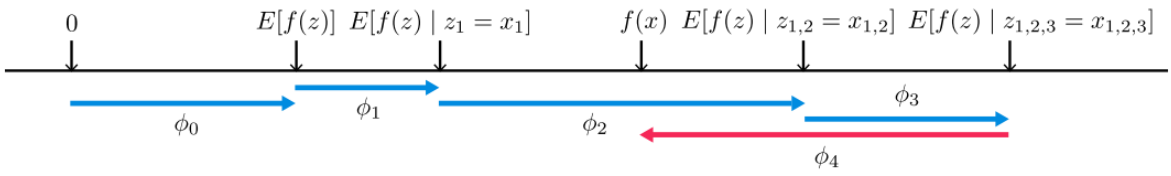
- the value function is the payout function for coalitions of players (feature values)

Note that Shapley values have very nice properties which we will only note without delving deeply into them : **Efficiency**, **Symmetry**, **Dummy** and **Additivity**, which together can be considered a definition of a fair payout.

As the number of possible coalitions increases exponentially with the number of features , computing the exact shapley value solution is almost unfeasable . Thus , we approximate it with **Monte-Carlo Sampling** .

7.4.3 Theory

SHAP consider the explanation as an **"additive features explanations/contributions model"** , The following figure shows what mean exactly by (SHapley Additive exPlanation)



Source (SHAP paper)

Features Values explain how to get from the base value $E(f(x))$ that would be predicted if we did not know any features to the current output $f(x)$.

Mathematically it is a linear model (function) as the following :

$$g(z') = \phi_0 + \sum_{j=0}^{N} \phi_j * z'_j$$

where g is the explanation model , $z' \in \{0,1\}^M$ is the coalition vector indicating presence/absence of a feature value, M is the maximum coalition size and $\phi_j \in \mathbb{R}$ is the feature attribution for a feature j, Many previous methods discussed previously share this additive contribution paradigm , especially LIME which inspired a lot SHAP framework .

A surprising attribute of the class of additive feature attribution methods is the presence of **a single unique solution** in this class with three desirable properties (the following are **SHAP paper properties** and not **Shapley values properties** cited previously) :

1. **Local accuracy** : the explanation model g should match at least the learning model f on the instance x which we want to explain (we denote the coalition vector of x

as x' where all features values are present) . Mathematically , it is formalized as the following :

$$f(x) = g(x') = \phi_0 + \sum_{j=0}^{N} \phi_j * x'_j$$

2. **Missingness** : Missingness says that a missing feature gets an attribution of zero. Note that x'_j refers to the coalitions, where a value of 0 represents the absence of a feature value , Mathematically it is :

$$x'_j = 0 \Rightarrow \phi_j = 0$$

3. **Consistency** : The consistency property says that if a model changes so that the marginal contribution of a feature value increases or stays the same (regardless of other features), the Shapley value also increases or stays the same .Mathematically it gives the following : Let $\boldsymbol{f_x(z') = f(h(z'))}$, $z' \setminus i$ denotes $z'_i = 0$

   For any two models f and f' if we have :

$$f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i)$$

   Therefore for all $z' in \{0,1\}^M$:

$$\phi_j(f', x) \geq \phi_j(f, x)$$

In practice, the computation of SHAP values is challenging that is why in SHAP paper the proposed approximations to estimate it , we have :

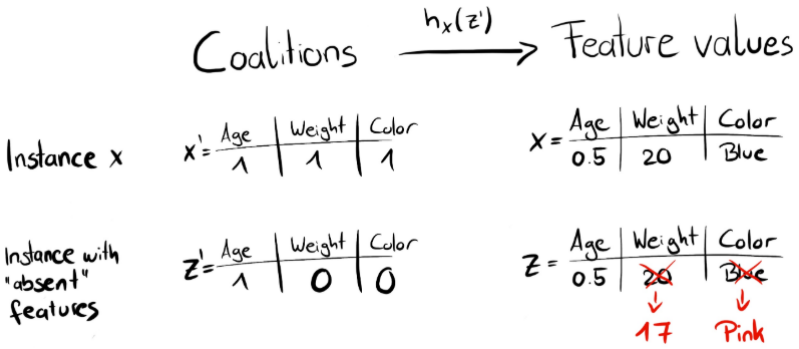
- **KernelSHAP(Lime + Shapley values)** :

  we will need :

  - as in LIME a mapping function from the interpretation space to learning space defined as $\boldsymbol{h(z') = z}$ where

$$\boldsymbol{h : \{0,1\}^M \mapsto R^P}$$

    The function h maps 1's **to the corresponding value from the instance x that we want to explain** and it maps 0's **to the values of another instance that we sample from the data**.



- The big difference to LIME is the weighting of the instances in the regression model. LIME weights the instances according to how close they are to the original instance .The more 0's in the coalition vector, the smaller the weight in LIME . SHAP weights the sampled instances according to the weight the coalition would get in the Shapley value estimation. Small coalitions (few 1's) and large coalitions (i.e. many 1's) get the largest weights .The intuition behind it is: We learn most about individual features if we can study their effects in isolation.In order to achieve Shapley weighting, they proposed the SHAP kernel as the following :

$$\pi_x(z') = \frac{M-1}{\binom{M}{|z'|} * |z'| * (M - |z'|)}$$

Here, M is the maximum coalition size (number of features) , and $|z'|$ the number of present features in instance z' , applying this kernel LIME gives Shapley values , far from its heuristical approximation about the kernels .

KernelSHAP is a model-agnostic approximation , its algorithm consists of 5 steps :

1. Sample K coalitions $z'_k \in \{0,1\}^M$
2. map them to the learning space and get their predictions :

$$f(h(z'_k)) = f(z_k)$$

3. Compute the weight for each z_k with the SHAP kernel
4. fit the weighted linear model by optimizing this Loss :

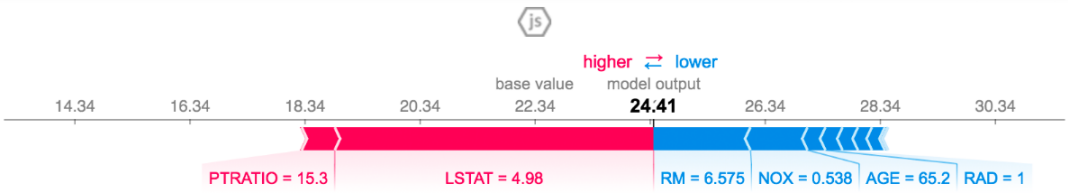
$$\mathcal{L}(f, g, \pi_x) = \sum_{z'} \pi_x(z')[f(h(z')) - g(z')]^2$$

5. Return Shapley values ϕ_j the coefficients from the linear model

- **DeepSHAP(DeepLIFT + Shapley values)** : While Kernel SHAP can be used on any model, including deep models . DeepSHAP is a Model-Specific Approximation dedicated to Deep neural nets .

- **TreeSHAP(Decision Tree + Shapley values)** : a variant of SHAP for tree-based machine learning models such as decision trees, random forests and gradient boosted trees. TreeSHAP is fast, computes **exact** Shapley values, and correctly estimates the Shapley values when features are dependent. In comparison, KernelSHAP is expensive to compute and only **approximates** the actual Shapley values.

  the following is a force plot of a TreeSHAP explainer example :



Source (SHAP github repo)

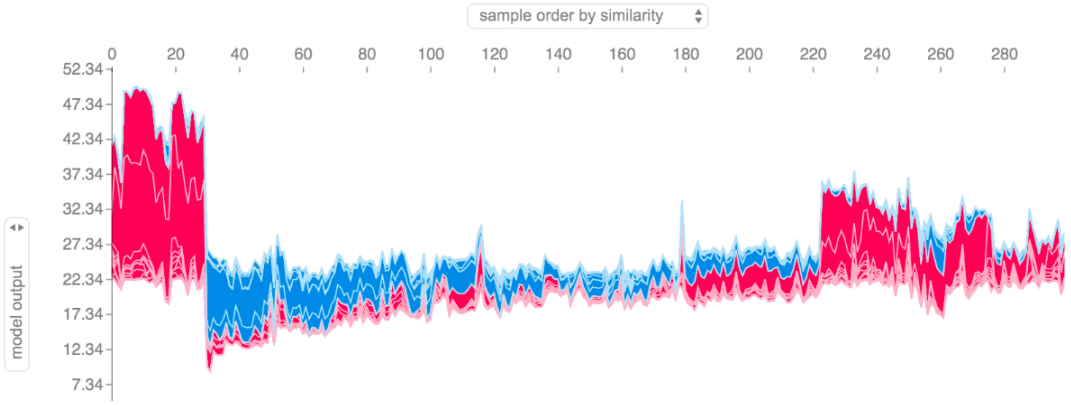
The above explanation shows features each contributing to push the model output from the base value (the average model output over the training dataset we passed) to the model output. Features pushing the prediction higher are shown in red, those pushing the prediction lower are in blue

7.4.4 Global Explanation with SHAP

Shapley values can be combined into global explanations. If we run SHAP for every instance, we get a matrix of Shapley values. This matrix has one row per data instance and one column per feature. We can interpret the entire model by analyzing the Shapley values in this matrix

1. **Clustering plot** :

   If we take many explanations such as the one shown above for the tree explainer , rotate them 90 degrees, and then stack them horizontally, we can see explanations for an entire dataset according to their clustering similarity using Shapley values as the following :

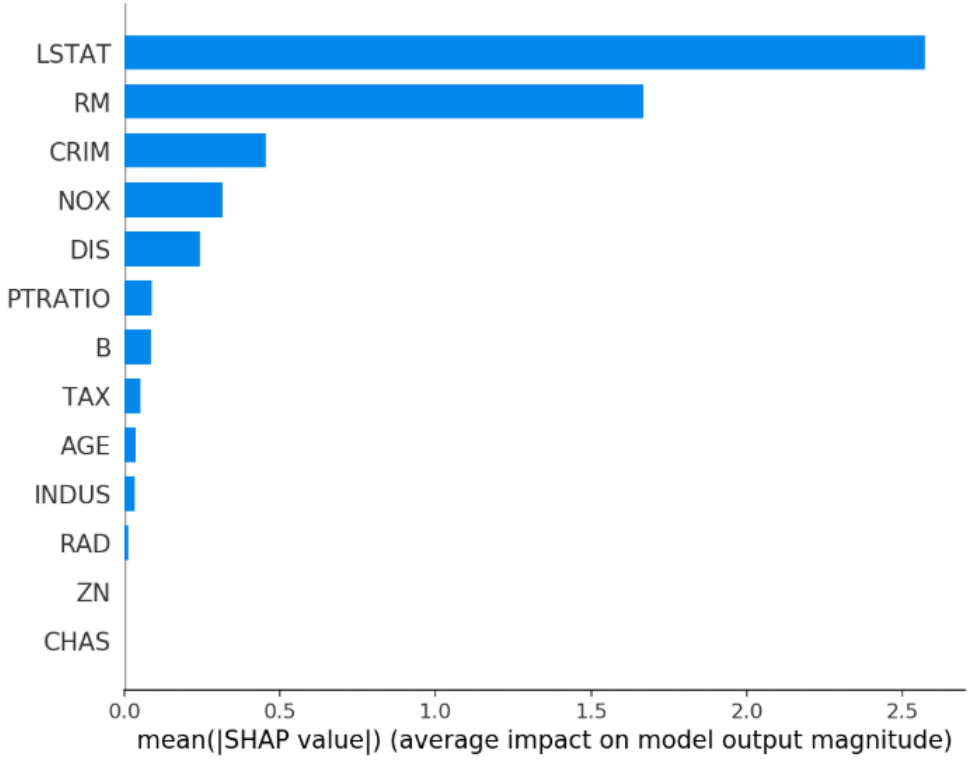


Source (SHAP github repo)

2. **Feature Importance** :

we average the absolute feature contribution (shapley value) over all instances as :

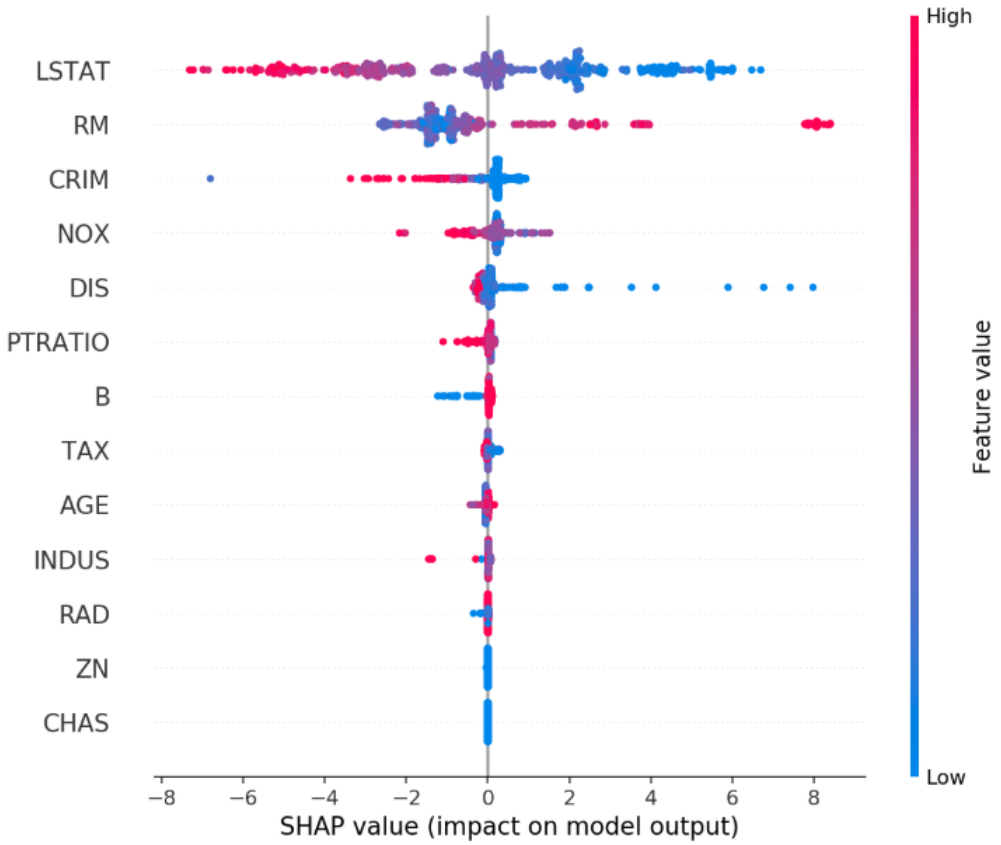
$$I_j = \frac{1}{n} \sum_{i=0}^{n} |\phi_j^{(i)}|$$



Source (SHAP github repo)

3. **Summary plot** :

   To get an overview of which features are most important for a model we can plot the SHAP values of every feature for every sample. The plot below sorts features by the sum of SHAP value magnitudes over all samples, and uses SHAP values to show the distribution of the impacts each feature has on the model output



Source (SHAP github repo)

7.4.5 Advantages

1. it has a solid theoretical foundations coming from game theory and ensures fairly distribution between features .

2. global interpretations are consistent with local interpretations

3. fast implementation for TreeSHAP

7.4.6 Limitations

1. The Shapley value is the wrong explanation method if you seek sparse explanations

2. KernelSHAP is slow and requires a lot of computing time

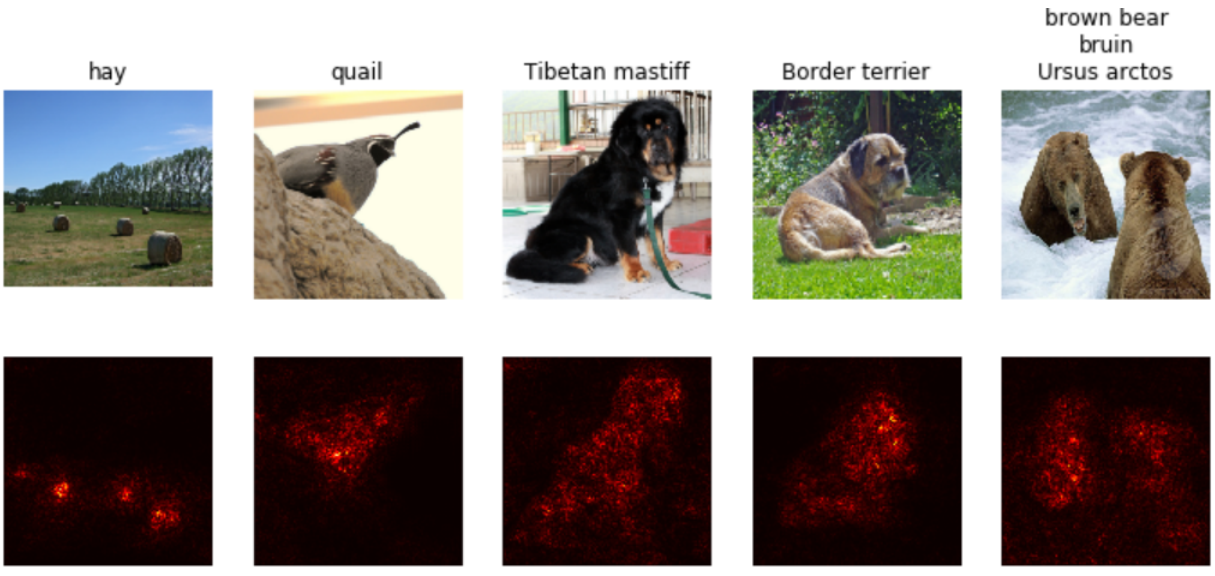
3. you need access to the data if you want to calculate the Shapley value for a new data instance because you need the data to replace parts of the instance . This can only be avoided if you can create data instances that look like real data instances but are not actual instances from the training data.

4. by sampling we could generate unrealistic data instances .

8 Example Based Methods

8.1 Saliency maps

We approximate the neural network f around an instance(image) x for a class i with a linear model $\tilde{y}_i = f_i(x) \approx w_i x + b_i$ with $w_i = \frac{\partial \tilde{y}_i}{\partial x}$ (importance = gradient to the predicted class) Note that we take the maximum value over the channels (3 for RGB) to produce our final 2D saliency map :

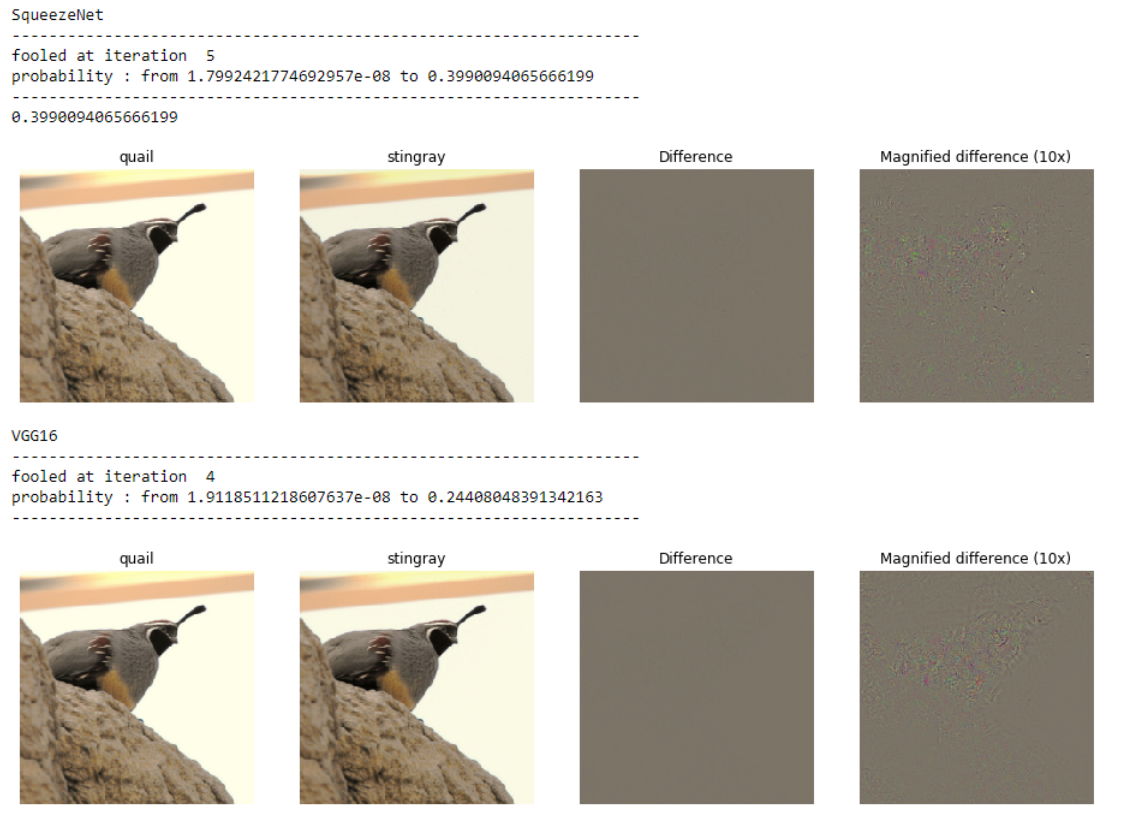


Explaining VGG16 neural network using saliency maps on ImageNet

8.2 Adversarial Attacks

the idea is about adding some noise to an image (which keep it almost the same for a human eye perception) leads to a drastic modification and the neural network will miss it the correct class at all . to achieve this we will perform an ascent gradient to our initial image x aiming to maximize the prediction $\tilde{y}_j$ of the class j that we want our model to predict :

$$x \leftarrow x + \eta * \frac{g}{||g||_2} \text{ with } g = \frac{\partial \tilde{y}_j}{\partial x}$$



Fooling SqueezeNet/VGG16 using Adverserial attacks

8.3 Deep Visualization

The principle is quite simple, considering an initiale image x (for example simply a random noise), we want to modify this image so as to maximize the score $\tilde{y}_j$ of the class j .To improve the quality of the results, it is proposed to use a certain number of regularization techniques.First, add the $L2$ standard of the image as regularization when calculating the gradient. In addition, we will regularly apply implicit regularizations to the image we are

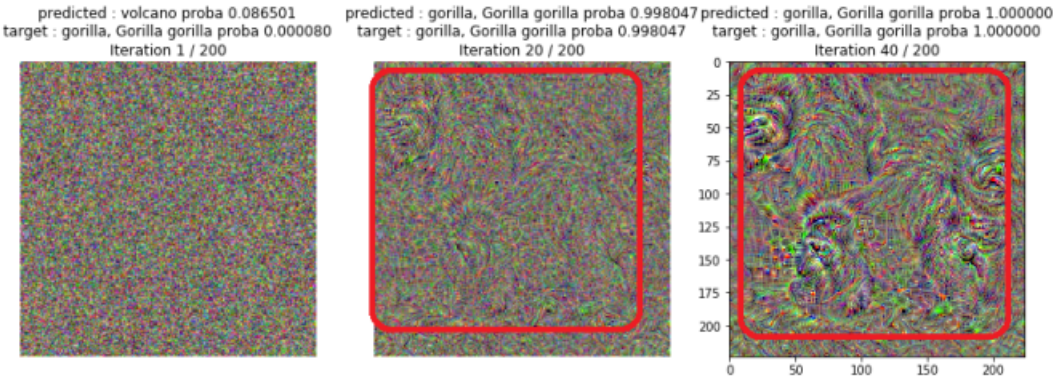
working on, by translations and blurs. our Loss that we want to maximize is :

$$L = \tilde{y_j} - \lambda ||x||_2$$

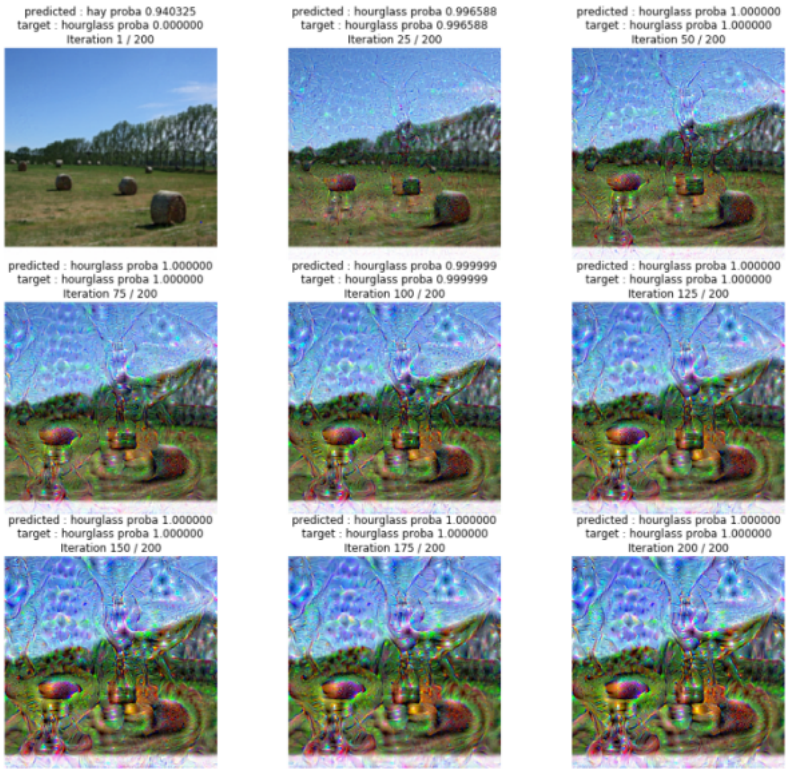
our Updating rule is :

$$x \leftarrow x + \eta * \nabla_x L$$

Literally , it is extracting learned patterns for a certain class and put them in an image whether to visualize them (simple image) or to fool our model (real image)



Visualize gorilla class for SqueezeNet model starting from a random image



Fooling SqueezeNet from a real image using learned patterns

9 Summary

Models	LIME	LRP	DeepLift	Attacks	SHAP
Model-agnostic	✓				✓KernelSHAP
Model-specific		✓	✓	✓	✓DeepSHAP
local approximation	✓	✓	✓	✓	✓KernelSHAP ✓TreeSHAP
global approximation	✓SP-LIME				✓Feature Importance and others
gradient-based				✓	
backprop-based		✓	✓	✓	✓DeepSHAP
perturbation-based	✓				✓KernelSHAP

Bibliography

[1] C. Molnar, *Interpretable Machine Learning*. 2019. `https://christophm.github.io/interpretable-ml-book/`.

[2] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, "Not just a black box: Learning important features through propagating activation differences," *ArXiv*, vol. abs/1605.01713, 2016.

[3] M. Tulio Ribeiro, S. Singh, and C. Guestrin, "" Why Should I Trust You?": Explaining the Predictions of Any Classifier," *arXiv preprint arXiv:1602.04938*, 2016.

[4] A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, and W. Samek, "Layer-Wise Relevance Propagation for Neural Networks with Local Renormalization Layers," in *Artificial Neural Networks and Machine Learning – ICANN 2016* (A. E. Villa, P. Masulli, and A. J. Pons Rivero, eds.), (Cham), pp. 63–71, Springer International Publishing, 2016.

[5] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, vol. 73, pp. 1 – 15, 2018.

[6] 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*, vol. 10, no. 7, pp. 1–46, 2015.

[7] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3145–3153, JMLR. org, 2017.

[8] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

[9] K. Fernandes, J. S. Cardoso, and J. Fernandes, "Transfer learning with partial observability applied to cervical cancer screening," in *Iberian conference on pattern recognition and image analysis*, pp. 243–250, Springer, 2017.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[11] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[12] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.