Interpretability of Deep Learning Models

Report of the M2 Internship

Yiwei ZHANG
ICFP, ENS Paris

Supervisor: Dr. Maria Rodriguez Martinez
Head of the system biology group of IBM Research Lab Zurich

June 17, 2020

Contents

1 Abstract

Machine learning(ML) has become a set of powerful techniques for processing large datasets in various scientific or real-world contexts. Simple ML models such as linear regression are transparent while of limited use. To fit complex non-linear data, Deep Neural Networks(DNNs) has been employed for learning due to their strong predictability. These deep learning(DL) models are mostly treated as black boxes because of difficulties in explaining the mechanisms for predictions. This is unsatisfactory not only in terms of human curiosity, but also for applications in risk-sensible fields like financial investment, medical diagnosis, automatic driving, etc where decision errors can be fatal. This makes it necessary to investigate how predictions are made by DNNs. Explanations for predictions can therefore broaden the employment of this powerful technique. In this report, the state-of-the-art of research in methods for explanation is reviewed. Local interpretability methods including LIME, Anchors, Gradient-based methods and SHAP, along with global interpretability methods from linear regression and logistic regression to decision tree, have been introduced. Some code experiments, including one with LIME and the other to explain the DeepBind model using LIME and Anchors, have also been described.

2 Introduction

With 7.8 billion people living on earth, every day, considerable amounts of data are born in various fields. Making full use of data thus becomes challenging due to limited capacity of human memory and abilities in inference. Recording data on paper once helped releasing the pressure for human memory, but data processing by hand still lacked accuracy and efficiency. Thanks to the birth of computer, which outperforms human in computing, tools for automatic data processing could be developed. Taking advantage of the rapid growth of computing performance, one expects that larger amount of data can depict models more precisely and drive more useful predictions. As extensions to traditional linear regression, *machine learning(ML)* techniques can perform more complex fittings including not only linear and nonlinear regressions but also classifications, which provide basis for recognition techniques. Amongst the ML techniques, *deep learning(DL)*, those using *deep neural networks(DNNs)* turn out to possess extreme powerful predictability. Use of DL has been taken in online translation, human face recognition, automatic customer service, etc. However, strong predictability calls for more precise model building, which usually indicates higher complexity and more difficulties in understanding the way it works. Whether to trust a result given by DL then becomes a vital issue in risk sensible fields where people prefer avoiding mistakes. It would be pitiful if such a strong technique cannot bring revolutionary promotions in medical diagnosis and financial investments in a convincing way, but human expert knowledge should indeed find its role in the DL-assisted decision-making processes. Interpretability of DL thus becomes crucial for this purpose, for that it provides explanations to the predictions, so that human experts can evaluate their reliabilities, especially when it comes to those counterintuitive results. In this report, several methods of interpreting deep learning results are to be reviewed. Notice should be taken that the two terms 'explain' and 'interpret' will be used interchangeably.

2.1 Machine Learning

2.1.1 What is machine learning?

The aim of ML is to improve the accuracy and efficiency of predictions made by computers based on data. A set of methods have been developed for this job, with a wide spectrum of complexities. Models should be adapted to the their applications, i.e. there is no universal ML method. Using this technique, one is able to tell, for instance, the price of a house based on the patterns in the past sales learned by a computer and the input information of the house to be evaluated. This is an example of *supervised learning* where the dataset to be learned contains the outcomes. That is, the computer is fed

with a set of exercises with given corresponding answers. In this report, only supervised learning is concerned.

2.1.2 How does one describe supervised learning?

Given a training dataset $\mathcal{T} = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}, i = 1, 2, ..., n\}$ to feed the computer, the aim is to allow it to tell a $y \in \mathcal{Y}$ from a new $x \in \mathcal{X}$. Here $\mathcal{T}$ is the set of *training data*. As the input part, $\mathcal{X}$ can be images, sounds, text, biomolecular sequences, etc, depending on the playground for ML. The outcome part $\mathcal{Y}$ determines the type of learning tasks. If $\mathcal{Y} = \mathbf{R}$, the learning task is called *regression*, providing *numerical* outcomes. If $\mathcal{Y}$ is finite or infinite countable, the learning task is called *classification*, providing *categorical* outcomes.

2.1.3 Why is it challenging?

After being trained, the computer is expected to learn the model as a function $f : \mathcal{X} \to \mathcal{Y}$. Its form may not be analytically explicit, but will be expected to give a certain output $y = f(x)$ once having an input $x \in \mathcal{X}$. However, getting the exact function f can be tricky since only a finite number of training data are used for training and the data themselves are noisy. One should also expect that the outcome of a new input $x \in \mathcal{X}$ contains noisy contributions: $y = f(x, \epsilon)$ where ϵ stands for noisy contributions. Noises introduce errors, but lack of noises brings deviations. When building a model, both underfitting and overfitting should be reduced if impossible to avoid. Neither errors nor deviations are ideal for precise predictions, but how should one assess the reliability of a prediction without prior knowledge about the learning mechanisms? This is where interpretability comes into play. Like in a paper addressing an unsolved scientific problem, like the Riemann's conjecture, it is far from sufficient to provide only the final keys. In order to convince the referees reading and assessing the paper, one should provide enough reasoning details in an understandable way for experts. Here in ML, especially for DL, the 'talented predictors', people also want their predictions to be explained. This will help the development of more reliable ML models and algorithms.

2.2 Interpretability

2.2.1 What is interpretability?

Unfortunately, because it is difficult to uniformly measure the interpretability of a learning model, no rigorous definition of interpretability can be formulated. Still, one can keep in mind that it evaluates how easy it is for human to understand the origin of a prediction made by machine. A straightforward corollary is that higher interpretability implies more user-friendly explanations for predictions.

2.2.2 Why bother?

For obvious reasons, one should be cautious about how predictions are made by machine when it is not clear how reliably the learning model works. But will interpretability no longer be an issue if one is sure(or almost sure depending on the field) about the learning performance? Not always! Indeed, when the machine learning model itself has no impact, interpretability seems redundant. Nevertheless, in other cases, the prediction itself does not fully answer the question, or even simply does not answer. For example, when one wants to know the sales performance in the future, a pessimistic prediction triggers further curiosity. Why things are going worse? Is there any way to turn it up? These all require explanations for the prediction, which compose a satisfactory 'pessimistic answer'.
ML helps human in investigation, but accepting predictions works just like learning fragmented news. One may be confused by apparently contradictory news, but it is the contradiction that implies new knowledge which potentially *harmonises* it. These new knowledges are embedded in the explanations for predictions, and interpretability thus becomes crucial again.

2.2.3 How do we classify interpretability methods?

More than one taxonomies can be applied to interpretability methods. According to the knowledge of the learning model, one classifies an interpretation method as *ante-hoc* or *post-hoc*. Ante-hoc methods are intrinsically built with interpretability, e.g. attention-based methods, decision trees, etc. Post-hoc methods are designed to be applied to explain other already-trained learning models, and are usually model-agnostic, e.g. perturbation methods, gradient methods, etc.
Depending on the scope of explanation, interpretation methods can be *local* or *global*. Local methods includes perturbation methods, gradient methods, etc. As the name suggests, they provide explanations for single predictions. Intuitively, they are easy to understand and fast to compute if one is only interested in explaining a single prediction. However, if one wants to infer outcomes for further inputs, whether the examples are representative will become an issue. To put the inference in an explained regime of datasets, a gigantic amount of instances need to be explained, which booms up the computation burden. Global methods pursue full transparency of a learning model, which automatically embeds explanations for all instances in the entire input space. This is analogous to having a solid background, becoming an expert with systematic knowledge of a field. As one can imagine, becoming an expert calls for painstaking trainings, and thus not every one is talented enough to become an expert. Analogously, accompanied with global methods are usually heavy computation and difficulty in understanding. In cases when human wants only explanations for single predictions, global knowledge contains redundant information. In the following chapter, we are going to discuss interpretability methods using the latter classification.

2.2.4 How do we evaluate interpretability methods?

Since there is no rigorous definition for interpretability, there is no fair play between interpretability methods without talking about applications. A good interpretability method should produce high-quality explanations, and is expected to be portable. Which explanations are favoured? First, it must not lack fidelity, the ability to approximate the predictions, otherwise it would be misleading. The price to pay is usually complexity, which reduces comprehensibility. If the explanations are too complex for human to understand, there will be no point in explaining. As a result, one should make a trade-off between fidelity and complexity. Also, one expects the explanations to be stable. That is to say, explanations should not vary too much under perturbation. This will become an issue for perturbation methods. For local methods, explanations for representative instances are also more significant than others. Below in the discussions on interpretability methods, evaluations are going to be made.

3 Methods of Explanation for Deep Learning Predictions

3.1 Local Methods

3.1.1 Linear Interpretable Model-agnostic Explanations(LIME)

As its name suggests, it is a local method applicable for all models, since we don't expect to know (agnostic) the model. It is based on perturbation around a single prediction. The aim is to provide an interpretation understandable by humans, no matter what actual features the model is using, like using the Fermi's theory as an effective field theory to describe weak interaction at low energy, while behind it stands the massive gauge boson.
While easy to interpret, we expect the model for explanation to be as close as possible to the real one. Thus we need a trade-off between fidelity and interpretability. The solution is to maximise the sum of the two, since both are positive remarks. This is the same to minimise the opposite of the sum of the two, which are easier to define. We define the opposite of interpretability as the complexity of a model $\Omega(g)$, where $g \in G$ is one of the candidate models to approximate the real model $f \in C^1(\mathbf{R}^d) : \mathbf{R}^d \to \mathbf{R}$. The performance of approximation is described by the fidelity function $\mathcal{L}(f, g, \pi_x)$, where $\pi_x(z)$ is the proximity of instance z to that used in the single prediction x in order to measure how close it is to x. The fidelity function is to measure how unfaithful an interpretation candidate model g is to the real model f. Finally, the ideal candidate model will be the one minimising the sum of fidelity function and complexity:

$$\xi(x) = argmin_{g \in G}(\mathcal{L}(f, g, \pi_x) + \Omega(g)) \tag{3.1}$$

around the instance x.

Sparse Linear Explanations In the paper read on LIME, only sparse linear explanations have been discussed. That is, G contains only linear models as candidates $g(z') = w_g z', \forall z'$ for explanation. Also the measure of instance proximity measure $\pi_x(z)$ is chosen to be Gaussian $\pi_x(z) = e^{-\frac{D(x,z)^2}{\sigma^2}}$ with respect to some distance function $D(x, z)$. In this way we can calculate the fidelity function

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \in Z} \pi_x(z)(f(z) - g(z'))^2 = \sum_{z,z' \in Z} e^{-\frac{D(x,z)^2}{\sigma^2}}(f(z) - g(z'))^2 \tag{3.2}$$

as a weighted sum of square deviation from the prediction made by the real model in the proximity of x. The complexity can be chosen according to the context. For example,

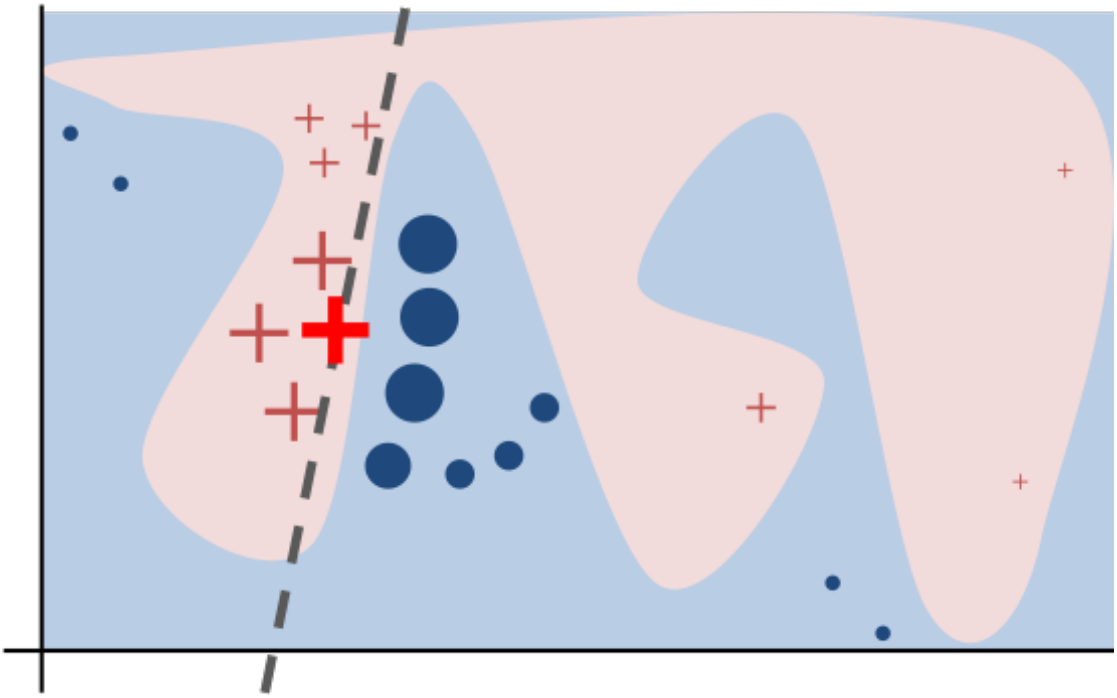


Figure 3.1: LIME: The red bold cross represents the input to be explained. The crosses and dots nearby are perturbations without and with the outcomes changed. See Ref. [6]

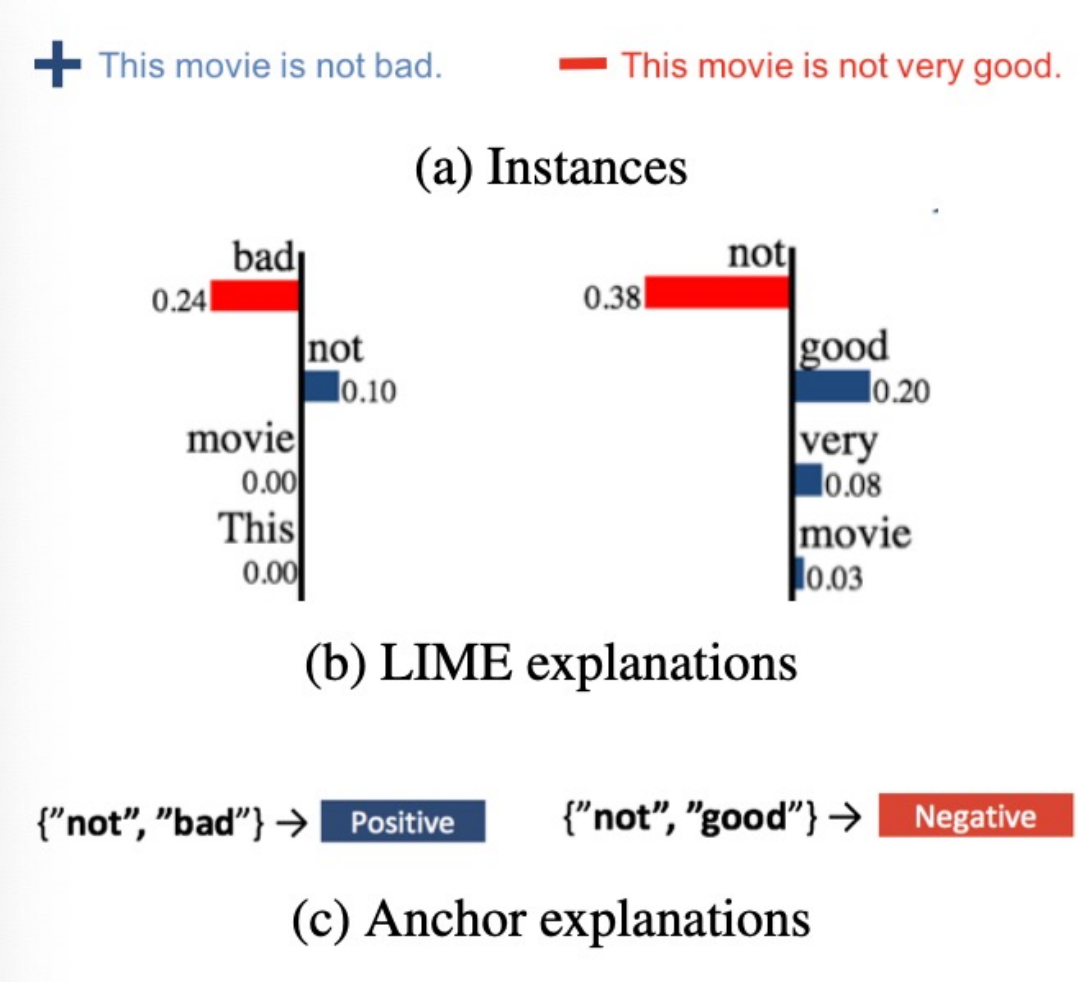


Figure 3.2: Anchors versus LIME: In Anchors, it is the 'anchor' that determines the outcome.

for text classification, one may set a upper cut-off on the number of words so that $\Omega(g) = \infty \mathbf{1}[||w_g||_0 > K]$.
There are certain limitations for this method. The set of candidate models may not contain ideal one for explanation of certain predictions somewhere. That is, one may not reach a global minimum within the window opened by G, or the global minimum is not low enough. Also, this assumes local linearity, i.e. the actual model f should be locally differentiable. Once not, there may be problems with the choice of linear candidate models.

3.1.2 Anchors

Similar to LIME, Anchors is also based on local perturbations around a certain prediction to be explained. Unlike LIME which treats all perturbations isotropically, there is a natural cut-off in Anchors in searching the locality around the instance x in order to maintain the same result with a high probability. For example, to explain why the sentence 'The film is not bad' is positive, one may search around all sentences containing the phrase 'not bad'. This sets a constraint in perturbation, i.e. the phrase 'not bad' should not be broken after perturbation. This constraint works like an anchor, confining the searcher in a certain compact area. The produces interpretations easy to understand intuitively.
Setting an anchor ensures a high probability for the perturbed instances to be predicted positive. The definition of 'high probability' can be somewhat arbitrary, i.e. through some threshold probability $\tau \in (0, 1)$ with which the constraint condition

$$A(x) = \mathbf{1}[E_{D(z|A)}[\infty[f(x) = f(z)]] \geq \tau] \tag{3.3}$$

is fulfilled. The conditioned perturbation domain $D(z|A)$ under anchoring condition A is unknown *a priori*. Therefore one needs to find this anchor first.
Then the problem comes to efficient calculation of anchors. One has now an unknown black-box model f, an instance x, a distribution D of perturbation around x. We set the threshold precision τ of the anchor A to be found, i.e. the probability that the perturbation does not change the result by the black-box model f. It is obvious by definition that $A(x) = \mathbf{1}[prec(A) \geq \tau]$, i.e. the anchoring condition is true if the precision reaches a threshold. Thus

$$prec(A) = E_{D(z|A)}[\infty[f(x) = f(z)]] \tag{3.4}$$

However, given an arbitrary D and the unknown black-box f, it is difficult to calculate the precision of the anchor. Therefore one follows an probabilistic approach, that is, to set a high probability with which the precision of the anchor reaches a certain threshold. Formally, $P(prec(A) \geq \tau) \geq 1 - \delta$ where $\delta \ll 1$. Based on this condition, the chosen anchor is expected to maximise its coverage. Based upon this principle, various algorithms building the anchors have been developed.
The bottom-up approach starts from an empty set. In each run, one adds an element into the anchor and test the increase in precision. It is intractable to find the absolute highest precision, but one can choose a threshold increase in precision $\epsilon \ll 1$ which concludes the building of an anchor.

3.1.3 Gradient-based methods

Gradient Input Perturbation-based methods provide good explanations to the marginal effects of certain features. However, when the number of contributing features grows, the calculation can be extremely slow. What's more, for certain models like DNN with built-in non-linearity, The linear perturbation regime is extremely tiny, making large

perturbations strongly influence the result. Therefore one would avoid introducing perturbations when investigating the importance of input features. Backpropagation-based methods attributes importance scores to all input features in a single backward propagation, avoiding testing the features cumbersomely. Gradient Input calculates the local gradient of the output wrt the input and then multiplies with the input in order to show the importance of each feature. However, the local output may be saturated to reach a plateau with zero gradient, while the importance of the corresponding features is non-zero. For example, when recognising an elephant in a picture, one may regard its long nose as an important feature. While a longer nose makes no difference to a long nose, the length of nose is still important. Therefore it needs to be ameliorated.

DeepLIFT Stands for 'Deep Learning Important FeaTures'. Instead of calculating local gradients, DeepLIFT calculates finite changes with respect to a reference prediction due to a finite change in the input with respect to the corresponding reference input. Given the finite change in input $\Delta x = (\Delta x_1, \Delta x_2, \ldots, \Delta x_d)$ for $x \in \mathbf{R}^d$, we find a finite change in output Δt. Then one can define the contribution of each input feature $C_{\Delta x_1,\Delta t}, C_{\Delta x_2,\Delta t}, \ldots, C_{\Delta x_d,\Delta t}$, such that $\sum_{i=1}^{d} C_{\Delta x_i,\Delta t} = \Delta t$.
In analogy with gradients, one defines the multipliers, i.e. 'discrete gradients' $m_{\Delta x_i,\Delta t} = \frac{C_{\Delta x_i,\Delta t}}{\Delta x_i}$. This can address the problem of saturation since finite change can be captured once having selected a good reference input.

Integrated Gradient To address the problem of feature saturation, one needs to move away from locality. Another way to do this is to integrate the gradient along a certain path. One expects an attribution method to satisfy two fundamental principles as axioms: Sensitivity and Implementation Invariance. The feature saturation crisis demonstrates the violation of sensitivity principle by gradient method, since it may lose sensitivity to saturated key features. DeepLIFT satisfies sensitivity, but violates implementation invariance. Implementation invariance ensures the equivalence between models with identical functions, i.e. same domain, same range and same mapping. Same as: two composite functions with the same domain, range and mapping should be identical no matter how they are composed respectively. The ignorance of intermediate composition implies the chain rule satisfied by gradients $\frac{\partial f}{\partial g} = \frac{\partial f}{\partial h}\frac{\partial h}{\partial g}$. However, the chain rule for its analogy in DeepLIFT works as $m_{\Delta x_i,\Delta t} = \sum_{\Delta y} m_{\Delta x_i,\Delta y} m_{\Delta y,\Delta t}$ where the finite changes in intermediate layer are summed over. Therefore when using the multipliers to attribute importance, the implementation invariance is violated. This may create two sets of values after training with same underlying functional network once there are redundant degrees of freedom. This will lead to a bias in importance attribution.
Suppose we have a function $F : \mathbf{R}^n \to [0, 1]$ representing a DNN. Then the integrated gradients from a baseline input $x' \in \mathbf{R}^n$ is defined as

$$IntegratedGrads_i(x) := (x_i - x_i') \int_0^1 d\alpha \left(\frac{\partial F(x' + \alpha(x - x'))}{\partial x_i}\right. \tag{3.5}$$

The integrated gradients integrates the gradients along the straight path from the baseline to the input. This can be generalised to integration along any smooth curve embedded in $\mathbf{R}^n$ connecting x' and x. Suppose we have a smooth curve $\gamma : [0, 1] \to \mathbf{R}^n$. One

can integrate the tangent vector tracing out the curve along the curve to get the sum of Path integrate gradients:

$$\sum_{i=1}^{n} PathIntegratedGrads_i := \sum_{i=1}^{n} \int_0^1 d\alpha X_i \gamma_i \tag{3.6}$$

Where $X_i = \frac{\partial F(\gamma(\alpha))}{\partial \gamma_i} \frac{\partial}{\partial \alpha}$ is the tangent vector component along the curve γ. We remove the sum to define $PathIntegratedGrads_i = \int_0^1 d\alpha \frac{\partial F(\gamma(\alpha)}{\partial \gamma_i} \frac{\partial}{\partial \alpha} \gamma_i(\alpha)$.
Integrated gradients are special cases of Pathintegrated gradients. They satisfy all desired axioms.
However still, the straight path is the canonical choice. This is because it satisfies symmetry preserving under permutations between input variables.
Although the integrated gradients in principle satisfies all desired axioms, the calculation of integral can be expensive.

3.1.4 SHAP

With a handful of interpretability methods to choose from, one has to make a trade-off between accuracy and interpretability. To explain a single prediction, the comparison between the method chosen and others needs to be made, in order to rationalise the preference for it. A unified framework encoding those methods addresses this issue. Shapley Additive explainations (SHAP) is one such framework, assigning importance values to features.
Usually the model f to be explained, such as DNNs, are complex, assigning each input instance x an outcome $f(x)$. A human-friendly explanation, especially locally, however, should be a simpler model g called *explanation model* mapping a simplified input instance x' with a reduced set of input features to the outcome. This simplified input x', though containing less information, is selectively mapped to x through a function $x = h_x(x')$. The aim of local explanation is to find a model g s. t. $lim_{z' \to x'} g(z') = f(h_x(z'))$ In the case of SHAP, the explanation model g is designed to be a linear function of simplified input features:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{3.7}$$

where $z' \in \{0,1\}^M$, the simplified input instance, is a binary vector. M is the number of input features in the simplified input instance. For all i, $\phi_i \in \mathbf{R}$. Some methods discussed above, including LIME and DeepLIFT, clearly have the interpretations satisfying the form (3.7). This is the basis of a unified framework.
There are simple properties supposed to be fulfilled by additive feature attributions. These are sufficient to uniquely determine the explicit form of the explanation model, through fixing the coefficients ϕ_i.
Local accuracy Locally, the explanation model g should agree with the original model f at the simplified input x' mapped to the original input x through $x = h_x(x')$

$$g(x') = f(h_x(x')) \tag{3.8}$$

Missingness The simplified input x' contains partial features of the original input x. Only those existing contribute to the outcome by the explanation mode g. i.e. Those missing in x' should have no impact.

$$x'_i = 0 \Rightarrow \phi_i = 0 \tag{3.9}$$

Consistency The 'consistency' lies between the marginal contribution of a feature and its attribution in the explanation model. A change in model, if inducing a non-decrease in the contribution of a feature regardless of the others, leads to an increase or invariance in the attribution of this feature in the simplified input. Denote $f_x(z') = f(h_x(z'))$ and $z' \setminus i$ setting $z'_i = 0$. Then for any two models f and f',

$$\forall z' \in \{0,1\}^M, f'_x(z') - f'_x(z' \setminus i) \geq f_x(z') - f_x(z' \setminus i) \Rightarrow \phi_i(f', x) \geq \phi_i(f, x) \tag{3.10}$$

These properties, having set constraints to the additive feature attribution (3.7), lead to a unique solution to the attributions

$$\phi_i(f, x) = \sum_{z' \subseteq x' \setminus i} \frac{|z'|!(M - |z'| - 1)!}{M!}[f_x(z') - f_x(z' \setminus i)] \tag{3.11}$$

where $|z'|$ is the number of non-zero entries in z' and $z' \subseteq x'$ means the non-zero entries in z' form a subset of those in x'. This solution is obtained from combined cooperative game theory, where the solutions are called Shapley values.
We recall that in LIME, the explanation model is to be found through minimisation of the objective function (3.1):

$$\xi(x') = argmin_{g \in G}(\mathcal{L}(f, g, \pi_{x'}) + \Omega(g))$$

But the solution obtained in this way does not follow all properties listed above. In order to find a solution matching that of SHAP, one has to set specific forms of $\pi_{x'}$, L and Ω:

$$\begin{aligned} \Omega(g) &= 0 \\ \pi_{x'}(z') &= \frac{M - 1}{C_M^{z'}|z'|(M - |z'|)} \\ L(f, g, \pi_{x'}) &= \sum_{z' \in Z}[f(h_x^{-1}(z') - g(z'))]^2 \pi_{x'}(z') \end{aligned}$$

One should be careful that the above solution recovering LIME only makes sense formally. LIME and SHAP are after all different interpretability methods and should naturally provide different explanations.

3.2 Global Methods

3.2.1 Linear regression

Given an input instance $\mathbf{x} = (x_1, x_2, ..., x_p)$ with p input features $x_i, i = 1, 2, ..., p$ weighted $\beta_i, i = 1, 2, ..., p$ correspondingly, the outcome from linear regression is $y =$

$\beta_0+\beta^T\mathbf{x}+\epsilon$, a linear combination of the input features, the dual vector $\beta = (\beta_1, \beta_2, ..., \beta_p)$ parametrising their weights, the intercept β_0 providing the background outcome and the assumed gaussian noise $\epsilon \sim N(\mu = 0, \sigma)$ creating both positive and negative errors with small ones preferred to large. This could be the entry-level model with built-in interpretability.
Therefore, it is the weights $\beta_i, i = 1, 2, ..., p$ and the intercept β_0 that determine the linear regression model. During a training process using n instances $\mathbf{x}^{(i),i=1,...,n}$, they are optimised through minimisation of the squared differences:

$$\hat{\beta} = argmin_{\beta_0,\beta_1,...,\beta_p} \sum_{i=1}^{n} (y^{(i)} - (\beta_0 + \beta^T\mathbf{x}^{(i)}))^2 \tag{3.12}$$

The linear relationship between the prediction and the input features makes the model globally transparent, while this also limits its use for explanations: There should be few non-linearities and no interaction between features. Additionally, a large number of existing features, being more commonly observed in practice, blunts the interpretability of linear regression, as one can be drown in the ocean of features with low weights. It is especially difficult when insufficient number of instances are available, since one cannot fit the complete linear regression model with fewer instances than features. In this case, the number of features has to be confined, with part of the features sacrificed. This leads to **sparse linear models**, where **sparsity** is introduced.
Lasso Lasso, standing for *least absolute shrinkage and selection operator*, is a straightforward way to reduce the number of features. By introducing a Lagrange multiplier λ, the optimisation is constrained so that one reaches a regularised munimum. Based on the original minimisation problem

$$min_\beta(\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \beta^T\mathbf{x}^{(i)})^2) \tag{3.13}$$

a regularisation term is added

$$min_\beta(\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \beta^T\mathbf{x}^{(i)})^2 + \lambda||\beta||_1) \tag{3.14}$$

where $||\cdot||_1$ stands for the L1-norm. The Lagrange multiplier tunes the strength of regularisation. As shown in Figure 3.3, the larger the λ, the more features vanish. In this way, one has less non zero features for all instances.

3.2.2 Logistic regression

While linear regression works transparently, it does not work for classification, the outcomes being categorical. We assume here that the categorical outcomes fall in the set $\{0, 1\}$, i.e. they are binary. If linear regression is used, only an optimised hyperplane to be produced. This hyperplane can predict values below 0 or above 1, which make

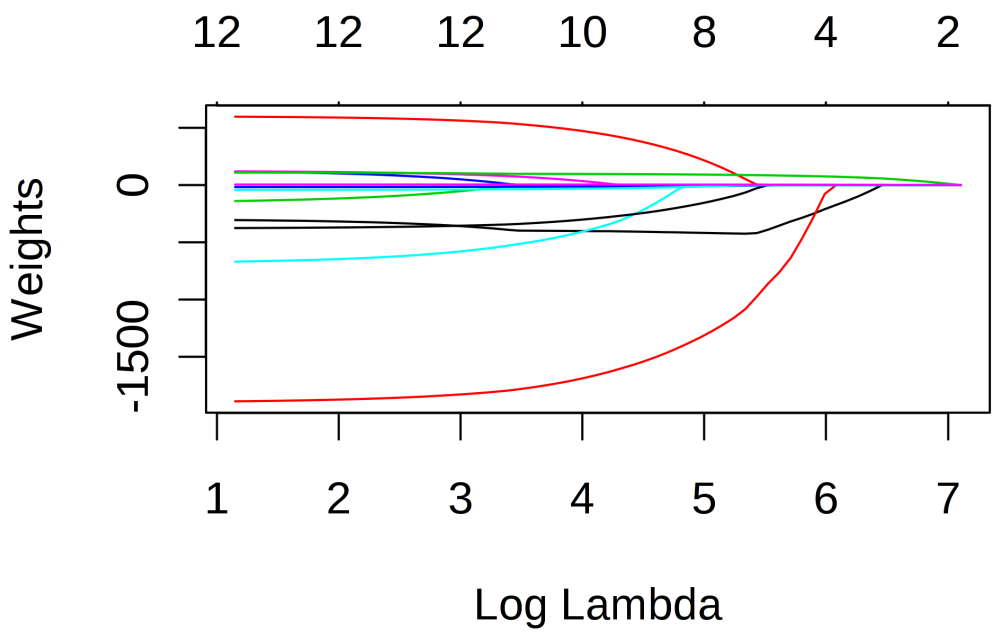


Figure 3.3: The evolution of weights with growing λ

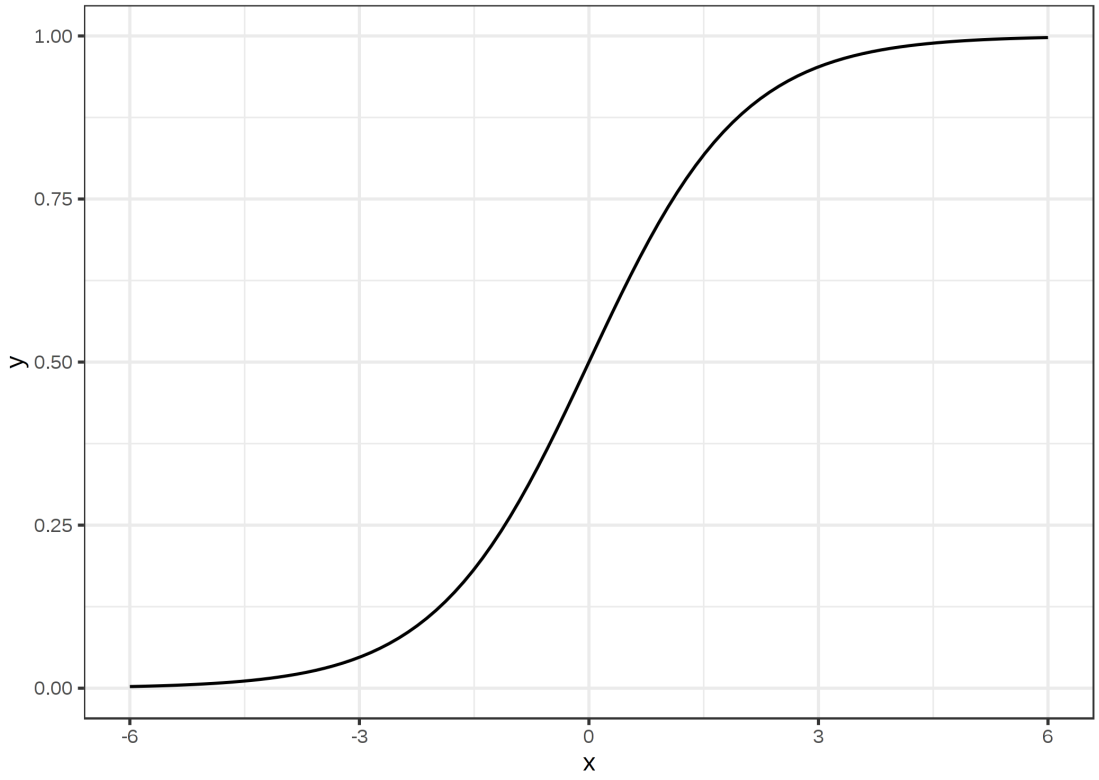


Figure 3.4: The logistic function

no sense. One step further, plugging the linear regression predictions into the logistic function

$$logistic(y) = \frac{1}{1 + e^{-y}} \tag{3.15}$$

such that the outcome falls in open interval (0,1) and can be interpreted as a probability(Figure 3.4). Indeed, we assign the result to be the probability to predict the class '1'

$$P(y^{(i)} = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta^T \mathbf{x}^{(i)})}} \tag{3.16}$$

Still, the model is determined by parameters $\beta_0, \beta_1, ..., \beta_p$, but they *per se* no longer serve as weights as in linear regression, since they cannot be interpreted as partial derivatives. To interpret the model, instead of looking at differences, we look at ratios.
We define the 'odds' function as the ratio between the probabilities of getting the two potential outcomes

$$odds := \frac{P(y = 1)}{1 - P(y = 1)} = \frac{P(y = 1)}{P(y = 0)} = e^{\beta_0 + \beta^T \mathbf{x}} \tag{3.17}$$

Then it is easy to see that $e^(\beta_j)$ can be interpreted as the influence of unit addition to the feature x_j on the odds function in terms of a ratio

$$e^{\beta_j} = \frac{e^{\beta_0+\beta_1 x_1+...+\beta_j(x_j+1)+...+\beta_p x_p}}{e^{\beta_0+\beta_1 x_1+...+\beta_j x_j+...+\beta_p x_p}} = \frac{odds(x_j+1)}{odds(x_j)} \tag{3.18}$$

It is clear that if we do $\frac{odds(x_j-1)}{odds(x_j)}$, we will obtain $e^{-\beta_j}$, the inverse of e^{β_j}. This implies that the 'weights' here do not have homogeneous effects for all feature values. One result is that, for example, if the one step forward by feature x_j crosses a critical value which deterministically separates the binary outcome, the weight e^{β_j} will diverge and so will β. These are harmful for an interpretation. Since it is one step further than linear regression, the global transparency and problems of linear regression still follow here.

3.2.3 Decision tree

While linear regression and logistic regression succeed in linear fitting where features contribute independently, further methods should be developed for nonlinear cases and those with interacting features. Decision tree is one example, featured by data-splitting into subsets with empty intersections between each other. By setting cutoffs to features, the dataset is divided into subsets accommodating instances. As is shown in Figure 3.5, several cutoffs can be applied stepwise, forming a tree structure, thus named 'decision tree'. Each cutoff creates a separation, giving birth to subsets called *internal nodes* or *split nodes*, until the end, where the final subsets at the bottom are called *terminal nodes* or *leaf nodes*, the data in each precisely fitted by a simple model. To each leaf node, the average outcome of training data in this node is assigned as its outcome prediction.

Given a raw nonlinear dataset with interacting features, one is expected to grow a tree in order to split the data and fit them separately with simple models. This should be implemented by algorithms. To grow an efficient decision tree, the crucial points lie in finding right splits at each level, which determines the tree structure, and simple models in the leaf nodes. These make the algorithms different from each other.

Once a decision tree is grown, global understandings for individual predictions can be realised, since the only thing to do is to look for the leaf node the instance belongs to step by step, where a straightforward simple model is waiting. In the case of short trees, this process is feasible.

This way of finding the outcome makes the results discrete in the neighbourhood of the cutoffs, where the marginal effect can be singular. This is not ideal for a good explanation. For continuous input features, next to a cutoff, a tiny increment, which makes no difference in practice, may cause a significant change in the outcome. This by no means provides a good explanation. Additionally, a long tree with numerous leaves represents the model globally, but its monstrous complexity makes it human-unfriendly, since nobody is able to understand it.

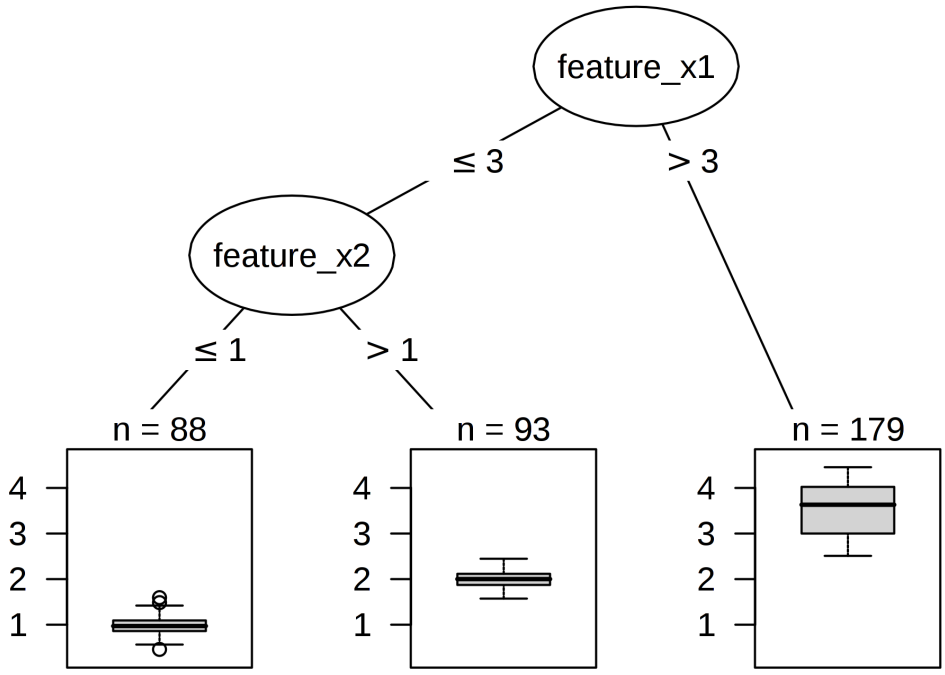


Figure 3.5: Decision tree

4 Toy Experiments

4.1 LIME

In this toy experiment, the iris dataset, as an example of tabular dataset, is used, due to its simplicity. The iris dataset groups its instances into 3 classes: Setosa, Versicolour and Virginica. Each instance is characterised by 4 features: Sepal Length, Sepal Widthm Petal Length and Petal Width. The ML model to be explained is a random forest classifier. The classifier is first trained using the entire dataset. As LIME is a local explainer for single instances, a randomly chosen instance in the training iris dataset is to be explained through local perturbation. The perturbed instances are assigned outcomes through prediction by random forest, the model to be explained. Amongst the perturbed instances, as depicted in the 'prediction probabilities', 91% are predicted to be versicolour, identical with the instance selected. 3% are setosa, and 6% virginica. The key critical feature values, the petal width and petal length, which separates versicolour from others, are 1.30 and 4.35 respectively. It is clear that 9% of the perturbations have gone across the critical feature values, leading to altered outcomes.

Entrée [7]:
```
from __future__ import print_function
import sklearn
import sklearn.datasets
import sklearn.ensemble
import numpy as np
import lime
import lime.lime_tabular
np.random.seed(1)
```

Entrée [8]:
```
iris = sklearn.datasets.load_iris()
train, test, labels_train, labels_test = sklearn.model_selection.tr
rf = sklearn.ensemble.RandomForestClassifier(n_estimators=500)
rf.fit(train, labels_train)
```

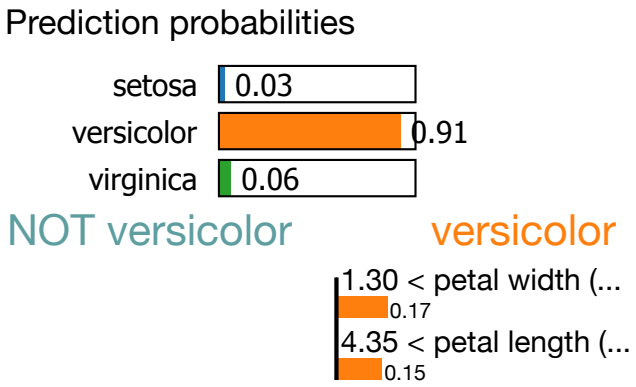
Out[8]:
```
RandomForestClassifier(n_estimators=500)
```

Entrée [9]:
```
sklearn.metrics.accuracy_score(labels_test, rf.predict(test))
```

Out[9]:
```
0.9666666666666667
```

Entrée [10]:
```
TabularExplainer(train, feature_names=iris.feature_names, class_name
```

Entrée [13]:
```
i = np.random.randint(0, test.shape[0])
exp = explainer.explain_instance(test[i], rf.predict_proba, num_fea
exp.show_in_notebook(show_table=True, show_all=False)
```



Feature	Value
petal width (cm)	1.50
petal length (cm)	4.50

4.2 DeepBind

A less trivial example of interpretability is that of DeepBind, a deep learning method predicting sequence specificities of DNA and RNA bond by protein.

In a genome sequence, at each position lives one of the four possible bases, represented by a letter. The genome sequence thus can be regarded as a letter sequence with an alphabet containing 4 letters. DeepBind uses a deep convolutional neural network trained with large datasets provided by experimentalists who are accessible to experimental results showing the motifs preferred in protein binding in a genome sequence. After being trained, DeepBind learns which letter is preferred by protein at each position and scores each letter at each position. Once a new sequence is known, the trained model scans it and predicts a binding probability at each position, since the scores of different letters at each position are known. In this way, the probable binding motifs can be found in any sequence. This helps biomedical practitioners identify the effect of mutation on protein binding and thus diseases.

Here using LIME and Anchors, we have tried explaining the DeepBind model. An input sequence leads to a single prediction, binding or not binding. Herein the input sequence 'AGTGTGTGTG' written in the box is to be explained. The local explanations, using LIME and Anchors, are implemented through perturbing the input sequence to get predictions for perturbed inputs. It is shown in Figure 4.2 that 86% of the perturbed sequences around the input still produce the same 'Not binding' result, the remaining 14% 'Binding'. Scores for the letters at each position on the input sequence has been ranked in the central column, with the orange ones contributing to binding while blue non binding. This rank has also been depicted in the text with highlighted words on the right hand side of the central column, where darker colours indicates higher scores, no matter orange of blue. In this way one can recognise in the sequence the letters at each rank. The lack of motifs with strong binding tendency leads to the 'Not binding' result. Meanwhile, another perturbation approach, used in Anchors, has been tried. Although no non-empty anchors have been built, the perturbation still gave a precision of 99%.

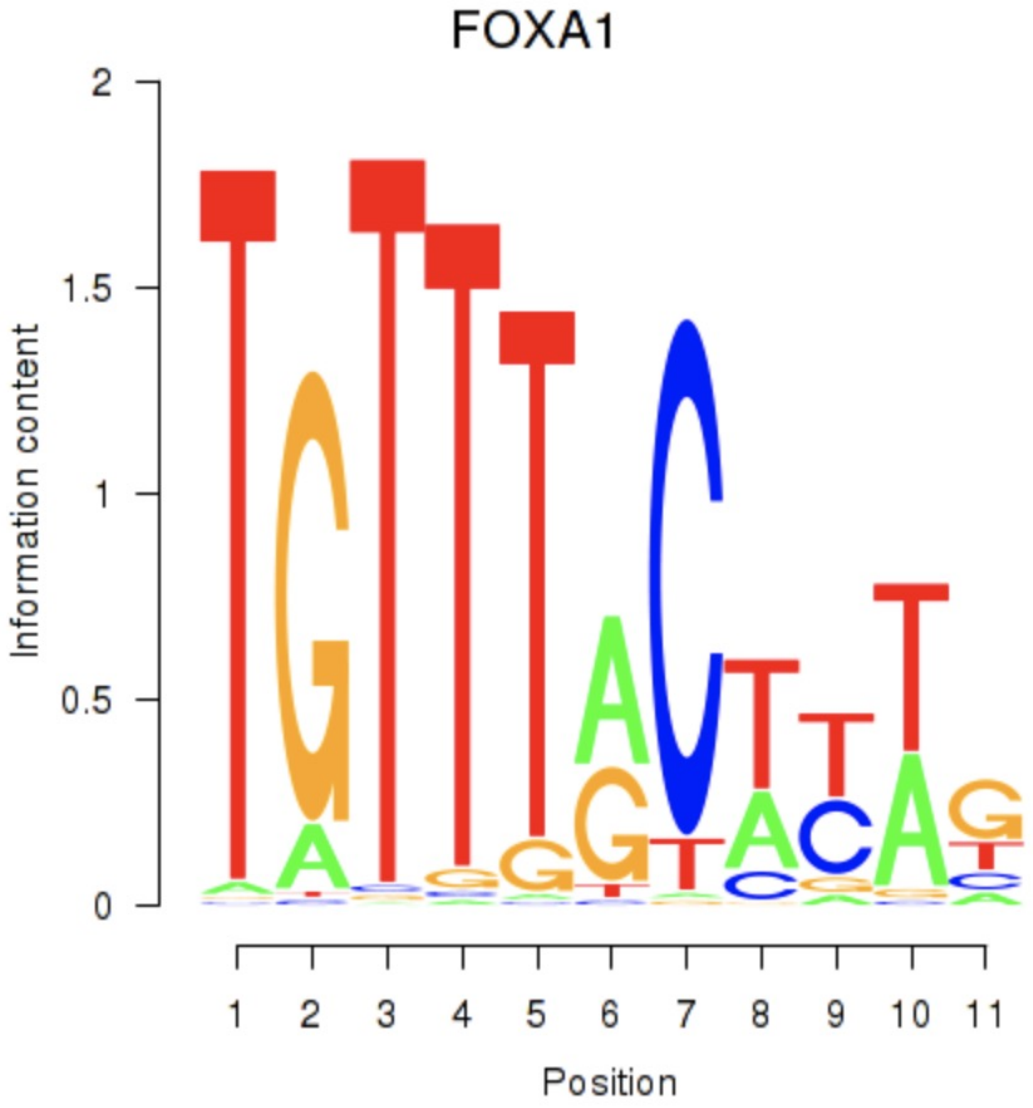


Figure 4.1: Explanation of DeepBind for a sequence of given length provided by perturbations

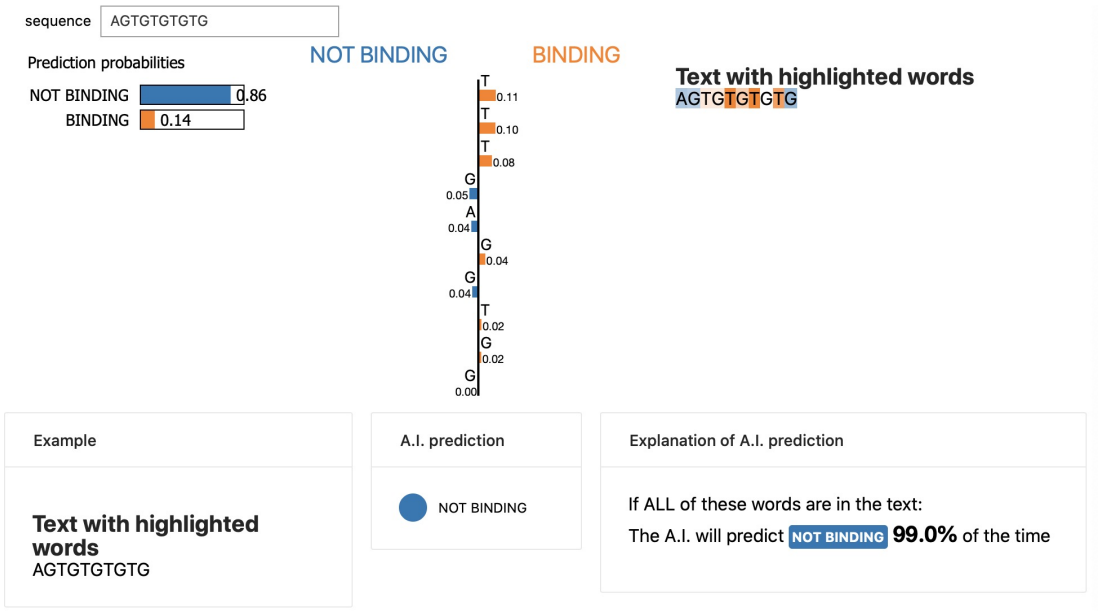


Figure 4.2: Explaining one prediction by DeepBind

5 Conclusion and Outlooks

In this report, several local and global interpretability methods for machine learning, especially deep learning, have been reviewed with some toy experiments. Interpretability, as has been shown, provides a strong reference for human experts to evaluate the results provided by machine black boxes. Local perturbations tell users how safe it is to make such a prediction through showing how far it is away from changing the prediction, while global methods work on making the learning model and prediction process fully transparent. Different methods shall be used depending on the context and computational resources. Can those explanations be made easier to understand? Can we rely more heavily on them? Is it possible to shorten the time for explaining the predictions? These are all under investigation by practitioners in this field. It is expected that accurate, stable, less complex and computationally cheaper interpretability can be developed and applied to more risk-sensitive areas.

6 Acknowledgement

First, I feel grateful for the effort by my supervisor, Dr. Maria Rodriguez Martinez and the head teacher of ICFP soft matter track, Prof. Frédéric Restagno to make the internship work despite all difficulties. It is wise to adapt the content of the internship to the current situation, and help from Mr. An-Phi Nguyen, PhD student in Maria's group in building local working environment is appreciated. Through this internship, my understanding in machine learning and interpretability has been promoted, along with Python programming skills. This experience will surely help in my future career. It is my pleasure to have once worked in a group in the prestigious IBM Research Lab Zurich, although remotely.

No one was able to tell beforehand what a difficult period the world is facing right now. The outbreak of CoVid-19 has changed every one's life. Facing the threats from the epidemics, it is the risk-taking effort by the medical practitioners and policemen that ensures every one's work and life at home. Having benefited from them, I must say thank you to those protecting us.

References

[1] Babak Alipanahi et al. "Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning". In: *Nature Biotechnology* 33.8 (2015), pp. 831–838. DOI: 10.1038/nbt.3300. URL: https://doi.org/10.1038/nbt.3300.

[2] Marco B Ancona et al. "A unified view of gradient-based attribution methods for Deep Neural Networks". In: *ArXiv* abs/1711.06104 (2017).

[3] Francis Bach. *Lecture Notes in Introduction to Supervised Learning*. 2020.

[4] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[5] Christoph Molnar. *Interpretable Machine Learning*. URL: https://christophm.github.io/interpretable-ml-book/. (accessed: 29.05.2020).

[6] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why Should I Trust You?": Explaining the Predictions of any Classifier". In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2016.

[7] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-Precision Model-Agnostic Explanations". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

[8] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features through Propagating Activation Differences". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 3145–3153.

[9] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 3319–3328.