Homework 2

CS 329T: Trustworthy Machine Learning Spring 2021

Due: April 23, 11:59 PM PST
Late Due Date: April 25, 11:59 PM PST.

Academic Integrity Policy

This is an individual homework. Discussions are encouraged but you should write down your own code and answers. No collaboration on code development is allowed. We refer to Stanford Honor Code.

Late Day Policy

You will have **5** late days to use during the whole course, but no more than **2** late days can be used for a single homework.

Written Exercises

This homework contains some written exercises. We will not accept hard-copies and do not hand-write your answers. Latex (Overleaf: `https://www.overleaf.com`) and Markdown are two recommended languages to generate the clean layout but you are free to use other software. You will need to submit the written part to **Homework2-PDF** on Gradescope, separately from you code submission. You can use the LaTeXtemplate provided to fill in the answers. Mention your name and SUNet Id at the top of the pdf.

Coding Exercises

You will be doing the coding part of the homwework in a jupyter notebook `hw2_solutions.ipynb`, provided as part of the zip folder. We recommend using Google Colab since one of the exercises needs a GPU, which is available for free via Colab. To open the notebook in Colab, go to `http://colab.research.google.com/`, and upload the notebook using *File* $\rightarrow$ *Upload Notebook*. Submit the completed `hw2_solutions.ipynb` as a jupyter notebook to **Homework2-Code** on Gradescope. **Make sure that the file is called `hw2_solutions.ipynb` or else the autograder will fail**.

Contents

Part I

Explainability for Traditional ML Models

1 Model-agnostic explanations

We first focus on explanation frameworks that attempt to explain decisions of a black-box model f which operates on a D-dimensional input vector $\mathbf{x} = (x_1, x_2, \ldots, x_D)$.

We will specifically discuss ways to establish *local* interpretability, or understanding why a decision is made for a single instance $\mathbf{x}$. In order to do so, we construct a *surrogate model* g which approximates the behavior of f in a region around $\mathbf{x}$. More concretely, we seek to optimize a function $L(f, g, \mathbf{x})$ which measures the closeness of the functions f and g in the neighborhood of $\mathbf{x}$, subject to strong constraints on g to ensure its interpretability – constraining g to be a linear model is the most natural way to ensure that a human can fully understand its behavior.

1.1 Local Surrogate Models (LIME)

LIME [5] proposes the construction of an interpretation by directly learning a linear model that mimics f in the neighborhood of a point of interest $\mathbf{x}$. A LIME model takes the following steps:

- Perturb $\mathbf{x}$ to generate a set of training points X in the neighborhood of $\mathbf{x}$.
- Call the black-box model f on each point in X to obtain predictions Y.
- Fit a linear model g to X, Y
- Report the coefficients of g, which show how changing each feature value would influence the prediction made by f.

A key choice in the implementation of a LIME model is determining how to perturb the data in a reasonable manner. Strategies to do so include:

1. Adding multidimensional Gaussian noise to a data point
2. Adding noise to certain dimensions of a data point, while leaving others fixed in accordance with some prior about the data – for example, we may expect a certain value is constant across the population of interest
3. For tabular or categorical data, it is unclear if a perturbed data point will even belong to the support of the data distribution, making the predictions of f meaningless. In such cases, we can instead select a weighted combination of other actual training instances, with weights chosen appropriately to increase the influence of points near $\mathbf{x}$. Note that even in this scenario, the definition of *near* is somewhat qualitative.

Written Exercise 1. [4 points] *Consider each of the following domains of data, and briefly discuss how you would perturb an instance of each data type:*

1. *Real-valued data in* $\mathbb{R}^d$
2. *Images of human faces*

3. Sentences of English natural language text

Coding Exercise 2. [8 points] *Consider a LIME algorithm where we define the neighborhood of a point x as a weighting of all other data points in the dataset; specifically, with the weight of another point x' equal to $e^{-|x-x'|^2/d}$ for some parameter d (this is similar to the exponential smoothing kernel used in the official implementation of LIME for tabular data).*

Using the code in `hw2_solutions.ipynb`*, develop interpretations for the univariate function f at $x = 0$ by weighting all other data points using the given kernel with varying values of d, and then fitting a linear regression to these labelled points. You should do this by implementing the* `weighted_neighborhood_LIME` *method within the notebook.*

The coefficient of the linear model should reveal if the feature x has a positive, negative, or neutral impact on the prediction made by f. What impact is shown when using a value of $d = 1$? What if $d = 10$ or $d = 100$?

Written Exercise 3. [2 points] *Suppose that feature importances given by LIME for a large, tabular dataset show that a small set of feature are extremely important while the rest have equally low importances. How would you force LIME to isolate only features with high importance?*

1.2 The Shapley Value

SHAP [4] and QII [2] are other ways to provide local interpretations of a black box model f in the neighborhood of a point $\mathbf{x}$ with a linear surrogate model. Unlike LIME, these methods choose the coefficients of this model not through regression, but in order to satisfy certain normative axioms for interpretability.

SHAP approximately decomposes the prediction of the model at $\mathbf{x}$ as

$$g(\mathbf{a}) \approx \phi_0 + \sum_i \phi_j a_j,$$

where the sum is over all features and each a_i is a binary variable indicating if feature a_i is used in make the prediction. In lecture, we outlined the axioms that Shapley values satisfy – these are exactly the normative axioms that methods such as SHAP and QII satisfy as well.

To formalize this notion, let S represent a set of features, $\setminus S$ all features not in S, and $f(x_S, X_{\setminus S})$ a random variable which represents the output of the black-box model f when given a feature vector which is identical to x for each feature in S and drawn randomly otherwise.

Then, the *value* of a certain feature set can be defined as

$$v_{f,x}(S) = E_{X_{\setminus S}|X_S}[f(x_S, X_{\setminus S})],$$

the expected value of the model's output when the features in S are fixed to their values at x and other features vary according to the conditional data distribution. The contribution of a feature i is then given by the Shapely value associated with $v_{f,x}$,

$$\phi_i = \sum_{S \subseteq [N]} \frac{1}{N\binom{N-1}{|S|}} \left(v_{f,x}(S \cup \{i\}) - v_{f,x}(S)\right).$$

Thus, we measure the impact of using feature i in the model at point x, averaged across all possible choices of other features to also use as model inputs.

Written Exercise 4. [8 points] *Consider a model $f(\mathbf{x}) = \prod_i x_i$ which returns the product of each component of its real vector-valued inputs $\mathbf{x} \in \mathbb{R}^d$. Assuming that across all of the data each x_i is independently distributed according to a standard normal distribution, calculate in closed form the SHAP scores for the features corresponding to each x_i at an arbitrary point $\mathbf{x} \in \mathbb{R}^d$. Comment on whether your answer matches intuition.*

In the previous question, you may find that SHAP does not provide a very intuitive answer for multiplicative models. Luckily, most black box models like Neural Networks instead rely on a combination of *additive* functions and point nonlinearities.

Written Exercise 5. [3 points] *In practice, it is not easy to sample from the conditional distributions $X_{\setminus S}|X_S$ of some arbitrary subset of features given values for the remaining ones. Another approach is to instead draw from the marginal distribution for $X_{\setminus S}$, disregarding the information given by X_S. This is called the* interventional *approach, and is often much more tractable and performant in practice.*

Discuss some potential drawbacks you may see in the interventions method. In particular, comment on issues that may arise when f is called on inputs that may lie outside the support of the overall joint distribution of the data.

Written Exercise 6. [5 points] *In this exercise, we will analyze how local explanations via Shapley values can be extrapolated to give more global insight about a model's performance. Within the SHAP section of* `hw2_solutions.ipynb`*, use the Boston housing prices dataset and the provided code to train an* `xgboost` *model on this dataset. Then use the* `SHAP` *package to generate Shapley value estimations for each data point. For each feature, create a Shapley dependence plot: on the x-axis, plot feature value, and on the y-axis, plot the corresponding Shapley value.*

Isolate 2-3 features with interesting trends (monotonicity, linearity, jumpiness, etc.). Paste your plots for these features below and comment on what makes them particularly interesting. You may want to refer to the Boston housing dataset documentation to understand each feature: https://scikit-learn.org/stable/datasets/toy_dataset.html#boston-dataset

2 Model-specific explanations

In certain cases, more insight about the particular model class can lead to faster, more accurate, or more interpretable explanations. We explore this here.

2.1 TreeSHAP

TreeSHAP is a variant of SHAP for tree-based models such as decision trees and random forests. TreeSHAP uses the condition distribution in defining its value function, i.e. $E_{X_{\setminus S}|X_S}(f(x)|x_S)$.

Written Exercise 7. [4 points] *Consider a feature that has no influence on a model's prediction. Is it possible that it is given non-zero influence with TreeSHAP? Describe why or why not this is the case.*

Part II

Attribution in Vision Models

In this section, you will use the `trulens` library to implement various gradient-based attribution methods for vision-based deep neural networks. In particular, we will be using the VGG16 model and the ImageNet dataset as examples.

Before starting this part, we recommend familiarizing yourself with the Trulens library here: https://truera.github.io/trulens/.

3 Gradient-Based Attribution Methods

3.1 Saliency Map

Saliency Maps [1] compute the local gradient of a class of interest w.r.t an input to find the attribution scores. Moreover, current work has found that the product of input $\times$ grad produces clearer visualization result. In the homework, we use the following definition for Saliency Map:

Definition 1 (Saliency Map)**.** *Consider a model $y = f(x)$ that takes an input $x \in \mathrm{R}^d$ and outputs y, a distribution of scores for each class. We denote the $y_c = f_c(x)$ as the scores for the class c. The Saliency Map $S_c(x)$ for class c is defined as*

$$S_c(x) = x \odot \nabla_x f_c(x) \tag{1}$$

where $\odot$ denotes the element-wise multiplication.

Coding Exercise 8. [6 points] *Implement* `visualize_saliency_map` *within the Homework 2 notebook using the* `trulens` *library. For the provided image of the weasel in the notebook, generate saliency maps for the top five predicted classes according to the loaded VGG16 ImageNet model and paste them below. Use the* `MaskVisualizer` *class within* `trulens` *to help you visualize the attribution.*

Written Exercise 9. [2 points] *Discuss and justify why we may want to use the penultimate layer of our predictive model f when calculating a saliency map, instead of the final outputs which are normalized with softmax.*

Written Exercise 10. [2 points] *What would be a possible drawback of multiplying the input with the gradient?*

3.2 Integrated Gradient

Integrated Gradient [6] aims to solve the vanishing gradient problem in Saliency Map, while it satisfies several desirable axioms.

Definition 2 (Integrated Gradient)**.** *Consider a model $y = f(x)$ that takes an input $x \in \mathrm{R}^d$ and outputs y, a distribution of scores for each class. We denote the $y_c = f_c(x)$ as the scores for the class c and x_b as the baseline input. The Integrated Gradient $IG_c(x, x_b)$ for class c is defined as*

$$IG_c(x, x_b) = (x - x_b) \odot \int_0^1 \nabla_x f_c(x_b + t(x - x_b))dt \tag{2}$$

where $\odot$ denotes the element-wise multiplication. In the implementation, we use the following equation to approximate Eq. 2.

$$IG_c(x, x_b) \approx (x - x_b) \odot \frac{1}{N} \sum_{i=1}^{N} \nabla_x f_c \left(x_b + (x - x_b)\frac{i}{N} \right) \tag{3}$$

where N is the number of steps used for the approximation.

Coding Exercise 11. [6 points] *Implement* `visualize_integrated_gradients` *within the Homework 2 notebook using the* `trulens` *library. For the provided image of the weasel in the notebook, generate integrated gradient visualizations for the top predicted class (should be* `weasel`*). Compare and contrast how the attribution map changes for the following choice of baselines:*

- *An all-black image*
- *An all-white image*

- *An image with random noise*

as well as how it differs from the vanilla gradients (saliency map) implementation.

Written Exercise 12. [3 points] *Explain the* completeness *axiom which Integrated Gradient satisfies.*

Written Exercise 13. [5 points] *Given a model* $y = min(2x_1 + 3x_2, 1)$ *and a baseline* $x_1 = x_2 = 0$*, compute the integrated gradient for point* $x_1 = 2, x_2 = 1$*.*

3.3 Saliency in Linear Models

Given linear model for d-dimensional inputs defined by the C-class score function $f(x) \stackrel{\text{def}}{=} Wx+b$, analytically derive the attribution of score $f_c(x)$ in terms of vectors x, b and matrix W according to:

Written Exercise 14. [2 points] *Saliency Map method.*

Written Exercise 15. [2 points] *Integrated Gradients method with baseline* $x_b = \vec{0}$*.*

Also:

Written Exercise 16. [2 points] *Is it useful to ask for the same according to the Influence Directed Explanations method as described by Leino et al. [3] ?*

3.4 Influence-Directed Explanations

Influence-Directed Explanation [3] introduces the concept of *distributional influence* in explaining the internal behavior of neural networks.

Definition 3 (Distribution of Influence)**.** *Consider a feed-forward model* $y = f(x)$ *that takes an input* $x \in \mathrm{R}^d$ *and outputs* y*, a distribution of scores for each class. We denote the* $y_c = f_c(x)$ *as the scores for the class* c*. Given a user-defined layer* l *that separates the model into the internal output of layer* l *as* $h^l(x)$ *and the rest part as* $g^l(h^l(x))$*. The distribution of influence* $\mathcal{I}_c^l$ *at layer* l *is defined as*

$$\mathcal{I}_c^l = \int_{x \in \mathcal{X}} \frac{\partial f_c(x)}{\partial h^l(x)} dx \tag{4}$$

where $\mathcal{X}$ *denotes the distribution of input. In the implementation, we use the following approximation as well:*

$$\mathcal{I}_c^l \approx \frac{1}{N} \sum_{i=1}^{N} \frac{\partial f_c(x)}{\partial h^l(x_i)} \tag{5}$$

where N *is the size of dataset sampled from the real-world distribution.*

By finding the neuron at layer l with highest influence score, we locate the expert neuron responsible for the classification towards class c. To visualize the input feature that has high contribution towards the expert neuron, we compute the gradient of the output of expert neuron w.r.t the input. Formally,

Definition 4 (Expert Neuron Attribution)**.** *Consider a feed-forward model* $y = f(x)$ *that takes an input* $x \in \mathrm{R}^d$ *and outputs* y*. Given a user-defined layer* l *with the distribution of influence* $\mathcal{I}_c^l$ *towards class* c*. Denote* $\mathcal{I}_c^l[i]$ *and* $h_i^l(x)$ *as the influence score and the output for the* i*-th neuron at layer* l*. The Expert Neuron Attribution score* $A(x)$ *is defined as*

$$A(x) = x \odot \nabla_x h_{i*}^l(x) \quad i^* = \arg\max_i \mathcal{I}_c^l[i] \tag{6}$$

Coding Exercise 17. [8 points] *Use* `trulens` *to implement the* `get_internal_influences` *function within the homework notebook for an arbitrary quantity of interest, distribution of interest, and layer of a model.*

Written Exercise 18. [3 points] *Report the indices of the post-RELU expert neuron at* `fc1` *layer w.r.t. the* `weasel` *class, (A) when the distribution of interest is the single weasel image and (B) when the distribution of interest is a linear interpolation between a zero-vector baseline (a black image) and the given point, respectively.*

Discuss what you observe: do the experts correspond to some human-understandable concept? Do they differ between the two DoIs? Feel free to visualize the expert on more than a single image to derive some conclusions.

Written Exercise 19. [2 points] *Repeat the same exercise above but for the class* `polecat`*.*

Written Exercise 20. [5 points] *For the two classes* `weasel` *and* `polecat`*, report and visualize the expert neuron (in the same layer) that distinguishes the first class from the second. You will need to use a quantity of interest that compares scores of the two classes. For the distribution of interest, use the same point-wise and linear interpolation DoIs as in previous exercises. Discuss what you observe: do the experts correspond to some human-understandable concept? Did the differences in distribution of interest have any impact?*

4 Attribution Method Evaluation

Evaluation of attribution methods remains an open space for the current researchers. Several metrics are discussed but a unified framework is still missing in this area. In this section you are going to implement metrics to evaluate the performance of attribution methods and apply them to the three methods you have implemented in this homework: Saliency Map, Integrated Gradient and Influence-Directed Explanations.

The dataset you are going to use is a small portion of ImageNet data and you can download by: `wget https://hw2dataset.s3.amazonaws.com/ImageNet_1K.npy`

4.1 Visual Comparisons

Written Exercise 21. [6 points] *Select a subset of the provided ImageNet images, using both images that are correctly and incorrectly classified. Compare and contrast the attributions produced by the three methods. Example points of comparison may include, but should not be limited to:*

- *Is one of the attribution methods consistently more convincing to you as a human who presumably can recognize the objects pictured?*
- *How do the methods fare in relation to pointing out elements of an image that should definitely have no relationship with the pictured object?*
- *Are some methods easier to interpret than others? What characteristics make an attribution easier to interpret?*
- *Is visual comparison a good approach to distinguish attribution methods? why or why not?*
- *Do you find that the baseline choice affects the attribution method greatly? Why do you think that is and what would be the most appropriate baseline?*

4.2 Average Drop %

Unfortunately visual evaluations are subjective and objective methods are hard to come by. In this homework we will explore two objective methods. The first is Average % Drop. This method and the one that follows operates on attributions that do not distinguish between the color channels. **To create a pixel attribution from the full attribution you have worked with so far by setting setting the attribution of a pixel as the average attribution of its three channels.**

Definition 5 (Average Drop %)**.** *Given a (pixel) attribution map A for an input x and the model $f(x)$, let $M_A(x)$ denote a mask function that keeps only the pixels in x whose pixel attribution score in A is positive while setting all other pixels to 0. The Average Drop % score AD is then defined for a set of instances D as:*

$$AD(D) \stackrel{\text{def}}{=} 100\% \times \frac{1}{|D|} \sum_{x_i \in D} \frac{\max\left[f_{c_i}(x_i) - f_{c_i}(M_{A_i}(x_i)), 0\right]}{f_{c_i}(x_i)} \tag{7}$$

where $f_{c_i}(x_i)$ is the pre-softmax output score for class c for instance x_i and $c_i \stackrel{\text{def}}{=} \arg\max(f(x_i))$.

Written Exercise 22. [6 points] *Implement the functions under the Average Drop section of the homework notebook to compute the average drop score for the provided images of the `flamingo` for vanilla Saliency Maps and Integrated Gradients (with an all-zero baseline). Report these two values.*

Written Exercise 23. [4 points] *Based on the definition of Average Drop %, what does this metric evaluate about an attribution map? Does a higher Average Drop % indicate an attribution map is better than another and in what sense? Discuss whether Average Drop % is a reasonable objective measure of attribution goodness.* [2 bonus points] *Bonus points available for especially thoughtful discussion.*

4.3 Necessity Ordering

Necessity Ordering Score [7] is another objective measure of an attribution's fidelity. It applies to individual images/attributions as opposed to entire datasets as was the case for average drop:

Definition 6 (Necessity Ordering Score)**.** *Given a (pixel) attribution map A of an input x and a model $f(x)$, denote $\pi_A(x)$ as a list of pixels of x ordered by their positive attribution score in A, biggest first. Denote $R(i, \pi_A)$ as the masking of x that replaces all the pixels up to the i-th according to ordering π_A by 0. Everything else about x is left unchanged. The Necessity Ordering Score for the attribution map $N_o(x, A)$ is defined as*

$$N_o(x, A) \stackrel{\text{def}}{=} \frac{1}{N+1} \sum_{i}^{N} \max\left[f_c(R(i, \pi_A)) - f_c(\vec{0}), 0\right] \tag{8}$$

where N is the number of features, $f_c(\cdot)$ denotes the pre-softmax output score for class c, and $\vec{0}$ is an all-zero "blank" input.

Written Exercise 24. [4 points] *Based on the definition of Necessity Ordering, what does this metric evaluate about an attribution map? Does a higher Necessity Ordering indicate an attribution map is better than another and in what sense? Discuss whether Necessity Ordering is a reasonable objective measure of attribution goodness.*

References

[1] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Mueller. How to explain individual classification decisions, 2009.

[2] A. Datta, S. Sen, and Y. Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 598–617, 2016. doi: 10.1109/SP.2016.42.

[3] Klas Leino, Linyi Li, Shayak Sen, Anupam Datta, and Matt Fredrikson. Influence-directed explanations for deep convolutional networks. *arXiv preprint arXiv:1802.03788*, 2018.

[4] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[5] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[6] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

[7] Zifan Wang, PiotrPiotr Mardziel, Anupam Datta, and Matt Fredrikson. Interpreting interpretations: Organizing attribution methods by criteria, 2020.