CS 281 Homework 2: Feature Attribution with SHAP

Due May 13 2024, 1:00 PM

Introduction

In this assignment you will explore the theory and applications of SHAP. As some portions of the assignment require running models on gpus, all starter code for the programming portions are provided in a jupyter notebook here. You can save a copy of this notebook on your own drive and run the code on Google Colab which gives you free access to gpus. Note that you must click "File > Save a copy .." in order to save your edits to the notebook and run the code. When running your code, make sure you are opting to use GPUs by selecting the GPU option in "Runtime > Change runtime type".

Deliverables: Please export the jupyter notebook containing your code and responses as a .pdf file, and submit a single .pdf file to Gradescope. Please also write your written response to Problem 1 in markdown cells within your jupyter notebook. Note that the markdown supports math equations, all you need to do is wrap your equations around $ $.

SHapley Additive exPlanations (SHAP)

Let $\mathcal{X}, \mathcal{Z}$ be the original input space and simplified input space, respectively. Further let $h_x : \mathcal{Z} \to \mathcal{X}$ be a mapping function specific to an input x which recovers the original inputs from the simplified inputs. For Natural Language Processing (NLP) tasks $x \in \mathcal{X} = \mathbb{N}^n$ may be a bag of words feature vector (where n is the total number of possible words) and $z \in \mathcal{Z} = \{0, 1\}^n$ may be a vector of 1's and 0's, which captures the presence/absence of each of the n words in a sentence. z may be viewed as meaningful "compression" of x even though $h_x(z) = x$ since h_x is a function specific to a single x.

Let $f : \mathcal{X} \to \mathbb{R}$ be the original prediction model (which usually belongs to a complex model class), such as a deep transformer neural network that performs sentiment classification of text sentences, and $g_x : \mathcal{Z} \to \mathbb{R}$ be a local explanation model which is an interpretable approximation of the original model around the neighborhood of x. Local explanation models seek to satisfy

$$z' \approx z \Rightarrow g_x(z') \approx f(h_x(z)).$$

where $x = h_x(z)$. Roughly speaking, $g_x(z)$ attempts to fit the first-order Taylor approximation of $f(h_x(z))$ around z. For $\mathcal{X} = \mathbb{R}^n$, a common choice is to have $\mathcal{Z} = \{0, 1\}^n$ and an explanation model g_x that is linear:

$$g_x(z) = \phi_0(f, x) + \sum_{i=1}^{n} \phi_i(f, x) z_i.$$

Intuitively, z is an indicator vector which only captures the presence/absence of features and $\phi_i(f, x)$ is a measure of how much the inclusion of feature i leads to an increase in $f(x)$, and hence is representative of the importance of feature i in the prediction $f(x)$.

One might wonder if such a linear explanation model g_x is uniquely identifiable given f and h_x. An astonishing result from Cooperative Game Theory is that a linear local explanation model g_x which satisfies Local accuracy, Missingness, and Consistency (see Section 3 of the original SHAP paper for more details) is uniquely identifiable given the original predictor f and mapping function h_x. While detailing these three properties are out of scope of this assignment, it suffices to understand them as highly desirable properties that we would want our explanation model g_x to have. The linear coefficients of the unique solution are called *Shapley Values* and are expressed as

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

where $h_x(z) = x$, $z' \subseteq z$ represents all z' vectors where the non-zero dimensions are a subset of the non-zero dimensions in z, $|z'|$ is the number of non-zero entries in z', i.e the number of included features, and $z' \setminus i$ denotes setting $z'_i = 0$. (see Theorem 1 of the original SHAP paper) Again recall that ϕ_i is meant to capture the importance of feature i. One useful intuition captured by the equation for the Shapley Values is that *a feature is important if it contains necessary information about making good predictions for x and there are no other redundant/correlated features that contain the same information.* In other words, having access to the value of feature i is the only way f can make a good prediction for x. For such features, the prediction gap $f(h_x(z')) - f(h_x(z' \setminus i))$ will be large no matter what other features the original predictor has access to in z'. Accordingly, most elements of the sum in the equation above will be large making the Shapley Value high.

In practice, exact computation of Shapley Values are hard as they involve summations over sets of all possible subsets ($z' \subseteq z$) which grow exponentially with n. Various versions of SHapley Additive exPlanation (SHAP) values may be understood as feasible approximations to the Shapley Values. A commonly used template for approximation is the following:

$$\phi_0(f,\bar{x}) = \mathbb{E}_{x\sim p(x)}[f(x)] \tag{1}$$

$$\phi_i(f,\bar{x}) = \mathbb{E}_{x\sim p(x|x_i=\bar{x}_i)}[f(x)] - \mathbb{E}_{x\sim p(x)}[f(x)] \quad \text{for } i = 1, ..., n, \tag{2}$$

where $p(x)$ is the data distribution. $\phi_0(f,x)$ is often called the *base value*. Intuitively, $\phi_i(f,x)$ is a measure of how much the prediction $f(x)$ changes when you fix feature i to be $x_i = \bar{x}_i$ (the observed value for the ith feature *for the given instance*), and thus once again measures how impactful feature i is in making a prediction for the given instance.

Problem 1. Linear SHAP [20 points] Let our original prediction model itself be a linear model

$$f_\theta(x) = \theta_0 + \sum_{i=1}^{n} \theta_i x_i$$

Further assume that we have feature independence so that all dimensions of x are independent. Show that equation 1 and equation 2 can be rewritten as

$$\phi_0(f,\bar{x}) = f(\mathbb{E}_{x\sim p(x)}[x]) \tag{3}$$

$$\phi_i(f,\bar{x}) = \theta_i(\bar{x}_i - \mathbb{E}_{x\sim p(x)}[x_i]) \text{ for } i = 1, ..., n \tag{4}$$

Problem 2. Implementing Linear SHAP for movie review sentiment analysis [45 points] Using the Large Movie Review Dataset, which contains 25,000 IMDB movie reviews, you will be asked to build a classifier to categorize a movie review as either positive or negative. Next, you will compute SHAP values to understand the contributions of a subset of features in your model.

The dataset has been loaded for you in the provided starter code. The inputs x have been transformed to a normalized bag of words representations of movie review sentences (using the TF-IDF representation), and targets y represent binary category labels "positive" (True) and "negative" (False), which have been decided on by the dataset authors.

Recall that a logistic regression model for binary classification is a parameterized function that outputs the Bernoulli probability

$$\frac{1}{1+e^{f_\theta(x)}},$$

where $f_\theta(x) = \theta_0 + \sum_{i=1}^{n} \theta_i x_i$ is a linear model.

(a). [**5 points**] Print the first 10 reviews in the test set, and examine the corresponding labels. Next, examine the `vectorizer.vocabulary_` dictionary, which provides mapping from words present in each review to indices in the vectorized representations. We will use this vectorized representation for training our sentiment analysis model. Select any 10 words which appear in the first sentence and which you'll be interested in examining SHAP values for. Then, select their indices, and save them into an `idxs` variable.

(b). [**5 points**] Fit a logistic model on *the training set* using `sklearn.linear_model.LogisticRegression` with arguments `penalty='l2', C=0.1`. Report the accuracy on the *test set* using the `.score()` method.

(c). [**20 points**] Compute the Linear SHAP value for the ten features you've selected in part (a) of the first test input `X_test[0, idxs]` using equation 3 and equation 4. To estimate the expectations $\mathbb{E}_{x\sim p(x)}[x], \mathbb{E}_{x\sim p(x)}[x_i]$, use the empirical mean of the feature vectors in the *training set*. (*Hint: you should be computing a total of 11 values = 1 base value + 10 for each feature*). Print the 10 words with their corresponding Shapley values and interpret your findings (2-3 sentences).

(d). [**15 points**] Now we use a standardized library to compute SHAP values. Use `shap.LinearExplainer` to compute the SHAP values. For the `model` argument, pass in our fitted logistic model. For the `masker` argument, pass in a tuple `(mean, cov)`. Here, `mean` is the mean of the feature vectors in the training set we used before. For `cov` you may simply pass in a dummy value 0 as we are assuming feature independence. Report the SHAP values for the features you picked in part (a), i.e., `X_test[0, idxs]` as we did in the previous problem using the `shap_values()` method. How do they compare to your manually computed SHAP values from part (c)?

Problem 3. SHAP with ConvNets [20 points] This time our original model will be a ConvNet trained for MNIST image classification. Code for training the model is already provided in the colab notebook.

(a). [**5 points**] Use `shap.DeepExplainer` with arguments `model=model, data=background`. Compute the SHAP values on `test_images` (defined in starter code) using `.shap_values()` method of the explainer.

(b). [**10 points**] Visualize the explanations using `shap.image_plot`. Attach the visualizations in your write-up (*Hint: you may need to use `np.swapaxes` or `np.moveaxis` functions in `numpy` to transpose your image dimensions to get proper visualizations*).

(c). [**5 points**] In the visualized example, compare the labels assigned to MNIST images to labels predicted by the model, and discuss your observations.