Explaining AuctionGym Policies with Gradient Based Attribution Algorithms

Miltiadis Chrysopoulos

Master of Science

Data Science, Technology, and Innovation

School of Informatics

University of Edinburgh

2023

Abstract

This thesis revolves around the application of algorithms from the field of Explainable Artificial Intelligence(XAI) on the trainable models of the AuctionGym simulation environment, with the ultimate goal of discovering the extend to which these algorithms can provide meaningful insights about the bidding policies of these agents. More specifically, we apply DeepLift on increasingly more complex instances of the AuctionGym environment and then empirically evaluate the quality of the results with respect to the ground truth of the simulation as well as other quantifiable metrics that we consider suitable for this problem setting.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Miltiadis Chrysopoulos*)

Acknowledgements

The completion of this Dissertation signals the end of my Master's degree studies in Data Science at the University of Edinburgh. This has been a challenging but also very rewarding year. This project fits in this description as well. It combined several domains which were unknown to me and I was excited to explore. I want to thank my supervisor Ben Allison who guided me during the initial steps of this project and helped me set clear objectives for the successful completion of this task. Also, I want to thank Robert Hu and Doudou Tang who supervised my progress throughout this project, provided constant support and ensured that after every meeting any questions I had were answered and any confusion was clarified.

Table of Contents

Chapter 1

Introduction

1.1 Purpose & Motivation

The authors of AuctionGym [13] have framed the problem of learning to bid in online auction scenarios as a contextual bandit problem. Additionally they have created an open source, configurable simulator of online advertising auctions. In that way, they provide a feasible way to frame and tackle the bidding problem as a reinforcement learning problem, without the insurmountable business cost it would involve to execute such a training regime in real world settings. Finally, they also propose different estimators to tackle the learning problem and provide empirical results that show that their *Doubly Robust Estimator* increases the overall profit of the participants when all of them train using this estimator.

Empirical results that indicate performance gains are common in novel applications that utilize Artificial Neural Networks. A common limitation however, that inhibits the adoption of these models, is their black box nature that provides little if any insights as to how they make their decisions. One way to alleviate this problem and enhance trust on these models, is to use explainability methods to derive insights about their decision making process. Unfortunately, interpretability introduces an inevitable trade-off. An inherently explainable model, has limited expressive power which undermines the original goal of applying an Artificial Neural Network in the first place. On the other hand, when trying to explain a complex non-linear model, a set of assumptions need to be made about its internal function. The extend to which these assumption hold for a given model, determine the quality of these explanations. As a result, when we apply an explainability algorithm, the question of whether we can trust its results persists. On top of that, evaluating explanation results is still an open research question.

In this project, we will apply explainability algorithms to the models of the AuctionGym environment and investigate the quality of these explanations as the parameters of the simulation change. We will focus on attribution methods and more specifically on gradient based algorithms. The motivation behind this is that these methods display superior computational performance which is a desirable property for any practical application. Additionally, we will define a set of evaluation metrics that we believe are suitable for this specific problem setting. Our main focus is to explore how the quality of the explanations changes, as we increase the dimensionality of the input data and the competitiveness of the simulation, so that the simulated scenarios progressively emulate realistic scenarios more closely.

1.2 Research Hypothesis

The main hypothesis of our work is the fact that gradient based explanation algorithms are computationally efficient. This means that they should remain computationally tractable while we increase the dimensionality and the size of the input dataset. Since we are able to utilize larger volumes of data we expect that we can derive more insightful explanations and preserve a satisfactory level of quality as we increase the complexity of the simulation. Although gradient based algorithms may violate some theoretical guarantees about the quality of their results, we believe that their computational tractability and scalability, are both very desirable properties that are worth exploring.

As we will explain in more detail in the following sections, the original models of the simulation are not compatible with Gradient Based algorithms. In order to use Gradient Based methods, we must use models that approximate the behaviour of the original models and use them to derive our explanations. In that sense, another hypothesis is that our assumptions are reasonable enough that they do not distort the explanations to the extend that they are uninformative.

1.3 Results Summary

In our experiments we first investigate how three popular Gradient-Based Algorithms scale with respect to the input dimension and input size. We find that all three methods are almost invariant to the dimension of the input for the sizes we tested. However, only DeepLift scales reasonably with respect to the volume of the input data. More specifically even for datasets of 50000 data points it needs less than one minute to

calculate attributions when run on CPUs. For comparison Integrated Gradients needs approximately 60-70 minutes for datasets of 10000 data points, using the same infrastructure. Moreover, DeepLift displays higher performance when applied on the same volume of data points.

A second series of experiments we conducted, explores how explanation quality changes when the dimension of the input data used by the models of the AuctionGym increases. Through our experiments, we identify that input dimension indeed impacts the performance of DeepLift, but even for larger dimensions the algorithm is still able to provide meaningful explanations. Another fact that is highlighted by these experiments, is that the choice of baseline value for the explanations has significant impact on the quality of the results.

Finally we explore the impact of increasing the complexity of the simulation, by increasing the number of available choices for every agent at every auction instance. This is done by increasing the catalogue of available advertisements they can choose from. We observe that even for mild increases along this dimension, the stability of the training is severely impacted so we constrain our analysis on smaller catalogues. According to our experiments, the quality of the explanations is correlated with the extend that an agent can learn a stable policy for a given set of simulation parameters.

Chapter 2

Background

2.1 Learning to Bid

In online advertising, the opportunity to display an advertisement is sold in real time. Advertisers bid on these opportunities and attempt to maximize their overall utility, which is the expected value they gain from displaying ads minus the cost they pay to participate in these auctions. The pricing mechanism can vary between second-price, first-price or any general pricing mechanism. In second-price auctions, the winner of an auction pays the second largest bid, whereas in first-price auctions the winner pays the bid they submitted.

Under second-price conditions, truthful bidding maximizes social welfare (overall net gain for the auction participants) and under certain conditions also maximizes revenue for the auctioneer. Truthful bidding is the bidding policy where a participant bids an amount equal to their expected gain from an opportunity. Nevertheless, these assumptions are frequently violated in practice, so most online advertisement auctions have adopted first-price mechanisms. In this setting, truthful bidding leads to zero expected net gain and participants need to lower their bids to generate profit. Lowering one's bids is called *bid shading*. As a result, in first-price settings, participants need to decide on a bid shading policy.

When an agent is presented with an advertising opportunity they need to make two decisions. First they need to choose an advertisement that will maximize their expected gain, assuming they win the auction. Then they need to decide on the amount they will bid for this opportunity or equivalently decide on a *shading factor* γ ($0 < \gamma < 1$) and bid an amount equal to the expected value of an ad times γ.

The authors of AuctionGym [13] propose the formulation of these two problems as

a combined contextual bandit problem. Under this problem formulation they propose a novel objective function, named *Doubly Robust Estimation*, to train the bidding policies and present empirical results that support the superiority of this method over existing approaches in literature, in terms of maximizing the social welfare of the advertisers, when all participants use the same objective function.

We now provide a prompt mathematical formulation of these two problems and a description of the proposed estimator that will be useful for future reference for the rest of this thesis.

2.1.1 The Allocation Problem

We assume that an ad opportunity is described by a context vector $x \in \mathbb{X}$. When an advertising opportunity is presented to the participants they need to decide which advertisement to display. For the scope of the thesis, we assume that each participant preserves a static catalogue of ads $\mathbb{A}$ and a static value v_a, for each ad in their catalogue. This value represents the monetary value of a conversion event C. C is a random variable which indicates if a conversion event happened after an ad was displayed and it follows a binary distribution that is parametrized by the conversion probability or *conversion throughput rate* P_{CTR} and is characteristic of every ad. A conversion event can have different interpretations within the online advertiseing setting. For example it may represent the click of a displayed ad by a user (which could be represented by lower v_a and higher P_{CTR}) or the sale of a product. The expected welfare or gain from a specific ad a_i is: $\mathbb{E}[\omega|A = a_i, X = x] = v_{a_i} * P_{CTR}(x, a_i)$. Apparently the true value of P_{CTR} is unknown and participants need to use an estimator $\hat{P}_{CTR}$ to estimate this probability. They can also introduce some noise to this estimator to enable exploration for the allocation problem. Under this formulation when an agent is presented a context vector x they choose an ad according to the rule:

$$a = \arg\max_{a \in \mathbb{A}} v_a * \hat{P}_{CTR}(x, a) \tag{2.1}$$

2.1.2 The Bidding Problem

After choosing the most promising ad to display each participant needs to decide on the bidding amount. The goal of a bidding agent is to maximize their utility $U = W(V - P)$, where W is a binary variable that denotes when an agent won an auction, V is the value of the displayed ad and P is the cost they paid to display the ad. When the agent wins an

auction all values are observable, whereas when they lose all values are 0. The bidders need to learn a bidding policy $\pi(b|X,A)$ to maximize their the expected utility. Using this policy, the expected utility is:

$$\begin{aligned}\mathbb{E}[U]_{b\sim\pi(b|A,X)} = \\ &\int P(W=1|V=v,B=b)(v-p) \\ &P(V=v|X=x,A=a)P(P=p|X=x,B=b)dvdxdp \\ &= \int P(W=1|V=v,B=b)(v-b)P(V=v|X=x,A=a)dvdx\end{aligned} \tag{2.2}$$

where in the last equation we consider the fact that in first-price auctions $P(P=p|X=x,B=b)$ is 1 when p=b and 0 elsewhere. The different bidding algorithms presented by the AuctionGym paper are essentially different methods to approximate the integral of 2.2 and minimize the difference between the estimated and observed value of this quantity.

2.2 The AuctionGym Simulation Environment

AuctionGym follows the common practice within the Reinforcement Learning community and provides a simulation environment to train and evaluate agents that participate in real time online advertisement auctions. The simulation works as follows:

- An advertisement opportunity is sampled from the data distribution $P(X)$. For the scope of this thesis we use synthetic data and experiment with data sampled from independed normal distributions.
- The opportunity is presented to a subset of the participants. Increasing the number of the bidders increases the competitiveness of the auction. We only consider simulations with two bidders chosen per auction event.
- The bidders choose an ad to display and place their bids.
- The simulation decides the winner and charges the winner according to the pricing mechanism defined. We will only consider the first-price mechanism.
- The chosen ad of the winner is displayed and a conversion event is created with probability $P_{CTR}(x,a)$.

The default mechanism of AuctionGym, that dictates how a conversion event is sampled, works as described here. At the beginning of the simulation a set of parameters $\theta_{\alpha,i} = (\phi_{\alpha,i}, \beta_{\alpha,i})$ is created, where $\phi_{\alpha,i} \in \mathbb{R}^D, \theta_{\alpha,i} \in \mathbb{R}$, D is the dimension of the fully observable context vectors and i is the number of agents. Using those parameters, the true P_{CTR} for every advertising opportunity with context x is calculated as $f_i(x,\alpha) = \sigma(\phi_{\alpha,i} * x + \beta_{\alpha,i})$, where σ is the sigmoid. Using this probability, a conversion event c is sampled $c \sim Bernouli(f_i(x,\alpha))$. The agents only have access to a part of the original context vector $x_{[1:k]}$. This simulates the fact that bidders need to make decisions with imperfect information.

In terms of bidder configuration, the AuctionGym offers three alternatives. The *Value Based Estimator*, the *Policy Based Estimator* and the *Doubly Robust Estimator*. All three configurations support the learning of a stochastic policy in the form of a Gaussian with parameters that depend on the context and ad selection. The value based method also supports the learning of a deterministic policy by approximating only the expected utility and then performing a discretized search over the bid space during inference time.

2.2.1 The Doubly Robust Estimator

We will now provide some additional implementation details for the Doubly Robust Estimator because we only focus our explainability experiments on models trained with this objective function. In terms of implementation, this method combines the elements of the other two, so extending our experiments to other methods is straightforward.

When the agents receive a context vector x, they first choose an ad to display according to the rule 2.1. For $\hat{P}_{CTR}$ every agent trains a Bayesian logistic regression. They also use *Thomson Sampling* [6] to enable exploration for the allocation step. Thomson Sampling adds Gaussian noise to the parameters of the regression, to enable occasional sub-optimal exploratory decisions. For the rest of the thesis we will refer to the part of the agent that handles the decision of the displayed advertisement, as the allocation model. After each agent chooses the ad with maximum expected welfare $v_a * \hat{P}_{CTR}$, the allocation model outputs the value v_a and the deterministic $\hat{P}_{CTRdeterministic}$. The deterministic value is the estimation without the noise of Thomson sampling. After the allocation decision is made, the policy network receives these two values as inputs and produces two output values. These values correspond to the mean and standard deviation of a Gaussian distribution. This distribution is the stochastic policy of the

shading factor for the given context and ad selection. Then a shading factor γ is sampled from the distribution and the bid $b = v_a * \gamma$ is placed. We also note that the standard deviation is kept over a minimum threshold to prevent the policy from collapsing to a deterministic sub-optimal policy.

2.3 Explainable Artificial Intelligence (XAI)

Although Artificial Intelligence models have achieved state of the art performance in various tasks, their ”black-box” nature still remains a notorious problem. There are multiple shortcomings whose root cause is this inherent nature of these models. First of all, there are multiple cases where these models extract unintended patterns from data and erroneously base their decisions on them. Such behaviour is very difficult for practitioners to identify before deployment. Especially when such models are intended for critical applications, for example healthcare or finance, then establishing trust on the decision making process of these models is equally if not more important than the performance gain they achieve. This justifies why, in many cases practitioners from other domains are reluctant to utilize Artificial Intelligence model despite their enhanced performance. If we also consider the fact that regulations around data manipulation are becoming increasingly strict [10], then finding ways to accurately describe the decision making of any model is of paramount importance and beneficial both for the development and the adoption of them.

Explaining Artificial Intelligence models, while preserving their ability to identify complex patterns in the data are two competing goals because their highly non linear nature is the source of their expressive power but simultaneously makes them non-transparent. Inevitably, we need to compromise either performance or quality of explanations. One approach is to use less expressive models that are inherently interpretable such as decision trees or random forests[7][5][18]. Alternatively we can use post-hoc approximations to attempt to explain the original model. This can be achieved by fitting a surrogate model to mimic the behaviour of the original. This model is either interpretable as the aforementioned ones or at least more amenable to explanation [8]. Alternatively, we can introduce some simplifying assumptions about the original model and attempt to explain it. Several popular alternatives exist under the latter framework, such as LIME [19], SHAP and its variants [16][22] and Gradient-Based Attribution methods [20][21][3][4]. All these methods try to approximate the model locally with a linear approximation. By interpreting the weights of the linear model

we can reason about the relative importance of the individual features on the original model's output. All methods come with their individual advantages and disadvantages in terms of accuracy, computational tractability and whether they can interpret any source model or they need to impose structural constraints to it. All these parameters need to be taken into consideration when choosing the best explanation algorithm for a specific application.

2.3.1 SHAP Value Attribution based Methods

SHAP based methods are named after **Shap**ley values [15] and they define the class of *additive feature attributions*. They prove that any algorithm in this class has a unique local solution that satisfies three desirable axioms (local accuracy, missingness, consistency). However, a model agnostic implementation of the algorithm has exponential complexity with respect to the feature dimension. If we enforce some structural constraints to the explained model, then computation becomes tractable [20][3][22].

2.3.2 Gradient based Attribution Methods

Gradient based algorithms are only pertinent to Neural Network models. They are based on the idea that the attributions of the input features can be backpropagated, by calculating the difference between the model's output and some reference output value and recursively attributing this difference through the network's layers. The most attractive property of these methods is that, since they are compatible with neural network implementations, they can leverage the algorithmic and hardware acceleration available for these models. This makes them in practice more suitable for models with large input dimension. Additional analysis on these methods [1] shows that under certain structural constraints of the original model, some of these methods may or may not satisfy some of the axioms of the additive attribution methods.

2.3.2.1 DeepLift

DeepLift [20] is a gradient based method that is based on the idea that if we define a reference input, then for any input example we can calculate attributions such that they satisfy the *summation to delta rule*: $\sum_{i=1}^{n} C_{\Delta_{x_i}} = \Delta_y$ where C_i is the attribution weight for feature i. This rule is equivalent to the local accuracy axiom of SHAP methods. Although there are two approximation rules to calculate the attributions in the original

paper, the original implementation only supports Keras models (it is the only one that supports both approximation rules), so we will only describe the Rescale Rule that is supported by all implementations except the original. For the non-linear layers, this rule approximates the attributions with $\frac{\Delta_y}{\Delta_x}$ or the gradient when $\Delta_x \to 0$, where Δ_x, Δ_y are the difference from the reference input and output accordingly. For linear layers the attribution rule is straightforward to derive from the summation to delta rule. This fast approximation is the source of the computational efficiency of DeepLift. However, this comes at a cost of being more restrictive on the nonlinear layers it supports. For example, the original implementation only supports layers that occur in Convolutional Neural Networks. Subsequent implementations extend the Rescale Rule for most non linear layers but still the calculation used for the attributions may not always be a good approximation. *DeepSHAP* is a variant of DeepLift where the attributions are calculated for different baselines and they are averaged on every layer before they are backpropagated to the previous layer. This variant approximates the Shapley values [16] but lacks in terms of computational efficiency. Since the attributions must be averaged on every layer before being transmitted to the previous, batched calculations can only be performed for multiple baselines. When the baselines are significantly less than the number of input examples, calculations become very inefficient.

2.3.2.2 IntegratedGradients

IntegratedGradients [21] is another popular Gradient Based explainability method that was published at a similar time as DeepLift. This method attempts to approximate the attribution weights by calculating the integral $IG_i(x) = \int \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} \partial \alpha$, where x' is the baseline input, for every dimension i. Integrated Gradients is much more computationally intensive, because for a single example, the integral needs several forward passes per input dimension to be approximated. On the other hand, this method does not impose any constraints on the original model, as long as the operations used are differentiable and thus produce gradient signal.

2.3.3 Evaluating Explanations

A common problem in the research field of XAI is the lack of generally applicable evaluation metrics. Even the notion of a "good" explanation is under debate. There are two popular criteria to evaluate explanations on, *plausibility* and *faithfulness* [11]. Plausibility refers to how much the explanations are acceptable by humans. Faithful-

ness measure how accurately the explanations describe the behaviour of the model. Potentially these two metrics might vary for the same model. Faithful explanations can be unintuitive and viseversa. A very popular example of a plausibility evaluation method is a salience map. The problem with such evaluation methods, is that they need human feedback which is costly. Moreover, such an evaluation procedure is unsuitable for systematic evaluation. On the other hand, faithfulness is more suitable for systematic evaluation but it is still an open question as to what is an effective and generally applicable faithfulness metric. When dealing with synthetic data, a possible choice is to compare the explanation results with some pre-established ground truth. When we are dealing with non synthetic data, the most common approach is to use some form of perturbation test [9] [20][17][2]. However, perturbation tests have their own shortcomings because changing features may create examples that are out of distribution and might provide misleading results.

Chapter 3

Conceptual Design and Implementation

3.1 Evaluation Metrics

As we mentioned in section 2.3.3, there are no general purpose evaluation metrics for XAI like accuracy or f1 score are, in supervised learning. Nevertheless, we need a systematic method to evaluate the results of our explanation algorithms. For this reason, we need to define a set of metrics that are suitable for the specific task at hand. We are mainly interested in explaining the bidding policy of the trained agents, so we certainly need a metric that quantifies this. If we trained our agents with real world data or in an actual online bidding setting, then we could argue that the aforementioned metric would be the only one we are interested in. However, within the simulation setting we have additional information about the generative process of the simulation data. We can leverage this information to reason about the quality of the explanations. Following this thought process, we propose that the evaluation of the explanations should be performed on two fronts. First, on the model that tackles the allocation problem and then on the ”end-to-end” model that receives as input the context of a given advertisement opportunity and produces as output the bid.

For the first evaluation task, we can use the generative process of the simulation to make a direct comparison between the features that the explanation algorithm identifies as important and the features that are actually important for a given simulation instance. When solving the allocation problem, the agents estimate P_{CTR} while having access only to a part of the context vector $x_{[1:k]}$. If the agents can effectively approximate the true CTR function and the explanation algorithm is able to produce meaningful explanations, then there should be an alignment between the ground truth vectors $\phi_{\alpha,i}$ and the importance weights generated by running an explanation algorithm on the model

used to approximate the CTR.

One way to capture this alignment, is by calculating the cosine similarity between $\phi_{\alpha,i_{[1:k]}}$ and the average of the normalized importance weight vectors that are produced by the explanation algorithm. The cosine similarity, is defined as: $S_C(a,b) = \frac{a \cdot b}{\|a\|\|b\|}$

The intuition behind this is that if we consider the average attribution weights for every item as a vector, then if the relative magnitude of these weights is similar or identical to the relative magnitude of the dimensions of $\phi_{\alpha,i_{[1:k]}}$ (as it should be for a well performing explanation method), then these two vectors should point to a similar direction, driving the cosine similarity closer to 1. However, as shown in fig. 3.1, this metric may provide misleading insights, where the vectors do not have the described property but cosine similarity is high.

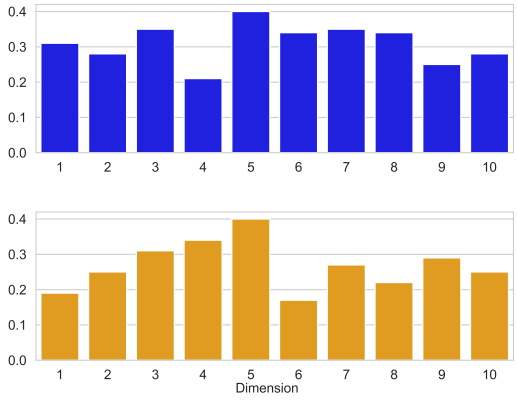

Figure 3.1: An example of two random vectors where cosine similarity provides misleading results. The cosine similarity of these two vectors is 0.959. The total ordering metric for these vectors is 0.2.

A better alternative is to use a metric that captures if and by how much, the sorted order of the importance weights corresponds to the sorted order of the ground truth vectors. The idea behind this is that the actual value of the importance weights is informative only locally(attribution weights should satisfy the summation to delta rule, section 2.3.2.1). The information that should be more globally applicable, is the relative importance of the weights. Assuming we have a ground truth vector a and a vector we want to test b, we define the *total ordering metric* as:

$$S_{TO} = \frac{\sum_{i=0}^{k} argsort(a)[i] == argsort(b)[i]}{k} \tag{3.1}$$

where k is the dimension of a and b. Algorithm 1 presents the pseudo code for the total ordering metric.

We need to note at this point that the complexity of this evaluation task scales linearly with the number of classes or the total ads in the catalogue of an agent, because we need to calculate and compare the attributions of each individual class with the simulation ground truth. Since the catalogue in realistic scenarios is much larger than the dimension of the data, it dominates the computational complexity and we would have to either sample classes to perform this test or have some sort of additional criterion to choose which classes are the most informative to test for.

Algorithm 1 Total Ordering Algorithm for vectors a and b

$arga \leftarrow argsort(a)$
$argb \leftarrow argsort(b)$
$c \leftarrow 0$
for $i = 0; i < len(arga)$ **do**
  if $arga[i] == argb[i]$ **then**
    $c \leftarrow c + 1$
  end if
end for
return $c/len(a)$

In the case of the ”end-to-end” model there is no ground truth that we can use to compare the explanation results to. For this reason, we will attempt to evaluate the explanations in terms of faithfulness to the explained model. To achieve this, we will adopt a test that is especially popular for evaluating faithfulness in NLP related tasks, the perturbation test. In such a test, we measure the change in the model's output when we perturb a part of the input. The idea is that the more important a feature is, the more significant the change of the model's output should be, if we alter the value of this feature. The perturbation can vary from zeroing the feature or scaling it by constant factor. By performing the perturbation on every dimension of the input, one at a time, we can get a vector that captures how significant the distortion of the output was when the selected feature was perturbed. We note, that since we are interested in deriving and testing global explanations, we will aggregate the results of the perturbation test, by averaging distortions per dimension.

Algorithm 2 presents the pseudo code for generating the perturbation vector. After we have generated the perturbation test output, we can test the level of agreement

between the relative magnitude of the importance weights and the magnitude of distortion of the output. For this purpose, we can use the total ordering metric we defined, between the attribution weights and the perturbation vector. Alternatively, we can also measure the top-k recall between these two vectors. Top k recall is defined as: $S_{Top-K} = \frac{TruePositives}{TruePositives+FalseNegatives}$
We treat the perturbation vector as the ground truth, so true positives are considered the features that are in the top-k most important positions of both the perturbation and the attribution vector and as false negatives the ones that only appear in the perturbation vector. The motivation behind this is that several features with similar attribution weights might result to output changes that are marginally different. In those cases the total ordering metric might be unnecessarily penalizing and top-k recall can help capture that. We will only constrain our test on perturbing one feature at a time to avoid out of distribution samples.

Algorithm 2 Perturbation vector for input example x and function f

$out \leftarrow []$
for $i = 0; i < len(x)$ **do**
  $x' \leftarrow x$
  $x'[i] = perturbation * x'[i]$
  $out[i] = f(x) - f(x')$

end for
return *out*

3.2 Processing the Attributions

In order to derive meaningful results from explanation algorithms, we need to be aware of what the explanations actually represent and consequently how we need to process them. All gradient based explanation algorithms generate local explanations. However, local explanations are of little use if the examined data point has no special importance. Especially in our case, where input examples are synthetic, local explanations are even less useful. We are more interested in producing a global explanation of the original model or at least a satisfactory approximation of it. This implies that we need to perform some form of aggregation of the local explanations. If we naively average them, then we will get misleading results. The reason is that for every input point x

used to calculate attributions, then it should be true that $\sum_{i=1}^{n} C_i \simeq f(x) - f(b)$ where i is the input dimension, f is the original model and b is the baseline reference value (assuming that the algorithm we use does not violate the local accuracy axiom). For points where the difference from the reference $f(x) - f(b)$ is large, the magnitude of all attributions will also be larger compared to points with small $f(x) - f(b)$. In that way we are weighting features that deviate a lot from the baseline more. But the numerical values of the attributions are only meaningful locally, so in order to get as much of an accurate global interpretation as possible, we must normalize the attribution vectors per example so that all examples have equal contribution. Otherwise the relative importance of the features over multiple examples will be distorted in favor of the examples that lead to larger deviation from the baseline output.

A second form of processing we need to do is due to a difference between the original DeepLift [20] paper and the Captum [14] library we use for our experiments. More specifically, the implementation we use is based on the proposed global attribution method in Table 1 of [1]. In this version, the attributions are multiplied by the difference $x - b$. The motivation behind this is that in order to get a proper insight about the marginal impact of a feature, then its average magnitude needs to also be taken into consideration. Although it is out of scope to verify this claim, section 3.2 of [1] provides an illustrative example.

In order to get meaningful insights from our aggregation when using the above implementation, we must use the absolute values of the attributions. Obviously, we lose information, concerning the positive or negative impact of a feature, but overall it helps significantly to capture the relative magnitude of importance between features. In an attempt to corroborate our choice, we provide an example that highlights why using the absolute value is important. Assuming we want to explain a linear model $\sum_{i=1}^{n} a_i x_i$. Let us assume that our dataset consists of two examples and we only consider 1 dimension j. Feature 1 has value v and feature 2 has value $-v+c$ where c is a constant. Then the average attribution for this dimension would be $\frac{(v-b)a_j+(-v+c-b)a_j}{2} = \frac{(c-b)a_j}{2}$. When $c-b$ is small, then the attributions diminish. Also, this obscures the impact of co factoring the magnitude of the input, as we originally intended by using the global attribution method. In our experiments, where the sampling distribution of every input dimension is a normal distribution, and for any data distribution that is symmetric, this is problematic because input pairs like the one we described are common and most attributions cancel out. We also note that since we make this processing choice we need to perform a similar processing to the perturbation vector. More specifically, instead

of averaging the distortions per dimension we will average the absolute values of the distortions.

3.3 Challenges of Applying Gradient based Methods

Gradient based attribution methods have the attractive property of being more computationally efficient compared to SHAP value based methods. This has several practical implications. First of all, these methods can be applied on datasets with large dimensionality, because the attribution weights are calculated using the efficient implementations of the explained neural model. For the same reason, we can apply them on larger datasets, which also improves the quality of the explanations. Nevertheless there are several limitations that we need to account for when we try to apply such methods to our problem setting.

The first problem is that in reality the bidding agents need to make two decisions in order to produce a bidding value. In] terms of implementation, this means that they are using two different estimators. The first estimates the P_{CTR} and tackles the allocation problem and the second estimates the parameters of the bidding policy. However, we are interested in producing a unified explanation for the end-to-end model. With model agnostic explanation methods this is not problematic but gradient based approaches are inherently designed to produce explanations for a single neural network. This limitation by itself is relatively easy to overcome by combining the two trained models into a single PyTorch model to be explained.

A second problem that is harder to overcome is the fact that gradient based methods can only be used by models that use differentiable functions. As we discussed, the allocation model uses the decision rule 2.1 that uses an *argmax* operator. This operator is not differentiable, so the gradients for this operator are 0. If we naively combine the two aforementioned estimators then any gradient based explanation algorithm will not be able to produce results. An additional complication is the fact that the allocation model actually uses two slightly different estimators, one stochastic to decide on the selected advertisement and one deterministic to calculate the P_{CTR} value that is propagated to the policy network. This enables exploration for the allocation problem but is problematic for our case because such logic is even harder to transform into a sequence of differentiable operators.

Finally, one last problem we need to take into consideration, is the choice of the baseline values we use to generate our explanations. As mentioned in Section 3.3 of the

DeepLift paper [20], the choice of baseline is crucial for the quality of the explanations. Given that we are using synthetic data in our problem setting, we have access to the true mean value of the input data based on the generative process we sample from. We need to highlight however, that the choice of the baseline value indeed impacts the performance of the DeepLift algorithm, as we verified empirically. When trying to apply the same algorithm to any dataset whose true distribution is unknown (as is the case for most if not all non-synthetic data), then choosing an informative baseline value is a non-trivial task.

3.4 Proposed Solutions

The most foundational problem we need to tackle, in order to use gradient based approaches, is the propagation of gradients between the allocation and the policy model. To achieve this we need to make two simplifying assumptions about the model we will use to generate explanations. The first is that we will explain only the deterministic elements of the allocation model. This means that we will only use the deterministic estimator of the allocation model both for deciding on the optimal ad to display and for calculating the $\hat{P}_{CTR}$ that is propagated to the policy network. Adopting such an approach is reasonable as long as the stochastic estimator leads to the same decision as the deterministic one in most cases. As we can see in table 3.1 the deterministic policy deviates from the stochastic policy less than 4% of the times in the dataset we use to generate our explanations, so this is a reasonable approach. Moreover, it is reasonable to expect that at the end of the training, the agents will choose the deterministic and optimal option almost always. It is obvious though, that such an approach would be unsuitable if we were interested in generating explanations throughout the training of the agents.

An additional simplification we attempted, was to change the target of our explanations. At its original form, the policy network produces two output values, the mean and standard deviation of the Gaussian that forms the bidding policy for that given context vector and then a value is sampled from this distribution. However, by examining table 3.2 we observe that the generated standard deviation is always about 1%. Additionally, if we examine the implementation of the Doubly Robust Estimator, we can see that the actual output of the policy network is offset by an additional 0.01. This means that the actual output of the policy network is at least one order of magnitude lower than 0.01. As a result, we can neglect this part of the output without losing signif-

icant information about the input attributions. By doing that we can instead generate explanations using as target the bids when always choosing the mean shading value of the bidding policy instead of the actual predicted bidding value. We expected that this would remove the noise that is introduced by the sampling of the shading factor and improve the quality of the explanations. However, explaining the stochastic bidding policy directly led to better empirical results.

Emb.Sz	Avg
5	$4.1\% \pm 0.03$
10	$3.0\% \pm 0.01$
20	$2.9\% \pm 0.00$
40	$3.52\% \pm 0.00$
50	$3.81\% \pm 0.00$

Table 3.1: The percent of times where the stochastic allocation estimator results in a different decision than the deterministic estimator.

Emb.Sz	Avg	Min	Max
5	0.0102	0.0100	0.0107
10	0.0105	0.0100	0.0105
20	0.0100	0.0100	0.0112
40	0.0101	0.0100	0.0119
50	0.0101	0.0100	0.0134

Table 3.2: Indicative values of the predicted standard deviation of the bidding policy by the policy network.

Although focusing on the deterministic components of the bidding policy can be an effective proxy to help us explain the original model, it still does not solve the use of the argmax operator. To overcome this difficulty there are several alternatives. The first is to train a new model that imitates the output of the original allocation model but only uses differentiable operators. For example this could be a regressor that takes as input the context vector and potentially also the values v_{a_i} and produces two outputs, the $\hat{P}_{CTR}$ of the original model and the corresponding value. We chose not to adopt this solution. The reason is that, training a new model is a non trivial task, which means that for every new simulation setting we want to analyze, we also need to train an additional model, potentially with different hyperparameter settings. This makes it harder to automate the process. Also as the simulation becomes more complex,

approximating the original model also becomes a harder task, which implies that the approximation will progressively not imitate the original allocator closely. This will introduce additional error that will degrade the quality of the explanations. Finally, the highly non-linear nature of the argmax operator is unlikely to be well approximated by a regressor. This will lead to output values that have not been observed and should not appear. This can have unexpected results since the policy model was observing only specific values during training.

An alternative approach is to try to transform the non-differentiable operation, to a differentiable one or at least partly. Our goal is to be able to propagate the tuple ($\hat{P}_{CTR}$,v) that corresponds to the ad with maximum $\hat{P}_{CTR} * v$ for the given context. We can use the original allocation model to calculate $\mathbb{E}[\hat{\omega}] = \hat{P}_{CTR} * v$ for every ad in the catalogue of the agent. Then we can use the decision rule $\max_a \mathbb{E}[\hat{\omega}]$ to find the maximum expected welfare for the given context. If we had a way to know the value that corresponded to the maximum welfare, then our problem would be solved. We can get this value by using the original allocation rule 2.1 and returning the value that corresponds to the selected add. Then we can propagate the tuple ($\max_a \mathbb{E}[\hat{\omega}]/v$,$v$) which is almost identical to the original output of the allocation model, with the exception of a small numerical error introduced by the division operation. The difference is that we now have a sequence of differentiable operators that connects the input vectors and the bidding output. We only miss the part of the information that corresponds to the decision of the value of the ad.

A final approach we attempted, is to replace the argmax operator with a softmax. To do that, we first use the allocation estimator (we can now choose between choosing the stochastic or the deterministic one) to generate a soft mask according to the rule $mask = softmax_a \mathbb{E}[\hat{\omega}]$. We need the mask to be sparse during the forward pass, so that the output of the explained allocator, is identical to the original one. To achieve this, we use the straight-through trick similar to the one used for Straight-Through Gumbel Softmax [12], where during the forward pass we use output of $argmax_a \mathbb{E}[\hat{\omega}]$ but during the backward pass we use the gradient of the softmax operator. The idea was that with more gradient information, the results of the explanations could improve. But empirically, this approximation leads to significantly less informative explanations.

One last limitation we need to mention here, is the fact that the implementation of DeepLift that is compatible with the PyTorch library [14], only supports the Rescale version of DeepLift. For out problem setting it would be more suitable to use the RevealCancel rule that is described in section 3.5.3 of the original paper [20]. This approximation rule is supposed to perform better when assigning attribution scores to

min/max functions. However, the Rescale rule is the one that has been studied in more depth [16] [1] and will only constrain analysis in this version of the algorithm.

Chapter 4

Empirical Results and Evaluation

For all of the experiments we train the agents using the Doubly Robust Estimator. We fix the number of agents to 3 and the number of participants per auction event to 2. Although this is not directly related to the explanation problem we examine, we keep a fixed ratio of 0.2 of the context vector hidden throughout our experiments because it makes the learning problem harder and imperfect information is a characteristic property of the online advertising problem. We train the agents for 50 iterations where every iteration consists of $\Delta_r = 10000$ online auction instances. The agents only train their policies every Δ_r steps. After saving the trained models and the parameters of the auction, we generate different amounts of data using the trained models and the simulation with the same parameters used during training. Then we use these data to derive the explanation results.

4.1 Replacing argmax with a max operator

All of the experimental results presented in this section correspond to the approximation we introduced in paragraph 4 of section 3.4.

4.1.1 Comparing Different Gradient Based Methods

In this part of the analysis we will compare the performance and computational efficiency of three different Gradient Based Explanation Methods, DeepLift, Integrated Gradients, DeepSHAP. Before we use the explanation algorithms we transform the models as described in paragraph 2 and 4 of section 3.4 and we compare the results on the perturbation test for the three trained agents for different data sizes. We observe that

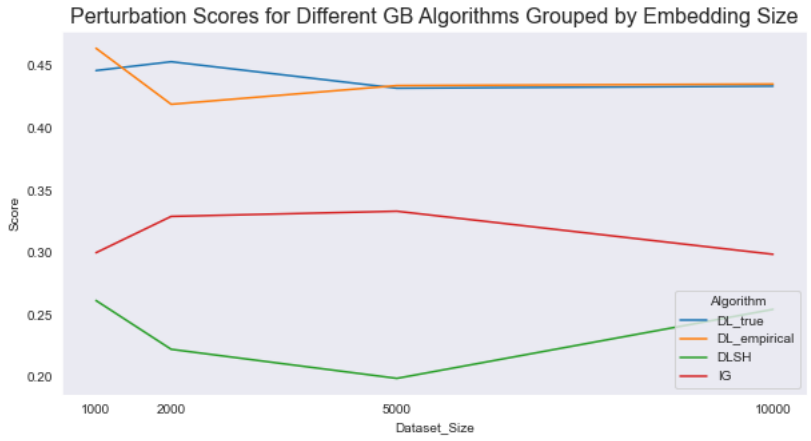

Figure 4.1: Performance of IntegratedGradients(IG), DeepLift(DL) using the empirical or the true mean of the input data as baseline and DeepLiftSHAP with 10 baselines per example. The results are aggregated by embedding sizes in [5,10,20,40,50] and for 3 trained agents per embedding size.

DeepLift displays better average performance on the perturbation test over all dataset sizes. However, the performance over experiments with different embedding sizes varies significantly, because as the input dimension increases the problem becomes more complex and the performance of the algorithms decays. However, a more significant advantage of DeepLift is that is very computationally efficient. In table 4.1 we see the CPU time for generating the explanations of 3 agent models using different algorithms over various dataset sizes. As we can see DeepLift is the only solution that scales reasonably with the size of the data processed. The reason for this is that DeepLift can fully leverage the speedup of PyTorch and calculate attributions with one forward and one backwards pass.

On the other hand Integrated Gradients needs to perform multiple calculation steps per input dimension to approximate the attributions, which impacts the computational performance over higher volumes of data and does not provide any gains in terms of quality of explanations. Finally, DeepSHAP performance varies depending on the number of baselines we use for our calculations. As shown in fig. 4.1 when using 10 inputs as baselines, performance is poor compared to the other alternatives. When using 100 computation becomes intractable and it is of little practical use. At this point we need to mention that the presented performance of DeepSHAP is potentially implementation specific up to some extend. According to the original developers of DeepLift, there are alternative implementations that might scale better when the size of

the input data is significantly larger than the baselines, but they do not provide support for the PyTorch library. Since DeepLift appears superior both in terms of performance and in terms of computational efficiency, for the rest of the experiments we constrain our analysis on the DeepLift algorithm.

Aglorithm	1000	2000	5000	10000
DeepLift	8.43s	8.46s	9.07s	10.64s
IG	59.3s	∼5min	∼15min	∼65min
DeepSHAP(10 baselines)	10.51s	16.0s	40.40s	7.2min
DeepSHAP(100 baselines)	7.1min	∼61.2min	∼390min	-

Table 4.1: CPU Time as measured at the student.compute cluster of Informatics. In cells with " " there was an inconsistency between different runs but the order of magnitude was the same.

4.1.2 Explaining the Greedy Policy

In this section we will examine the performance of DeepLift over different simulation configurations. We keep the generative process of the data to its default settings. Each feature of the context vector is sampled from a normal distribution. The coefficients of the ground truth vectors that are used to estimate P_{CTR} are also sampled from a normal distribution. Finally the values for every ad in the agents' catalogues are sampled from a lognormal distribution.

As we argued, the deterministic policy of the agents at the allocation step, is almost identical to the stochastic policy of the agents at the end of the training. For this reason, to overcome implementation difficulties, we will use the deterministic policy to generate our explanations but we will perform our perturbation test using the original model as ground truth. We also use a different dataset to calculate our explanations and a different one to calculate the perturbation vector. The motivation behind this is that when we use the same dataset for both tasks, smaller datasets display inflated performance. This is similar to using the same dataset for training and evaluation.

4.1.2.1 Increasing the Context Dimension

For this part of the experiments we change the input context dimension and explore how the quality of the explanations evolves. DeepLift needs a reference value to use

as baseline. In the plots below, we can observe that the choice of baseline actually impacts the performance of the explanations. More specifically we perform two runs of DeepLift. On the first we use the true mean of the input context as baseline and on the second the empirical mean. Apparently, access to the true mean is only available as long as we are using simulated data.

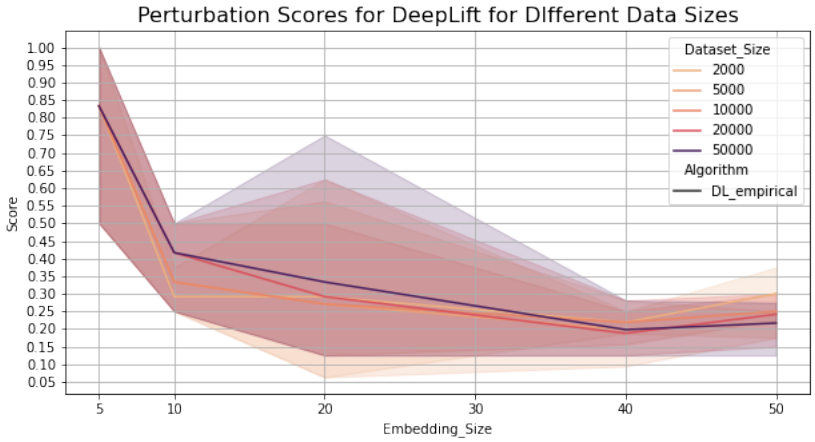

Figure 4.2: Performance of DeepLift(DL) using the empirical mean of the input data as baseline. The results are aggregated for 3 trained agents per embedding size.

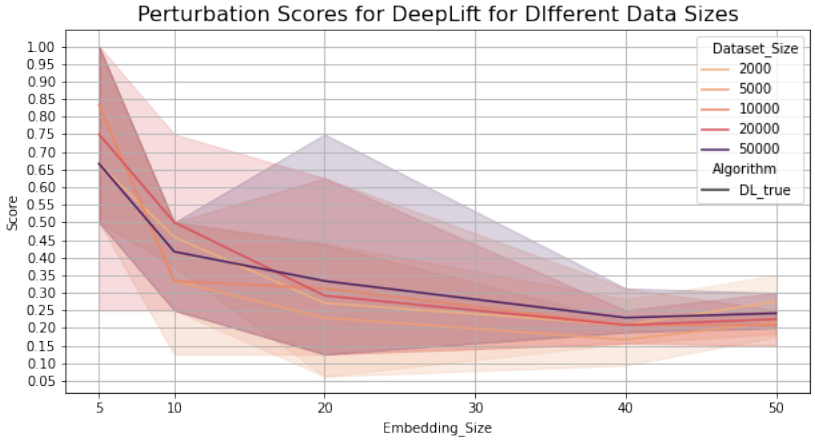

Figure 4.3: Performance of DeepLift(DL) using the true mean of the input data as baseline. The results are aggregated for 3 trained agents per embedding size.

By observing fig. 4.2 and fig. 4.3 the first thing we identify is that performance decays as we increase the dimension of the input, regardless of the input data we use for the explanations. This is expected for multiple reasons. First of all, DeepLift is originally designed to derive local explanations for a single input example. However,

we are trying to generalize explanations by aggregating over multiple examples. As we increase the dimension of the input it is less likely that a single dimension will be consistently more impactful than the rest in isolation, which introduces noise in our results. The second reason is that the perturbation test we perform, also uses aggregated information. That said, when multiple features have similar magnitudes of perturbed outputs, then they will be placed in close positions in the ground truth vector of the perturbation test. However, when their difference is marginal, we cannot be certain that the order for these features accurately captures the behaviour of the model. If we also consider the fact that the output of the original model has some inherent stochasticity due to the sampling of the shading factor, our confidence is reduced further. This phenomenon is more likely to occur in higher dimensions and might erroneously obscure the results of the perturbation test.

It is reasonable then to wonder how the algorithm performs on a more relaxed version of the perturbation test. For that reason we calculate the ground truth vector with the same method as before but now we measure only the top-k recall for different subset sizes of the input dimension. In fig. 4.4 we see that the algorithm has more consistent performance on this test. Especially as we decrease the size of the subset (which only considers the more important features), performance is very high which shows that DeepLift can scale well with the number of dimensions.

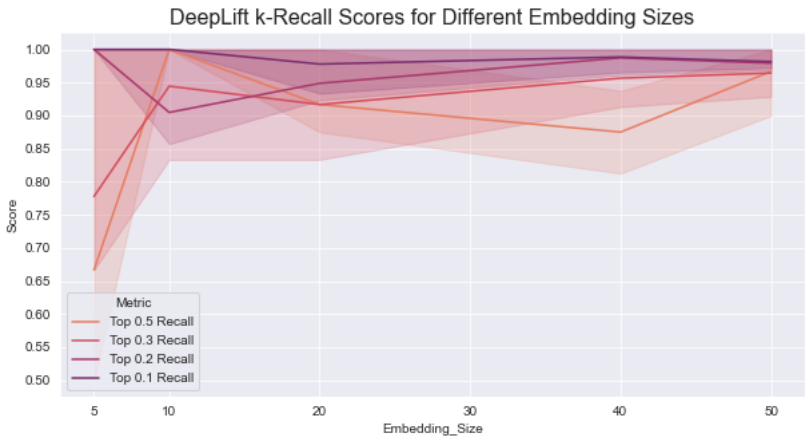

Figure 4.4: Top-k recall of DeepLift(DL) using the empirical mean of the input data as baseline for a dataset of 50000 data points. The results are aggregated for 3 trained agents per embedding size.

Another observation from fig. 4.3, fig. 4.2 that is counter intuitive is that performance does not increase significantly on average for larger datasizes, especially for larger

dimensions. It is important to note however that when using the true mean as baseline, larger datasets consistently improve the worst case performance, as we can observe by the minimum variance. This is important because the training of all agents is not symmetric especially for larger dimensions. Some agents win more often because they have more favorable auction parameters, while others win rarely. When agents win they gain more positive reinforcement which allows them to achieve better training. This also becomes more intensive in the experiments with higher dimensionality. Smaller dataset sizes are more susceptible to this variance and perform better for agents with better performance on the simulation, whereas larger dataset sizes are more robust to this fluctuation. This is an important property that we need to consider because our aim is to generate explanations for any agent regardless of the parameters that are out of its control. This observation also highlights the impact of the baseline on the quality of the results.

Due to the fact that the perturbation test becomes less reliable as the complexity of the simulation increase, it is also important to evaluate the performance of the explanations on the allocation problem in isolation. For this part of the problem we have access to actual ground truth and that can give us more reliable insights. Using DeepLift on the allocation model in isolation, we generate the explanations for each item individually and then compare its alignment with the corresponding ground truth parameters of the simulation. fig. 4.5 presents the aggregated results for all three

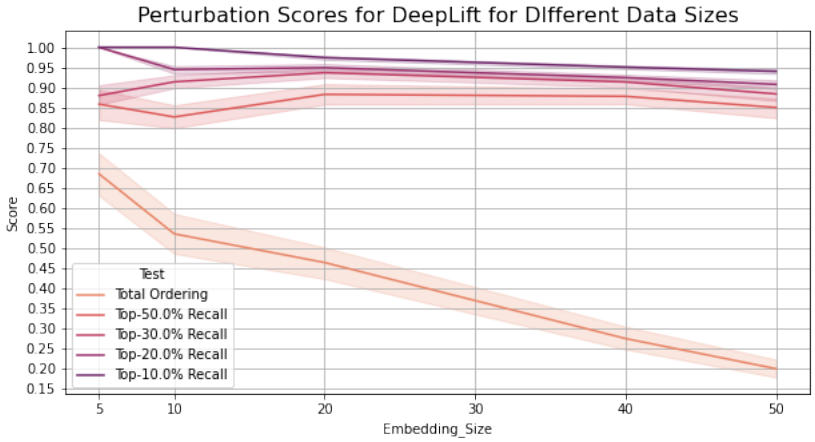

Figure 4.5: Different Metrics for the performance of DeepLift on the allocation problem. Results are aggregated over different dataset sizes and 3 agents.

agents and for different dataset sizes used to generate the explanations. We observe the same trend as in the plots of the bidding problem performance.However, the variance is

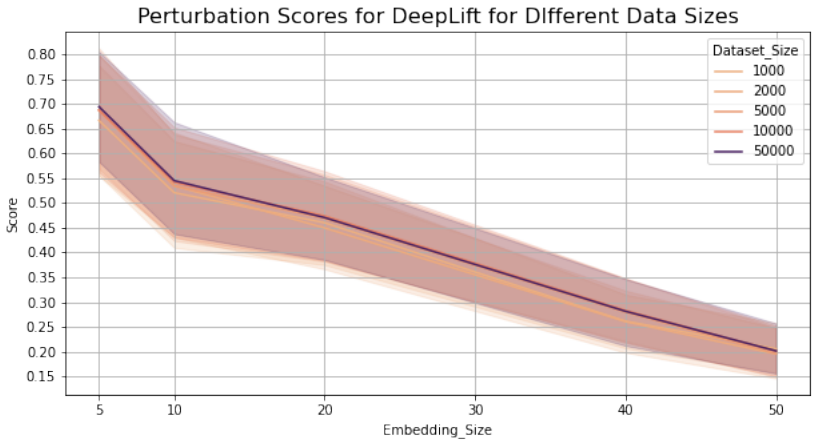

Figure 4.6: Total Ordering performance of DeepLift on the allocation problem. Results are aggregated for 3 agents.

significantly reduced compared to the bidding problem which means that DeepLift's performance is relatively invariant with respect to dataset size and examined agent. This is also made apparent by fig. 4.6 which shows that increasing the dataset only leads to marginal improvements in the total ordering metric. For the top-k recall metrics, one potential reason of this invariance in performance, is that DeepLift explanations achieve high performance, even for small datasets, so the margin of improvement is small.

We would expect however, that increasing the dataset would lead to more significant performance gains in the total ordering metric. There are multiple factors that led to this counter intuitive result. First of all, the way we process our explanations makes the problem harder. Since we are only analyzing the explanations in terms of magnitude, we need to make a comparison between the absolute values of the ground truth vectors and the explanations. This loss of sign information doubles the probability of two features being marginally different and harder to distinguish, although originally they would be easily separable because of the difference in sign. The second reason is that the allocation models of the simulation do not fit the ground truth function perfectly. As a result, the error of the original model is propagated to the explanations and since we are comparing with the ground truth we are also observing the error of the original model.

4.1.3 Increasing the Ad Catalogue Size

For this part of the experiment we investigate the impact of catalogue size on the quality of the explanations. Increasing the catalogue size directly affects the complexity of the

allocation problem because there are more choices available that must be explored and learn the parameters for them. We initially considered to keep a larger dimension of context vectors but the simulation fails to stabilize the training even for catalogues of size 30 (the default is 12). In the experiments that produce fig. 4.7, we used a Δ_r of 50000 in an attempt to stabilize training but it still fails for 2 out of 3 agents.

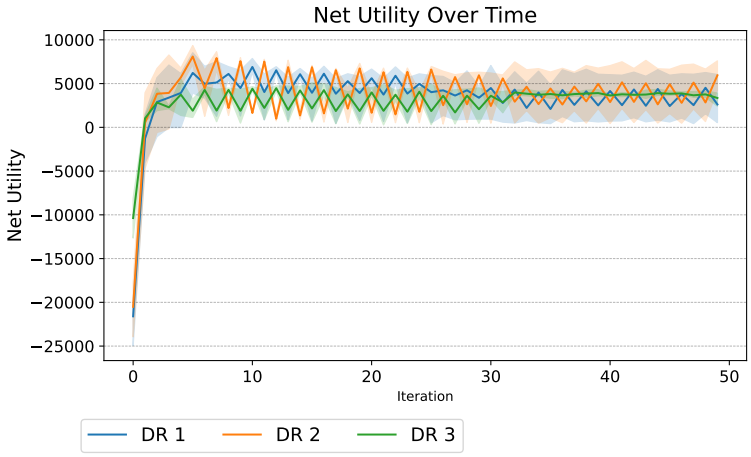

Figure 4.7: Net utility for the simulation agents for catalogue size of 30 and input dimension 40. The results are aggregated over 3 runs of the training simulation with the same simulation parameters.

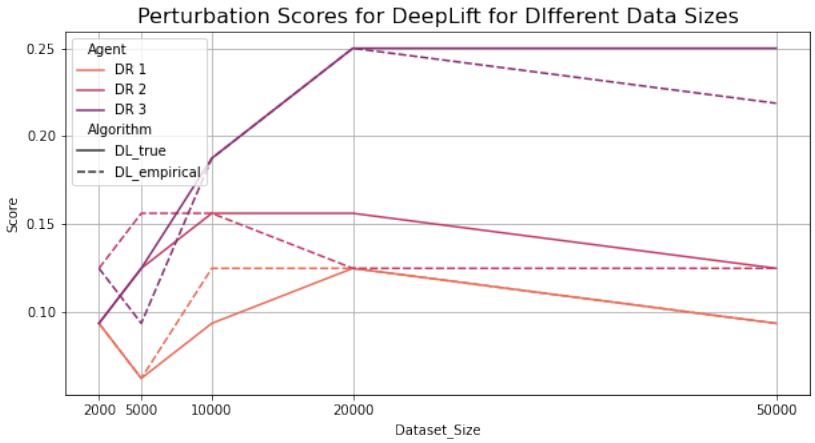

Figure 4.8: Perturbation test results for the explanations of DeepLift(total ordering). The explanations were generated using either the true or the empirical mean of the data as baseline.

By fig. 4.8 we observe that there is an apparent discrepancy in explanation quality between agent 3 and the other 2 agents. We notice that agent 3 is also the only one that

is able to converge to a stable policy during training. The perturbation test evaluates the quality in terms of faithfulness between the explanations and the original model so even if the policy is not performing well, the explanations could potentially be faithful to the model. One potential reason for the observed results is that the agents with unstable policies preserve a significant stochastic component in their decision pipeline, which makes the explanation algorithm unable to correctly assign attributions to the input features. Moreover, since we are neglecting the stochastic part of the allocation decision, this problem is even more severe.

4.2 Replacing argmax with a softmax operator

Our attempts to increase the quality of the results by replacing the argmax operator with a softmax were not successful. We note that this change reduces the violation of the local accuracy axiom significantly. With the max approximation, the violation error can reach up to 10^{-1} and with the softmax approximation it is in the order of 10^{-2}. This means that DeepLift is able to approximate the original output with attribution weights more accurately when using the softmax approximation. The discrepancy however, between the forward and the backward pass distorts the results and makes the explanations less informative. A possible reason is that DeepLift is based on the assumption that the output of any well behaved neuron is locally linear. However, the function that represents the decision rule for the value of a selected ad does violate this rule because even an infinitesimal change in the input can shift the allocation decision and change the output value significantly. A second possibility is that for the explained models, the value of the P_{CTR} dominates the decision of the bidding value. When we use the max approximation model, we are neglecting the attribution that corresponds to the value of the conversion event and potentially we inadvertently remove noise from the results. Finally, since the agents are trained without the softmax operator, the conversion probabilities can take arbitrary values in the range [0,1], as long as they solve the allocation problem. On the other hand, softmax enforces a competition between inputs during training. Using the softmax operator during training might lead to better explanation results afterwards. Figure 4.9 and fig. 4.10 display the performance of the two approximations on the perturbation test.

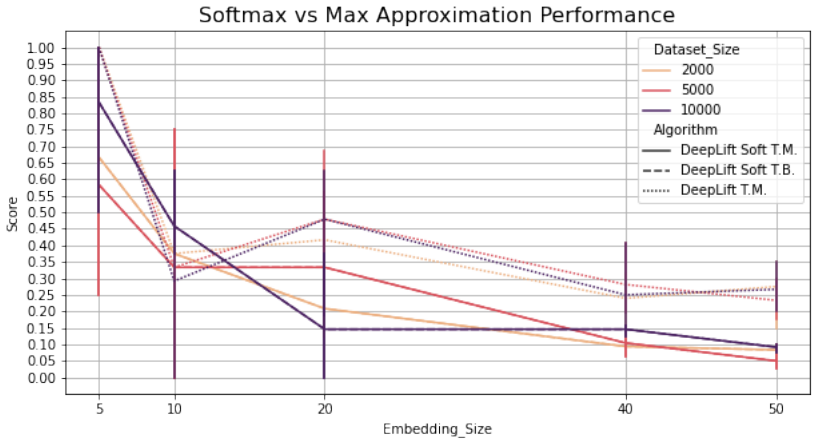

Figure 4.9: Comparative results of the Softmax vs Max approximation performance on the perturbation test. ”T.M.” denotes using the mean bid as target and ”T.B” using the sampled bid as target. Results are aggregated for 3 agents.

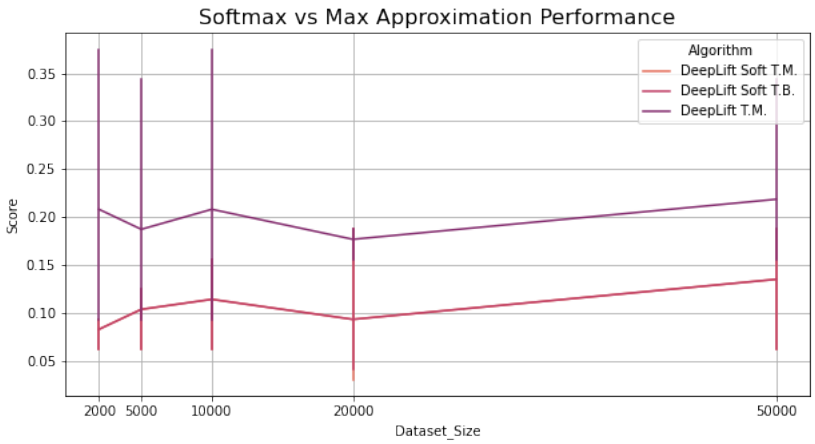

Figure 4.10: Comparative results of the Softmax vs Max approximation performance on the perturbation test for a catalogue size of 30 and embedding size of 40. ”T.M.” denotes using the mean bid as target and ”T.B” using the sampled bid as target. Results are aggregated for 3 agents.

Chapter 5

Conclusions & Future Work

5.1 Conclusions

For the scope of this thesis we utilized the DeepLift algorithm and applied it on models trained with different configurations of the AuctionGym environment. We studied how the quality of the explanations changes as the complexity of the underlying simulation increases. By comparing its computational performance we found out that it is superior compared to alternative algorithms that are popular in the relevant literature. Also for the scope of our experiments it appears that it produces better results in terms of faithfulness to the original model. We believe that one reason behind this observation, is that the models used are shallow which means that the simplifying assumptions that the DeepLift algorithm relies on conceptually, are more likely to hold in our problem setting. We also need to mention that we only used Gradient Based Algorithms to generate global explanations of the models by generalizing over multiple local examples. The fact that DeepLift displayed superior performance on this task is by no means indicative of its performance on the task of generating local explanations.

For the DeepLift algorithm we found through our experiments that it is able to identify important features even for simulations with inputs of higher dimensionality. This result is consistent with experiments conducted in the original paper of DeepLift, where it was tasked with identifying a subset of important features. When evaluated against the harder task of deriving a total order of features' importance, then its performance decays more rapidly when the dimension increases. We note however, that these results are less definitive because the test we used is susceptible to noise as the dimension of the input increases. We also displayed empirical evidence that choosing the correct baseline value is indeed important for the performance of the algorithm. Although there

is no general rule for identifying the correct baseline value, we believe that choosing the mean value is the most reasonable choice for the given problem.

We then experimented with the catalogue size parameter of the AuctionGym environment. We observed that when an agent is able to converge to a stable policy, then DeepLift is able to provide meaningful explanations but otherwise fails. We note however that this degradation in performance is potentially not entirely due to the limitations of the algorithm used but also due to the simplifications we are forced to introduce in order to explain the original model.

For this reason, we attempted to use a different approximation that makes the model completely differentiable and potentially more amenable to explanations using Gradient Based algorithms. The empirical results were not encouraging and the quality of the explanations diminished systematically.

5.2 Future Work

Although we conducted several experiments which circumscribe the potential of using DeepLift for the models of the AuctionGym, our analysis is by no means exhaustive. One additional avenue we would like to explore is how are the results are affected when we change the generative process of the input data. Since in a realistic setting, different dimensions of the input follow different distributions, exploring how this affects the performance of DeepLift is interesting. We need to note however, that when using a more diverse generative process we might need to change the method of processing the local explanations to derive a global explanation.

As we discussed, another reason that affects the quality of the results, is that we are not explaining the original models directly but a simplified version of them. Even though our attempt to derive a more accurate approximation of the original model failed, it is still an interesting research question. We can take this thought process one step further and change the trained model with one that is differentiable, for example by swapping the argmax operator with a softmax or a Gumbel softmax, if we also want to allow exploration at this step. After all, the shortcomings we encountered when trying to generate post-hoc explanations also affect training. The two models are not connected through a differentiable pipeline and they must be trained sequentially. It would be interesting to explore, if introducing the aforementioned change would not only make the model more compatible with gradient based explainability methods, but also improve its performance since gradients can be backpropagated through both

models.

Bibliography

[1] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[2] Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. A diagnostic study of explainability techniques for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3256–3274, Online, November 2020. Association for Computational Linguistics.

[3] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In Alessandro E.P. Villa, Paolo Masulli, and Antonio Javier Pons Rivero, editors, *Artificial Neural Networks and Machine Learning – ICANN 2016*, pages 63–71, Cham, 2016. Springer International Publishing.

[4] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In Alessandro E.P. Villa, Paolo Masulli, and Antonio Javier Pons Rivero, editors, *Artificial Neural Networks and Machine Learning – ICANN 2016*, pages 63–71, Cham, 2016. Springer International Publishing.

[5] L. Breiman, Jerome H. Friedman, Richard A. Olshen, and C. J. Stone. Classification and regression trees. 1984.

[6] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[7] Zhengping Che, Sanjay Purushotham, Robinder G. Khemani, and Yan Liu. Interpretable deep models for ICU outcome prediction. In *AMIA 2016, American Medical Informatics Association Annual Symposium, Chicago, IL, USA, November 12-16, 2016*. AMIA, 2016.

[8] Jonathan Crabbe, Yao Zhang, William R. Zame, and Mihaela van der Schaar. Learning outside the black-box: The pursuit of interpretable models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[9] Yingqiang Ge, Shuchang Liu, Zelong Li, Shuyuan Xu, Shijie Geng, Yunqi Li, Juntao Tan, Fei Sun, and Yongfeng Zhang. Counterfactual evaluation for explainable ai. *ArXiv*, abs/2109.01962, 2021.

[10] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, 38(3):50–57, Oct. 2017.

[11] Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online, July 2020. Association for Computational Linguistics.

[12] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

[13] Olivier Jeunen, Sean Murphy, and Ben Allison. Learning to bid with auctiongym. In *Proc. of the AdKDD Workshop at the 28th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, AdKDD '22, 2022.

[14] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for pytorch, 2020.

[15] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[16] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA, 2017. Curran Associates Inc.

[17] Dong Nguyen. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[18] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986.

[19] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 97–101, San Diego, California, June 2016. Association for Computational Linguistics.

[20] 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, page 3145–3153. JMLR.org, 2017.

[21] 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, page 3319–3328. JMLR.org, 2017.

[22] Jilei Yang. Fast treeSHAP: Accelerating SHAP value computation for trees. In *eXplainable AI approaches for debugging and diagnosis.*, 2021.