Better Counterfactual Model Reasoning with Submodular Quadratic Component Models

Abstract

In this work, we propose a new heuristic for *component modeling*—the task of building a model that estimates the effect of model components on model behavior. Empirically, we show that even at scale, *actively* learning a quadratic model instead of a linear model increases accuracy and decreases sample complexity. Prior work has favored a linear model, because outside of accuracy, interpretability is a key desiderata for a component model. By exploiting properties of the degree 2 Fourier representation, we derive an individual influence for each point, that strictly generalizes the coefficient in a linear data model. This notion corresponds to the discrete derivative of the function at a given point, and has the benefit that it incorporates information about the rest of the dataset into the individual influence estimate. We also introduce the idea of enforcing submodularity, which theoretically can allow for better and nontrivial optimization for interesting counterfactual reasoning tasks with set cardinality constraints.

1 Introduction

Large ML models such as deep neural networks (DNNs) can be black-boxes. *Component attribution* aims to distill DNNs into a collection of common sense parts, each equipped with a *score* or *attribution* that represents the importance of that part as it pertains to a specific task. For instance, component attribution systems can assign scores to individual components which correspond to how much they contribute to a model's prediction. The components themselves are meant to represent different aspects of a model's architecture depending on the desired granularity (e.g., convolutional filters in an image classification model or attention heads in a language model).

Beyond scientific interest in cracking open black-box DNNs, component attributions have been shown to be effective in a variety of downstream model editing tasks. For example, Shah et al. (2024) show how to use good component attributions to correct individual vision model predictions, selectively "forget" entire classes of images, and improve against typographic attacks. Component attribution is also related to a separate line of work in ML interpretability which essentially asks how individual components affect model behavior. More generally, there is a highly active line of work called *mechanistic interperatability* that aims to understand different model compononents or even individual neurons in order to ultimately make models more safe and steerable – see Bereska and Gavves (2024) for a survey.

Linear component models for attribution. A recent and successful approach (Shah et al., 2024) to component attribution takes a predictive modeling perspective. Here, towards component attribution, one learns a function (a component model), that accurately predicts the *counterfactual* effect of ablating a subset of components when running a forward pass of the model on a given input.[1] For a certain ML model M, the specific approach taken by Shah et al. (2024) is to:

[1] In this case, ablating a component essentially means setting it's output to 0. In the rest of the paper, we will use the definition of ablation. However, this is not necessarily the only way to define ablation.

Preprint. Under review.

- Construct a dataset of component counterfactuals by sampling a set S of m ablated models of n components, each represented by a vector $x \in \{0,1\}^n$, where $x_i = 0$ indicates that the i^{th} component is ablated out. Each x_i is sampled i.i.d. from a random variable with mean μ.
- For any given test point z, compute m forward passes on the model indicated by the set of vectors S. Given the model output $M_z(x)$ (denoting the output of the ablated model indicated by x on z), compute the correct class margin $f(M_z(x))$.[2]
- Using the "dataset" $D = [x^{(i)}, f(M_z(x^{(i)}))]_{i \in m}$, compute linear regression to find a linear model $\theta_z : \{0,1\}^n \to \mathbb{R}$ that minimizes squared loss on D.

Shah et al. (2024) call their method Component Attribution via Regression (COAR).

Evaluation of component models. To evaluate the predictive performance of a learned component model, Shah et al. (2024) suggest the general idea (borrowed from Datamodels of Ilyas et al. (2022)) of computing the Linear Data Score (LDS) on a holdout set S_{test} sampled in the same way as S (note, performance can also be evaluated out-of-distribution by sampling S_{test} with $\mu' \neq \mu$). The LDS score is essentially the Spearman rank correlation between the predicted values $[\theta_z(y), f(M_z(y))]_{y \in S_{\text{test}}}$. Additionally, component models can be evaluated directly on their performance on downstream tasks.

Why linear models? A trilemma. We could achieve optimal LDS by merely computing ablated forward passes on counterfactuals of interest, so why learn a linear component model? The benefit lies in the way linear models attack a trilemma involving efficiency, interpretability, and usefulness.

- *Fast*. For large ML models, computing ablated forward passes every time we want to peek at some counterfactual x, $f(M_z(x))$ is computationally expensive. Thus, we want to estimate $f(M_z(x))$ given x using a much more lightweight model, such as a linear model.
- *Interpretable*. Learning a linear model gives us actual insight into the value of each component, by "reading off" attributions from the coefficients. Due to linearity, the learned coefficient θ_i attached to x_i essentially models the individual contribution of component i towards increasing correct class margin. This coefficient can thus be used as a proxy for the "importance" of a specific component to a correct prediction of model M on input z.
- *Useful*. Linear component models already achieve surprisingly high LDS with reasonable sample size m (see Figure 1), and are useful for downstream tasks in model editing (e.g. naive optimization over groups of components).

1.1 Our Contribution in a Nutshell

Linear component models do well on the trilemma by being fast, interpretable, and useful. In this work, we study whether linear models are really the best we can do. Or, does there exist another model class—perhaps nonlinear—that *increases* predictive power (LDS) and/or *decreases* sample complexity (less forward passes)? Indeed, we must not compromise speed, interpretability and usefulness when studying this question.

In a nutshell, we propose learning a quadratic polynomial as a component model. However, merely increasing the complexity of the component model to quadratic or even cubic size (i.e., number of parameters) is computationally and statistically prohibitive in large scale ML settings, where the number of components is on the order of thousands or even tens-of-thousands. Hence, it is not a priori clear that increasing component model complexity would remain fast and interpretable, nevermind useful—we would need to compute possibly millions of forward passes to satisfy a large data-hungry component model.

Due to these bottlenecks, we design a method for **efficiently** learning a quadratic polynomial, which, all at once, is potentially:

- More predictive than previous linear methods (higher LDS).
- Requires *less* data to train than previous linear methods (fewer forward passes needed).
- At least as interpretable as previous linear methods (can still "read-off" individual component attributions).

[2]The function f can also be another function of interest but Shah et al. (2024) use correct class margin.

- More useful on a certain counterfactual reasoning task, which asks for the least k such that there are k components that would change the model's prediction (and what components they are). This is a combinatorial optimization task over sets of components.

The design of our method is guided by techniques and ideas from Boolean harmonic analysis and combinatorial optimization. In particular, we introduce the idea of encouraging *supermodularity* in the learned component model. We give a brief overview of how our method achieves each bullet above.

- Our method has greater capacity for predictive modeling compared to linear methods, since the supermodular polynomial class generalizes the linear model class.
- Our method requires less data because it *adaptively* introduces stronger inductive bias despite having more parameters.
- Our method is at least as interpretable, since we use ideas from Boolean harmonic analysis to show that we can still "read off" individual component influences from the parameters of our component model.
- Our method is more useful, since a supermodular component model allows for better optimization over sets of ablations for counterfactual reasoning tasks.

We present preliminary experimental results in the setting of a ResNet-50 trained on Imagenet in section 3.

2 The New Method

2.1 Preliminaries

To describe our new method in more detail, we start by formalizing our setting. Recall that for a model M and example point z, the counterfactual $x, f(M_z(x)) \in \{0,1\}^n \times \mathbb{R}$ represents the correct class margin on example point z of the model M with component i ablated if and only if $x_i = 0$.

In this case, note that the composed function $f(M_z(\cdot))$ takes as input a binary vector x and outputs a real number. Denote this composed function by $\phi : \{0,1\}^n \to \mathbb{R}$ (let M, z be treated as fixed for now. It should be clear from the context). Learning a component model θ essentially amounts to approximating ϕ w.r.t. to a loss function such as squared loss and a distribution over x.

2.2 Useful ideas

Optimal least squares are Fourier coefficients. A recent work by Saunshi et al. (2022) illuminates a connection between studying a function like ϕ and Boolean harmonic analysis.[3] In Boolean harmonic analysis, a function $g : \{0,1\}^n \to \mathbb{R}$ is viewed as a linear combination of basis functions which are Boolean parity functions $\chi_S : \{0,1\}^n \to \{-1,1\}$, where $\chi_S(x) = (-1)^{\sum_{i \in S} x_i \bmod 2}$. Every g has a unique representation in terms of the *coefficients* of this linear combination, also known as the Fourier coefficients. We use $\hat{g}(S)$ to denote the Fourier coefficient on χ_S.

Among other results, Saunshi et al. (2022) show that the optimal parameters of a linear data/component model for the function ϕ, when the model is trained on minimizing mean squared error are the degree-1 Fourier coefficients of f. The degree d Fourier coefficients are those that are attached to the parity functions χ_S for $d = |S|$. Hence, learning a good linear model θ for ϕ is equivalent to estimating the n degree 1 Fourier coefficients.

Supermodular and submodular functions have few large Fourier coefficients. It is well known that one can learn a function by estimating all the Fourier coefficients. Since in general there are 2^n Fourier coefficients, this is not computationally practical. In large scale settings where the number of components n is very large, even $O(n^2)$ coefficients is very problematic. However, when the function is known to have certain bias in the distribution of the coefficients, efficient learning can be

[3] Saunshi et al. (2022) do this in the context of datamodels (Ilyas et al., 2022), which is basically "dual" to our setting.

possible. For example, the success of COAR demonstrates that a large amount of the total weight of the Fourier coefficients of the *ground truth* component model is on just degree 1.

We now highlight the following theorem from Feldman and Vondrák (2016).

Theorem 2.1 (Feldman and Vondrák (2016)) *Let $g : \{0,1\}^n \to [0,1]$ be a submodular function and $\alpha, \beta > 0$. Let*

$$I = \{i \in [n] \mid |\hat{g}(\{i\})| \geq \alpha\} \cup \{i \in [n] \mid \exists j, |\hat{g}(\{i,j\})| \geq \beta\}.$$

Then $|I| \leq \frac{2}{\min\{\alpha,\beta\}}$.

This theorem very nontrivially bounds the number of large coefficients of submodular functions, on degree 1 and degree 2, in terms of their minimum size. Since the theorem only deals with the magnitiude of coefficients, it also holds for supermodular functions, since negating all Fourier coefficients of a supermodular function yields a submodular function.

Additionally, we observe a straightforward way to find large (i.e., important) coefficients on without needing to estimate all n^2 degree 2 coefficients: first, find k large degree 1 coefficients, and then check the degree 2 coefficients that involve the k largest variables x_i only. All in all this estimates kn coefficients for $k << n$, as opposed to n^2 coefficients. This can be further optimized to the heuristic of only estimating all degree 2 coefficients on parity functions that contain only the top k variables. This means estimating $n + k^2 < kn$ coefficients.

Supermodular functions and positive degree 2 coefficients. Finally, we define supermodularity and submodularity.

Definition 2.1 (Submodular functions) *A set function $g : 2^N \to \mathbb{R}$ is called **submodular** if it satisfies the **diminishing returns property**. Formally, for every $A \subseteq B \subseteq N$ and every $x \notin B$, the following inequality holds:*

$$g(A \cup \{x\}) - g(A) \geq g(B \cup \{x\}) - g(B)$$

One can think of sets $A \subseteq B$ as indicator vectors, and the union operation as flipping a bit from 0 to 1.

Definition 2.2 (Supermodular functions) *A set function $g : 2^N \to \mathbb{R}$ is supermodular if and only if $-g$ is submodular.*

When a function has **negative** degree 2 coefficients, that makes it more likely to be submodular (conversely, **positive** degree 2 coefficients make it more likely to be supermodular). Observe that this is the case because degree 2 coefficients measure the effect of setting a certain variable x_i from 0 to 1 (adding i to the set), in the presence of another variable x_j already being set to 1. Hence, a negative coefficient implements a *diminishing* return, since it subtracts from the naive value of $\hat{g}(\{i\}) + \hat{g}(\{j\})$.

2.3 Description of new method

We now give a full description of our proposed heuristic. In brief, we will describe a heuristic that leverages all of the useful facts above to **efficiently** learn a supermodular quadratic polynomial as a component model for ϕ, for any fixed M and z.

1. Apply COAR to learn a linear model θ_{lin} for ϕ. Use mini-batch SGD to implement regression.
   - *From Saunshi et al. (2022), we know that the learned coefficients of θ_{lin} are estimates for the degree 1 Fourier coefficients of ϕ.*
2. Select the top k coefficients from θ_{lin}. Let the set of indices be $K \subset [n]$. Augment the linear model with interaction terms for all $K(K-1)/2$ pairwise combinations of indices in K.
   - *From the theorem of Feldman and Vondrák (2016), looking at the top k coefficients and the pairwise interactions is a reasonable heuristic for finding all the important coefficients and skipping unimportant ones (i.e., those very close to 0 in magnitude.).*

3. Continue projected mini-batch SGD on the augmented model θ_{sq}. At each SGD step, project all negative interaction term coefficients 0.

    - *The projection to nonnegative coefficents encourages supermodularity because reasoning outlined in the previous section.*

3 Imagenet: Preliminary Results for LDS by Sample Complexity

We conduct preliminary empirical studies on the predictive performance of our designed heuristic, with respect to various samples sizes. We compare our results to an implementation of COAR.

Specifically, we work in the setting of learning a component model for a ResNet-50 network trained on ImageNet. The ResNet-50 has 22,720 components, which each correspond to one convolutional filter. We generate counterfactual component ablations as our training dataset by sampling 50,000 randomly ablated ResNet-50 models, and computing forward passes on a set of 16 random images from the Imagenet test set. We use a test set of 5,000 held out random ablations to test our own component model. The random ablations are sampled by ablating each component with probability 0.1. Linear Data Score (LDS) is used as the primary metric to compare the predicted margins against the actual values.

To instantiate our heuristic, we set $k = 16$; so we use the top 16 features, as determined by preliminary coefficient magnitudes from the learned linear model, to derive 120 interaction terms. We train all models for the same number of gradient steps (at each sample size). In particular, the COAR implementation is trained for 300 epochs with mini-batch size of 64. Our method uses 200 epochs to learn the linear coefficients (step 1), and then continues training using projected SGD for another 100 epochs in step 3.

Our results are summarized by Figure 1, where we compare LDS by sample size for our method vs. COAR.

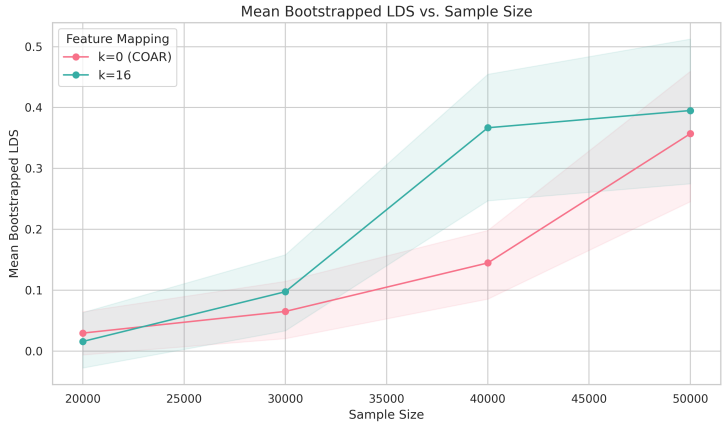

Figure 1: This lineplot gives the mean LDS of executing our method on various sample size, versus the COAR method, on 16 random test images from Imagenet. We see that our method effectively reduces sample size by 20%—from 50,000 to 40,000, in order to achieve the same LDS near 0.4. Our method can also marginally increase LDS when holding sample size constant. We plot 95% confidence intervals by the shaded areas, which are computed using 1000 bootstrapped simulations.

4 Applications

In this section, we discuss ways that our submodular quadratic component models still retains the interpretability of previous linear methods. We also discuss how we can apply more nontrivial optimization methods despite nonlinearity, in the appendix, section A. The current section should be read prior to section A.

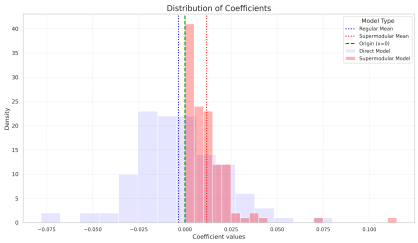

Figure 2: Showing the effect of projected gradient descent on the learned interaction coefficients.

4.1 Individual influence estimates

Let P be a p-biased distribution over x (where each bit of x is sampled from a Bernoulli random variable with mean p). The i^{th} discrete derivative of a function $g : \{0,1\}^n \to \mathbb{R}$ at a point $x \in \{0,1\}^n$ is defined as

$$\mathbf{D}_i^p[g(x)] = \sigma \cdot \frac{g(x^{i\to 0}) - g(x^{i\to 1})}{2} \tag{1}$$

Here, σ is the standard deviation of P.

While discrete derivatives are a hallmark of Boolean Fourier analysis, $\mathbf{D}_i^p[\phi(x)]$ is a quantity of high interest when attempting to understand the role of the i^{th} component towards a prediction or correct class margin of a (possibly ablated) model $M_z(x)$ on example point z. In this setting it is referred to as the individual influence of component i.

Context-aware individual influence estimates. It turns out that $\mathbb{E}_x\left[\mathbf{D}_i^p[g(x)]\right] = \hat{g}(\{i\})$.[4] Hence, what linear methods for component models optimize for is learning estimates for $\mathbf{D}_i^p[\phi(x)]$ for each i, where x is distributed the same way the training distribution is (p-biased distribution). The i^{th} learned coefficient in COAR is the estimate for $\mathbf{D}_i^p[\phi(x)]$.

However, we would really like to be able to directly estimate $\mathbf{D}_i^p[\phi(x)]$ for any x. We call this the *context-aware* setting. For downstream applications, we may be sitting on a model trained on a certain dataset x, so we want $\mathbf{D}_i^p[\phi(x)]$ instead of the average over P.

In this setting, we can use a direct identity for $\mathbf{D}_i^p[g(x)]$.

$$\mathbf{D}_i^p[g(x)] = \sum_{S: i \in S} \hat{g}(S)\chi_{S\setminus\{i\}}(x) \tag{2}$$

Now, this identity demonstrates not only that computing individual influence estimates from quadratic models is possible, but that it leads to better estimated discrete derivatives in the context-aware setting. With a quadratic model, we can now estimate $\mathbf{D}_i^p[g(x)]$ by using the direct identity (2), as follows:

$$\mathbf{D}_i^p[g(x)] \approx \theta_i + \sum_{j \neq i} \theta_{ij} x_j \tag{3}$$

Here θ_{ij} denotes the learned interaction coefficient for term $\chi_{\{i,j\}}(x)$. It follows from the same general idea that learning these coefficients is equivalent to estimating Fourier coefficients of ϕ. We thus observe that in this way, our estimates for individual influence actually *depend* on the current ablation x. We note that however that our heuristic does not estimate every θ_{ij} for all i, j.

A Top-k Counterfactual Reasoning and Group Influence via Submodular Maximization

An interesting application of a component model is for solving the *top k counterfactual* problem: find a set of at most k components such that, when ablated, most reduce the correct class margin of the

[4]Here, $x \sim P$, and $\hat{g}$ is the p-biased Fourier transform. We do not define p-biased Fourier transform in the present work, and refer the reader to a text such as O'Donnell (2014).

model M on example point z. This would correspond to the largest *group* influence of groups of size k. Questions like this are combinatorial optimization problems that can be written as follows.

$$\max_{S:|S|\leq k} \phi(\mathbf{1}) - \phi(\mathbf{1} \setminus S) \tag{4}$$

Here $\mathbf{1}$ denotes the complete set. Given a component model θ, we can try to approximately solve (4) by computing:

$$\max_{S:|S|\leq k} \theta(\mathbf{1}) - \theta(\mathbf{1} \setminus S) \tag{5}$$

Now, since $\theta(S)$ has only positive quadratic coefficients, this means that $\theta(\mathbf{1} \setminus S)$ also exhibits a compounding returns property and is indeed supermodular as well. Then, when considering $c - \theta(\mathbf{1} \setminus S)$ for a constant c, we can see that (5) is a submodular maximization problem.

Greedy submodular maximization. Submodular maximization with cardinality constraints is en extremely active area of research. We can import known algorithms that achieve good approximation guarantees for (5).

For instance, if we assume that for a certain k, $\theta(\mathbf{1}) - \theta(\mathbf{1} \setminus S)$ is monotone (i.e., growing the argument set only increases the function value), then a simple greedy optimization algorithm achieves a $(1 - 1/e)$-approximation guarantee. This was proved by Nemhauser et al. (1978).

Our quadratic model θ allows for nontrivial implementation of the greedy algorithm. Note that the greedy algorithm is inherently *context-aware*, in that the choice at each stage of the algorithm depends on the previous choices. In the previous section, we showed exactly that our quadratic model allows us to actually re-estimate individual influence at each stage, because our individual influence estimate takes into account the current ablation set.

In contrast, applying a greedy method to a linear model is trivial. This is because the individual influence estimate is independent of the current ablation set. Hence, the greedy algorithm reduces to using the sum of the top k individual estimates (i.e., the largest k coefficients in θ_{lin}). Thus the greedy algorithm is trivialized when the component model is linear, but not when it is quadratic.

If we cannot assume that for a certain k, $\theta(\mathbf{1}) - \theta(\mathbf{1} \setminus S)$ is monotone,[5] then we can used unconstrained submodular maximization algorithms. For example, in the unconstrained case, Buchbinder and Feldman (2018) show that:

Theorem A.1 (Buchbinder and Feldman (2018)) *Let f be a non-negative submodular function. There exists a deterministic algorithm that has an approximation ratio of $\frac{1}{e}$ for the problem*

$$\max_{S:|S|\leq k} f(S).$$

The algorithm makes $\mathcal{O}(k^3 n)$ queries to f.

This algorithm can be applied in linear time for the unconstrained case to solve the top k counterfactual problem. We can assume that $\theta(\mathbf{1}) - \theta(\mathbf{1} \setminus S)$ is nonnegative up to our k of interest. In this case, $k + 1$ would correspond to the fewest ablations needed to make the model's prediction incorrect.

References

Bereska, L. and Gavves, E. (2024). Mechanistic interpretability for ai safety – a review.

Buchbinder, N. and Feldman, M. (2018). Deterministic algorithms for submodular maximization problems. *ACM Transactions on Algorithms (TALG)*, 14(3):1–20.

Feldman, V. and Vondrák, J. (2016). Optimal bounds on approximation of submodular and xos functions by juntas. *SIAM Journal on Computing*, 45(3):1129–1170.

Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. (2022). Datamodels: Predicting predictions from training data. *arXiv preprint arXiv:2202.00622*.

[5]This is definitely not *always* a good assumption, since Shah et al. (2024) show that ablations can be used to *increase* correct class margin.

Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14:265–294.

O'Donnell, R. (2014). *Analysis of boolean functions*. Cambridge University Press.

Saunshi, N., Gupta, A., Braverman, M., and Arora, S. (2022). Understanding influence functions and datamodels via harmonic analysis. In *The Eleventh International Conference on Learning Representations*.

Shah, H., Ilyas, A., and Madry, A. (2024). Decomposing and editing predictions by modeling model computation. *arXiv preprint arXiv:2404.11534*.