Recovering Pairwise Interactions Using Neural Networks

Tianyu Cui \(^1\) Pekka Marttinen \(^*\)\(^1\) Samuel Kaski \(^*\)\(^1\)

Abstract

Recovering pairwise interactions, i.e. pairs of input features whose joint effect on an output is different from the sum of their marginal effects, is central in many scientific applications. We conceptualize a solution to this problem as a two-stage procedure: first, we model the relationship between the features and the output using a flexible hybrid neural network; second, we detect feature interactions from the trained model. For the second step we propose a simple and intuitive interaction measure (IM), which has no specific requirements on the machine learning model used in the first step, only that it defines a mapping from an input to an output. And in a special case it reduces to the averaged Hessian of the input-output mapping. Importantly, our method upper bounds the interaction recovery error with the error of the learning model, which ensures that we can improve the recovered interactions by training a more accurate model. We present analyses of simulated and real-world data which demonstrate the benefits of our method compared to available alternatives, and theoretically analyse its properties and relation to other methods.

1. Introduction

Detecting interactions between variables, is a challenge common to many data science tasks, and particularly important in genomics, for instance. Having the right interactions in a model makes it more understandable and interpretable. At the simplest this could mean that the output \(y\) depends on inputs \(x_1\) and \(x_2\) as:

\[
y = \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + e, \tag{1}
\]

where \(\beta_1\) and \(\beta_2\) represent main effects, \(\beta_{12}\) is the strength of the interaction, and \(e\) is noise. Traditional methods from the statistics community usually require a polynomial number of hypothesis tests (e.g. ANOVA based methods, Fisher, 1936; Wonnacott & Wonnacott, 1990), which can be time-consuming. Alternatively all possible interactions have to be pre-specified (e.g. in the regression based methods of Bien et al., 2013; Lim & Hastie, 2015; Kong et al., 2017), which restricts the scope of interactions that can be considered to, for example, multiplicative interactions as in Eq.1.

Black-box machine learning algorithms, especially neural networks, can be trained to model all useful interactions directly from the data. Therefore, an intuitive way to recover interactions is: first train a flexible machine learning model on the data, after which useful interactions related to the prediction task should be encoded into the model. Second, find the encoded interactions by interpreting the trained model. Current interpretable machine learning algorithms are often interested in finding marginal effects instead of interactions, and most of them require the model to be differentiable (Simonyan et al., 2013; Hechtlinger, 2016). Another challenge is that the misbehavior of modeling and the misbehavior of interaction measure are entangled; consequently it is not clear how to improve the procedure if the result does not follow human intuition (Sundararajan et al., 2017).

In this work, we propose a general interaction measure (IM) that can detect interacting features from nearly any black-box learning model, as long as the model produces an output for a given input. This interaction measure can be understood generically as the difference between the joint effect and the sum of the marginal effects of two features. For differentiable functions, it equals the Hessian of the input-output mapping averaged over the sample space when features are i.i.d uniformly distributed. A significant theoretical advantage of IM is that the interaction recovery error is linearly upper bounded by the maximum generalization error of the trained model, which means we can improve the recovery accuracy and stability by improving the learning model.

In the rest of the paper, we review related work about measuring marginal effects in Section 2.1, and some existing interaction detection algorithms in Section 2.2. In Section 3, we present two neural networks that we use to model interactions. We present our general interaction measure in Section 4.1, and compare it theoretically with \(H\)-statistics.
in Section 4.2, followed by a discussion about interaction recovery error in Section 4.3. Section 5 presents empirical results with both simulated and real-world datasets. We conclude in Section 6.

2. Previous Work and Notations

2.1. Measuring marginal effect

Black-Box models, especially neural networks, are hard to interpret. Consequently, many approaches have been proposed to interpret neural nets by finding contributions of features for a certain decision.

Gradient of the neural network output w.r.t. its input provides a natural analogy to linear regression coefficients (Hechtlinger, 2016; Sundararajan et al., 2017). Simonyan et al. (2013) use gradients of the class score function with respect to each pixel of an image to construct a saliency map, which highlights the areas of a given image for a certain class, and they also show saliency map is equivalent to deconvolutional networks (Zeiler & Fergus, 2014) except when ReLUs are used. Hechtlinger (2016) proposes using the gradient at a specific data point to represent local feature importance, and use averaged input gradients over a test set to indicate global feature importance. Montavon et al. (2017) use Taylor decomposition to illustrate that the product of the feature gradient and feature difference can represent feature contribution, and show how to backpropagate the contribution layer by layer. Bach et al. (2015) show that local gradient of class score function at the prediction point can be misleading, instead, the gradient with respect to a root point at the decision boundary should be evaluated. Montavon et al. (2017) show how to find a good root point by optimization.

Naive gradient based algorithms are easy to understand and implement, but they may ignore important features when the neural net has many activation functions that contain 'flat' areas, such as ReLU and Heaviside step function (Sundararajan et al., 2017; Bach et al., 2015; Shrikumar et al., 2017). LRP algorithm proposed by Bach et al. (2015) and DeepLIFT by Shrikumar et al. (2017) backpropagate the prediction difference for each input feature through hidden layers of the neural network. These methods do not suffer from flat areas in activation functions, because they propagate a 'discrete gradient'. But as Sundararajan et al. (2017) have mentioned, since chain rule does not hold for discrete gradient, different implementations of these algorithms can yield different results for the same function. Sundararajan et al. (2017) propose using a path integral of the gradient from the baseline to the prediction point, which can overcome the flat area issue, but this algorithm is not efficient because it requires approximating the integral.

Another popular approach to interpret marginal effects is perturbation, whereby the feature importance is defined as the difference between the outputs with and without a certain feature. An advantage of these methods is that they are model-agnostic, i.e. applicable to any black box model mapping an input to an output. Robnik- Šikonja & Kononenko (2008) propose prediction difference analysis, which simulates the absence of a feature by marginalizing the feature according to its empirical distribution. Zintgraf et al. (2017) improve prediction difference analysis by replacing the empirical distribution with the conditional distribution given other features. Lipovetsky & Conklin (2001) propose Shapley regression values, which are calculated by training models on all possible feature subsets containing the feature of interest, and averaging the perturbation effects over all these models. Shapley regression values are extremely difficult to compute, and Štrumbelj & Kononenko (2014) propose using a sampling approach to approximate them. A drawback of Shapley values is that they do not provide sparse explanations. Lundberg & Lee (2016; 2017) propose SHAP, an idea similar to the Shapley values, but which can estimate contributions of more explainable, e.g. sparse features instead of the original ones.

Both gradient and perturbation based methods can be adjusted for the purpose of detecting interactions, as we discuss in Section 4.

2.2. Interaction Detection

Many interaction detection methods have been proposed in statistical genetics. One successful approach is based on latent variable modeling. Zhang & Liu (2007) propose BEAM (Bayesian Epistasis Association Mapping) to detect feature interactions by using a generalized naive Bayes approach, where an interaction is indicated by latent variables associated with each feature. Zhang et al. (2011) generalise BEAM by including a conditional dependence between features, which relaxes the assumption behind naive Bayes. Wang et al. (2015) extends the approach to continuous phenotypes. Another approach is to learn a Bayesian network. Ji et al. (2015) proposed using information strength and model evidence to learn the Bayesian network structure without main effects, which means $X_i \rightarrow Y \leftarrow X_j$ structure can only represent an interaction between $X_i$ and $X_j$ when predicting $Y$. Zeng et al. (2016) improved the efficiency of this approach by using an exhaustive instead of greedy search. These methods are inefficient, especially for large datasets.

Compared to measuring marginal effects, literature on detecting interactions from neural nets is very limited. Tsang et al. (2017) propose Neural Interaction Detection (NID) to detect statistical interactions by inspecting weights of edges entering the same hidden node of a neural network. Tsang et al. (2018) transform a fully connected neural network into
a sparse network with interpretable structure that contains interaction information. These methods can detect higher order interactions with unrestricted forms of interaction, but they only perform well on multi-layer perceptrons.

In this paper, we compare our method with the state-of-the-art NID method (Tsang et al., 2017).

2.3. Notation

We use lowercase letters \( x, y \) for scalars, and boldface lowercase letters \( \mathbf{w}, \mathbf{x} \) for vectors. All of \( x \) are used for inputs, and \( y \) for output. If \( x \) has one subscript \( x_i \), it represents the \( i \)th feature and \( \mathbf{x}_i \) represents the whole feature vector except \( x_i \). If \( x \) has two subscripts \( x_{k,i} \), it stands for the \( i \)th feature for \( k \)th data item, and \( \mathbf{x}_{k,i} \) means the whole feature vector for the \( k \)th data item. Both \( x \) and \( \mathbf{x} \) can represent values of random variables \( X \) and \( \mathbf{X} \), so we can assign a probability such that \( P(x) = P(X = x) \). The output \( y \) has only one subscript, \( y_k \), standing for the target of the \( k \)th data point.

3. Modeling Interactions

Real world data can contain both main effects and interactions, where a main effect represents the influence of one of the independent variables (features/inputs) on the dependent variable (target/response/output) alone, regardless of other independent variables. Interaction effect is the joint effect of multiple independent variables on the dependent variable that cannot be explained by the main effects (Winer, 1962).

The purpose of modeling interactions is to separate the main effect from interactions. We use linear regression, \( y^{\text{lin}} = \mathbf{w}^T \mathbf{x} \), to model the main effects, and a black-box model, \( y^{\text{nn}} = g(\mathbf{x}; \theta) \), to capture the interactions. The prediction of this hybrid model is the sum of the two effects \( \hat{y} = y^{\text{lin}} + y^{\text{nn}} \) (as shown in Fig. 1 left). We train the hybrid model jointly by minimizing a loss function over a dataset of \( n \) samples \( (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n) \):

\[
L(\mathbf{w}, \theta) = \sum_{k} l(y_k, \hat{y}_k(\mathbf{w}, \theta)) + \lambda_1 \Omega_1(\mathbf{w}) + \lambda_2 \Omega_2(\theta),
\]

where \( \lambda_1 \) and \( \lambda_2 \) are the weights of regularization for the linear regression and black-box models. Since the black-box model is usually much more flexible than the linear regression, we regularize \( \theta \) heavily and have hardly any regularization on \( \mathbf{w} \), so that \( g(\mathbf{x}; \theta) \) will come to encode information that cannot be learned by the main effect model.

The benefit of the hybrid model is that, to detect interactions as described in Section 4, we can after learning simply discard the linear regression part \( \mathbf{w}^T \mathbf{x} \) to remove the main effects, and analyze \( g(\mathbf{x}; \theta) \) for interactions. In preliminary experiments (results not shown), we noticed that using a single neural network instead of hybrid model may affect interaction detection, because in practice the main effect is usually stronger than the interaction, in which case it is difficult to regularize \( g(\mathbf{x}; \theta) \) to capture both. For example, if \( g(\mathbf{x}; \theta) \) is a neural network with a small number of hidden units, it will only capture the main effects unless we incorporate a separate model for them, as is done in the hybrid model. A similar idea was also used by Tsang et al. (2017).

In this paper, we use neural networks for \( g(\mathbf{x}; \theta) \) to model interactions, but our general interaction measure is applicable to any black-box model. We consider two alternative neural network architectures for \( g(\mathbf{x}; \theta) \): the conventional Multi-Layer Perceptron (MLP) and a structure called product network, which is able to model all possible multiplicative interactions directly, as reviewed next.

3.1. Product Network

The product unit (Fig.1 right, Durbin & Rumelhart (1989)), was proposed early on to extend the family of neural network architectures, and it can model all possible multiplicative interactions between features. In classical neural networks a hidden unit calculates the weighted sum of its inputs and passes the result to an activation function. The product unit \( \Pi(x) \) instead calculates a weighted product of its inputs, where the weights correspond to the powers of the inputs, i.e.,

\[
\Pi(x) = \prod_{i=1}^{D} a_i^{w_i},
\]

where the \( w_i \) are weights learned from data and \( D \) is the number of features. However, higher order multiplication operation can be computationally slow. Therefore, using the basic properties of the exponential function (Durbin & Rumelhart, 1989) we rewrite the multiplication unit as

\[
\Pi(x) = \exp\left(\sum_{i=1}^{D} w_i \log x_i\right).
\]

In words, multiplication unit can be formed using a regular summation unit, by first log-transforming the inputs, and after the summation back-transforming using the exponential, thereby speeding up the forward and backward paths.
of most deep learning frameworks (Urban & van der Smagt, 2015). However, the logarithm transformation requires positive inputs. One approach is to train the neural network in the complex domain, which is complicated and requires approximations. Another way is to convert the data to \( \mathbb{R}_{>0} \) by adding a constant, used in this paper (Section 5.1).

4. Detecting Interactions

4.1. Interaction Measure

In this section, we define our new measure for detecting interactions from a black-box model. To introduce notation, let \( im_g(x_i, x_j) \) denote the strength of interaction between features \( x_i \) and \( x_j \) in model \( g(.) \). The \( m_g(x_i, x_j) \) is the amount of change in \( g(.) \) when \( x_i \) and \( x_j \) change simultaneously in a specified way, and \( m_g(x_i) \) is the amount of change in \( g(.) \) when \( x_i \) alone changes. With this notation, we define the strength of an interaction by the general formula

\[
im_g(x_i, x_j) = m_g(x_i, x_j) - m_g(x_i) - m_g(x_j),
\]

which intuitively implies that if the sum of two marginal changes does not equal the total change, there must be an interaction. Note that Eq.5 is very flexible, as we can choose any marginal effect measure \( m_g \) as long as it only involves finite summation and expectation operations of model outputs at different data points,

\[
m_g = \sum_k \alpha_k g(x_{k,:}) + \sum_p \beta_p \int g(x_{p,:})dP,
\]

where \( \alpha_k, \beta_p \in \mathbb{R} \), and \( P \) is a probability measure. This is sufficient to ensure the property that interaction recovery error is upper bounded by the modeling error (Section 4.3). If only the magnitude but not the direction of interaction is of interest, we can simply use \( |im_g(x_i, x_j)| \).

We demonstrate the concepts introduced in this section using a running example in Fig. 2, which considers the simple interaction in Eq.1 and assumes \( x_1, x_2 \) are uniformly distributed over the region \( ABCD \). The left panel shows the interaction between \( x_1 \) and \( x_2 \), while the right panel has no interaction. Informally, an interaction means that with different \( x_1 \), an equal change \( \Delta x_2 \) results in different changes of the output \( \Delta y \). Intuitively, if we change \( x \) along the different vertical arrows in Fig. 2, \( \Delta y \) values are different in the left panel because of the interaction but the same in the right panel, and the same applies to the horizontal arrows. We denote with \( \Delta y(A \to B) \) the change in \( g \) when we move from \( A \) to \( B \).

Properties of \( im_g \) depend on the properties of \( m_g \). For example it can be used to measure either regional or point-wise interactions, by choosing correspondingly regional or point-wise marginal effect measures \( m_g(x_i) \). Below we give one option of \( m_g(x_i) \) for each, but other choices are also possible.

**Point-wise Interaction:** Point-wise interaction refers to how much an interaction between two features contributes to the prediction at a certain data point \( x_{k,:} \). In this case, one option for \( m_g(x_{k,:}) \) and \( m_g(x_{k,:}, x_{k,:}) \) is to use prediction difference analysis (Robnik-Šikonja & Kononenko, 2008):

\[
m_g(x_{k,:}) = g(\mathbf{x}_{k,:}) - g(\mathbf{x}_{k,:}\backslash i),
\]

\[
g(\mathbf{x}_{k,:}) - \int g(\mathbf{x}_{k,:})P(x_i|^x_{k,:})dx_i,
\]

and

\[
m_g(x_{k,:}, x_{k,:}) = g(\mathbf{x}_{k,:}) - g(\mathbf{x}_{k,:}\backslash i,j)
\]

\[
g(\mathbf{x}_{k,:}) - \int g(\mathbf{x}_{k,:})P(x_i, x_j|^x_{k,:})dx_idx_j.
\]

We can interpret \( m_g(x_{k,:}) \) as how much the prediction at point \( x_k \) will change if we do not know the value of feature \( x_{k,:} \). The same intuition follows for \( m_g(x_{k,:}, x_{k,:}) \).

An example of point-wise interaction for data point \( P \) is shown in Fig. 2. The pointwise measure \( m_g \) considers the difference in \( g(.) \) between the current point and the mean, which here means that \( m_g(x_{k,:}) = \Delta g(P \to M) \), \( m_g(x_{k,:}, x_{k,:}) = \Delta g(P \to N) \), and \( m_g(x_{k,:}, x_{k,:}) = \Delta g(P \to M \to O) = \Delta g(P \to N \to O) \), where \( O \) is the center of space \( ABCD \). Therefore, the IM at \( P \) is the simply the difference between \( \Delta g(N \to O) \) and \( \Delta g(P \to M) \) or between \( \Delta g(M \to O) \) and \( \Delta g(P \to N) \).

**Regional interaction:** Regional interaction metric measures the effect of an interaction in an interval, for example the whole input space of \( g(.) \). Suppose we consider an interaction between features \( x_i, x_j \) in \( [a_i, b_i] \times [a_j, b_j] \). We denote by \( m_g|[a_i, b_i] \times [a_j, b_j](x_i) \) the marginal effect of feature \( x_i \in [a_i, b_i] \) when \( x_j \in [a_j, b_j] \), which can be calculated by:

\[
m_g|[a_i, b_i] \times [a_j, b_j](x_i)
\]

\[
= \int P(x_i| x_j \in [a_j, b_j], x_i \in [a_i, b_i]) \times (g(x_{i,j}, b_i, x_j) - g(x_{i,j}, a_i, x_j)) dx_{i,j},
\]

where we use \( g(x_{i,j}, b_i, x_j) \) to represent \( g(x_{i,j}, x_i = b_i, x_j) \). This can be understood as the expected change in model \( g(.) \) if we change \( x_i \) from \( a_i \) to \( b_i \) for all \( x_{k,:} \in [a_i, b_i] \times [a_j, b_j] \). Similarly, we define the marginal effect of two features \( x_i, x_j \) in \( [a_i, b_i] \times [a_j, b_j] \) for model \( g(.) \):

\[
m_g|[a_i, b_i] \times [a_j, b_j](x_i, x_j)
\]

\[
= \int P(x_i, x_j| x_j \in [a_j, b_j], x_i \in [a_i, b_i]) \times (g(x_{i,j}, b_i, b_j) - g(x_{i,j}, a_i, a_i)) dx_{i,j}.
\]
Then the interaction between \( x_i \) and \( x_j \) can be calculated using Eq.5.

Another intuitive way to measure interactions is to use the Hessian of \( g(\mathbf{x}; \theta) \) w.r.t. the input, which recovers \( \beta_{12} \) in Eq.1. In the regional interaction detection case, if we are interested in an interaction between \( x_i \) and \( x_j \) in \([a_i, b_i] \times [a_j, b_j]\), it is natural to use

\[
\int P(\mathbf{x}|x_i \in [a_i, b_i], x_j \in [a_j, b_j]) \frac{\partial^2 g(\mathbf{x})}{\partial x_i \partial x_j} \, d\mathbf{x}, \quad (11)
\]

where we average the point-wise Hessian over the whole data space of interest. It is easy to show that the average Hessian (Eq.11) is a special case of IM (Eq.5), when features are independent and uniformly distributed (Appendix 1).

In Fig. 2, the average Hessian (Eq.11) can be viewed as averaging all the gray arrows inside \( ABCD \), and a direct integration shows that it is equivalent to the difference between \( \Delta_g(B \rightarrow D) \) and \( \Delta_g(A \rightarrow C) \) or between \( \Delta_g(C \rightarrow D) \) and \( \Delta_g(A \rightarrow B) \). Similarly, IM constructed by Eq.9 and Eq.10 calculates the sum of two averages: the average of differences between \( \Delta_g(A \rightarrow B) \) and \( \Delta_g \) along each blue-dashed horizontal arrow, and the average of differences between \( \Delta_g(B \rightarrow D) \) and \( \Delta_g \) along each blue-dashed vertical arrow. These differences are all 0 in the right panel because there is no interaction.

![Figure 2](image)

**Figure 2.** An illustration of pairwise and regional interaction measures based on the contours of Eq.1 with \( \beta_1 = \beta_2 = 1 \). The first panel represents the case with \( \beta_{12} = 10 \), and the second panel is for \( \beta_{12} = 0 \). \( ABCD \) is the space that we are interested in. Gray arrows stand for the Hessian, and they only exist in the left figure since the right figure does not contain interactions.

If the region of interest equals the whole domain of the feature, a regional interaction becomes a global interaction.

A practical way to handle real-valued features is quantile normalization using the standard Gaussian, after which the interval \([-2.8, 2.8]\) covers 99.5% of values. In the following experiment part, we will use the global interaction measure Eq.5 with form Eq.9 to Eq.10.

### 4.2. Relation to H-statistic

Instead of comparing marginal effects, Friedman et al. (2008) propose H-statistic to detect global interactions by comparing function values directly, i.e.:

\[
H_{ij}^2 = \frac{\sum_k (PD_{i,j}(x_{k,i}, x_{k,j}) - PD_{i}(x_{k,i}) - PD_{j}(x_{k,j}))^2}{\sum_k PD_{i,j}^2(x_{k,i}, x_{k,j})} \quad (12)
\]

where \( PD_{i,j}(x_{k,i}, x_{k,j}) \) is the marginal effect of \( x_i \) as the function value if we only know \( x_i \), while our method (Eq.7 and Eq.9) measures the marginal effect as the change of function if we do not know \( x_i \), which are the two sides of the same coin.

One problem of Eq.12 is that \( H \) may not equal 0 even if \( x_i \) does not interact with \( x_j \) (Appendix 2). Moreover we need to integrate out \( D - 1 \) features to calculate \( PD_i \), while our method only requires one integration.

### 4.3. Interaction recovery error

The error when recovering interactions from data can be caused both by modeling error (step 1) and error in detecting interactions from the model (step 2). A consequence of this is that when the recovered interactions violate our intuition, we cannot distinguish between: (1). the machine learning model learns wrong interactions (Section 3), so that even a good interaction measure will make mistakes. (2). The machine learning model captures all interactions, but a bad interaction measure detects wrong interactions (Section 4).

Sundararajan et al. (2017) suggest to design interpretable machine learning methods in an axiomatic manner, so that in the case of errors we should always adjust the modeling part.

To separate the two sources of error, suppose \( f(\cdot) \) is the underlying data generating model, and \( g(\cdot) \) is the machine learning model. We define interaction recovery error as

\[
L_d = \sum_{i,j} |im_g(x_i, x_j) - \text{im}_f(x_i, x_j)|.
\]

If \( \text{im}_g \) is given by Eq.5 and \( m_g \) is assumed to follow Eq.6, then \( L_d \) can be written as a function of prediction errors directly. Moreover, \( L_d \) is linearly upper bounded by the maximum prediction error \( \epsilon (\epsilon = \sup_{k}|(g_f(x_{k,:}))|):

\[
L_d = \sum_{i,j} \sum_k \alpha_k |g_f(x_{k,:}) + \sum_p \beta_p \int (g_f(x_{p,:})) | dP| \leq D^2 \gamma \epsilon, \quad (13)
\]

where \( D \) is the number of features, and the constant \( \gamma = \sum_k |\alpha_k| + \sum_p |\beta_p| \) is determined by the choice of im. Therefore \( \epsilon \) represents the modeling error and \( \gamma \) the detection error. This means that by training a better model with a smaller \( \epsilon \), \( L_d \) will have a tighter upper bound, which makes the recovery result more stable and accurate, and so does finding an interaction measure with smaller \( \gamma \). If the interaction
5. Experiments

In this section, we apply our approaches on simulated datasets and three publicly available real-world datasets, and compare with the state-of-the-art NID (Tsang et al., 2017) method.

5.1. Simulated Data

Experimental Setup: We simulate datasets with 100 features and 4 multiplicative interactions, using the model

\[ y_i = \sum_{j=1}^{100} \beta_j x_j + \sum_{k=1}^{4} \alpha_k x_{i,k,1} x_{i,k,2} + \epsilon \]  \hspace{1cm} (14)

where \( x_{i,k,1}, x_{i,k,2} \) are features involved with the \( k \)th interaction. Interactions of 4 different effect sizes \( \alpha_k \) are considered: \([0.55, 1.29, -1.28, 1.36]\), and these same interaction coefficients are used in all datasets. The \( x_j \) are drawn uniformly from \((-1, 1)\). Noise \( \epsilon \) is Gaussian with zero mean and variance adjusted to a specified signal-to-noise ratio. Each simulated dataset includes 20,000 samples for training, 5,000 for validation, and 5,000 for testing.

Modeling Interactions: To model interactions according to Section 3, we use two neural network architectures for \( g(x; \theta) \): 1. A MLP with 3 hidden layers of sizes 75, 50, and 20 nodes. All activation functions are ReLUs. 2. A product network with one hidden layer (Section 3.1) with 20 product units. We use Eq.4 to speed up the training, and we translate \( x \) to \( \mathbb{R}_{>0} \) by adding a constant 2. The regularization of \( g(x; \theta) \) for both architectures is \( L_2 \) with coefficient \( e^{-4} \), and we set no regularization on the main effects. We use Adam with learning rate 0.001 to optimize the models.

Detecting Interactions: We apply IM and NID to detect interactions from the trained neural networks. We calculate their values for every possible pairwise interaction, and rank the feature pairs accordingly from low to high. We calculate the percentile of each true interaction in Eq.14.
for both IM \(p_{i,m}^\text{im}\) and NID \(p_{i}^\text{NID}\). A good interaction measure should make the percentile of true interactions close to 1.

**Experimental Results:** For each signal-to-noise (S/N) ratio (1.0, 0.5 and 0.1), we generated 20 datasets. Histograms in Fig. 3 show the difference between percentiles of NID and IM for each dataset \(i (p_{k,i} = p_{k,i}^\text{NID} - p_{k,i}^\text{im})\). If the \(p_{k,i}^\text{im} < 0\), it means IM is better (i.e. ranks true interactions higher) than NID for the \(i\)th dataset and \(k\)th interaction, otherwise NID is better.

Top three panels in Fig. 3 show that when we use MLP as \(g(x; \theta)\), NID and IM perform similarly with all S/N levels. However, when we use product nets (second row in Fig. 3), IM outperforms NID with all S/N values. Table 1 summarizes the mean of percentile differences in each panel.

In Fig. 4, we increase the S/N gradually from 0 to see which method can recover certain interactions with the smallest S/N. We find that IM with product networks ranks the true interactions highly sooner than the other methods for the three strongest interactions. With the weakest interaction, these methods perform similarly, except for NID with a product network, which is clearly worse. Another observation is that if product nets are used instead of MLPs, then, although the modeling error decreases (Fig. 5), NID method does not improve significantly (it is even worse for coef 0.55), but IM always yields a better result with product nets (green lines are always above yellow), because of the bounding property in Section 4.3.

### 5.2. Real-world datasets

**Datasets:** We use 3 publicly available regression datasets: California housing prices, Bike sharing, and Energy efficiency datasets. California housing prices dataset (Pace & Barry, 1997) aims to predict housing prices using 9 features, such as location, number of rooms, number of people, etc. Bike sharing dataset (Fanaee-T & Gama, 2014) predicts the hourly bike rental count from environmental and seasonal information. Energy efficiency dataset (Tsanas & Xifara, 2012) aims to predict the load of heating and cooling from the shape of a building. We use 70% of data for training, 20% for validation, and 10% for testing.

**Experimental Setup:** In real-world datasets, we do not know the true interactions. Therefore, we create a ground truth by injecting fake interactions into the datasets, and try to recover those. We inject three types of interactions \(I_{in}\): a multiplicative interaction, a multiplicative interaction with different powers, and a complex interaction. We only inject one interaction at a time in a dataset. Details are shown in Fig. 6. The output used to train the models equals the sum of the true output and injected interactions: \(\hat{y} = y + I_{in}\).

**Modeling Interactions:** Here our goal is to compare IM and NID for interaction detection, and hence we use only MLP for the modeling step, i.e. for \(g(x; \theta)\). We use three hidden layers with 75, 50, and 20 units for the first two datasets, and one hidden layer with 30 units for the third dataset. Regularization and learning rates are the same as with the simulated data.

**Detecting Interaction:** We compare IM with NID by ranking all possible pairwise interactions using both IM and NID. We apply IM and NID to the same trained neural net. Injected interactions should have a small rank (rank 1 is the smallest) if the interaction measure is good.
Interaction Recovery

Figure 6. Results for real-world datasets with injected interactions. Columns correspond to the three datasets, and rows to different types of interactions (multiplication, multiplication with powers, complex). The detailed form of an interaction is shown above each panel. The ranks of the injected interactions are shown for each dataset with different interaction strengths. The \( w \) is the weight (strength) of the interaction. In the Cal house dataset, \( x \) and \( y \) are the number of rooms and number of households. In Bike share, \( x \) is the temperature and \( y \) is the wind speed. In energy efficiency data, \( x \) and \( y \) stand for the orientation and glazing area.

Experimental Results: Results for varying interaction strengths are shown in Fig. 6. We see that as we increase the interaction strength, IM consistently yields a lower (i.e. better) rank to the injected interaction, as expected. However, NID is not consistent in this way, especially with the complicated interactions (the third row in Fig. 6), and a noisy dataset (the second column in Fig. 6). Even for those interactions for which NID works consistently, ranks with IM decrease much faster, which means that IM is able to find weaker interactions than NID.

Finally Table 2 shows the top 5 interactions in the real-world datasets by using IM when no interaction is injected. We notice that most of the top interactions are meaningful. In Cal house price dataset, the location (latitude, longitude) and population density (population, number of houses) are obvious factors that can affect house prices. In Bike share dataset, whether the day is a working-day or not, together with the time of the day and temperature, affect bike rental. In Energy efficiency dataset, interactions between wall area and height, wall area and roof area, surface area and height, roof area and height can be understood as the building shape from different perspectives, which can affect heating and cooling.

| Cal House                  | Bike Share                  | Energy efficiency          |
|----------------------------|-----------------------------|----------------------------|
| house age, income          | working, hour               | wall area, height          |
| latitude, longitude        | working, temp               | wall area, roof area       |
| population, # houses       | hour, humidity              | surface area, height       |
| latitude, # rooms          | year, temp                  | roof area, height          |
| population, # bedroom      | month, temp                 | closeness, wall area       |

Table 2. Top 5 interactions in the three public datasets, recovered using IM, without injected interactions. We can find that most of the top interactions are meaningful.

6. Conclusions

Our first contribution is to propose an intuitive and flexible interaction measure (IM) to detect pairwise interactions from any black-box machine learning model, as long as it defines a mapping from an input to an output. IM can be implemented easily by evaluating the learned model for some inputs. The second contribution is that we study a two-stage procedure to recover interactions from data by first training a hybrid neural network, and then detecting interactions from it. This procedure becomes meaningful when we bound the interaction recovery error with modeling error, which means that we can always improve the recovered interactions by training a more accurate neural network. With this connection, we can turn an interaction recovery problem into a supervised learning problem. Open questions include how to recover higher order interactions.
efficiently, and how to evaluate the confidence in recovered interactions, e.g., by using probabilistic models.

References

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one, 10(7):e0130140, 2015.

Bien, J., Taylor, J., and Tibshirani, R. A lasso for hierarchical interactions. Annals of Statistics, 41(3):1111, 2013.

Durbin, R. and Rumelhart, D. E. Product units: A computationally powerful and biologically plausible extension to backpropagation networks. Neural Computation, 1(1):133–142, 1989.

Fanaee-T, H. and Gama, J. Event labeling combining ensemble detectors and background knowledge. Progress in Artificial Intelligence, 2(2-3):113–127, 2014.

Fisher, R. Statistical Methods For Research Workers, 1936.

Friedman, J. H., Popescu, B. E., et al. Predictive learning via rule ensembles. The Annals of Applied Statistics, 2(3):916–954, 2008.

Hechtlinger, Y. Interpretation of prediction models using the input gradient. arXiv preprint arXiv:1611.07634, 2016.

Jiang, X., Jao, J., and Neapolitan, R. Learning predictive interactions using information gain and bayesian network scoring. PloS one, 10(12):e0143247, 2015.

Kong, Y., Li, D., Fan, Y., Lv, J., et al. Interaction pursuit in high-dimensional multi-response regression via distance correlation. The Annals of Statistics, 45(2):897–922, 2017.

Lim, M. and Hastie, T. Learning interactions via hierarchical group-lasso regularization. Journal of Computational and Graphical Statistics, 24(3):627–654, 2015.

Lipovetsky, S. and Conklin, M. Analysis of regression in game theory approach. Applied Stochastic Models in Business and Industry, 17(4):319–330, 2001.

Lundberg, S. and Lee, S.-I. An unexpected unity among methods for interpreting model predictions. arXiv preprint arXiv:1611.07478, 2016.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In Advances in Neural Information Processing Systems, pp. 4765–4774, 2017.

Montavon, G., Lapuschkin, S., Binder, A., Samek, W., and Müller, K.-R. Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition, 65:211–222, 2017.

Pace, R. K. and Barry, R. Sparse spatial autoregressions. Statistics & Probability Letters, 33(3):291–297, 1997.

Robnik-Šikonja, M. and Kononenko, I. Explaining classifications for individual instances. IEEE Transactions on Knowledge and Data Engineering, 20(5):589–600, 2008.

Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. arXiv preprint arXiv:1704.02685, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2013.

Štrumbelj, E. and Kononenko, I. Explaining prediction models and individual predictions with feature contributions. Knowledge and Information Systems, 41(3):647–665, 2014.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. arXiv preprint arXiv:1703.01365, 2017.

Tsanas, A. and Xifara, A. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. Energy and Buildings, 49: 560–567, 2012.

Tsang, M., Cheng, D., and Liu, Y. Detecting statistical interactions from neural network weights. arXiv preprint arXiv:1705.04977, 2017.

Tsang, M., Liu, H., Purushotham, S., Murali, P., and Liu, Y. Neural Interaction Transparency (NIT): Disentangling Learned Interactions for Improved Interpretability. In Advances in Neural Information Processing Systems, pp. 5809–5818, 2018.

Urban, S. and van der Smagt, P. A neural transfer function for a smooth and differentiable transition between additive and multiplicative interactions. arXiv preprint arXiv:1503.05724, 2015.

Wang, J., Joshi, T., Valliyodan, B., Shi, H., Liang, Y., Nguyen, H. T., Zhang, J., and Xu, D. A bayesian model for detection of high-order interactions among genetic variants in genome-wide association studies. BMC Genomics, 16(1):1011, 2015.

Wilcoxon, F. Individual comparisons by ranking methods. Biometrics Bulletin, 1(6):80–83, 1945.
We show that average Hessian (Eq.11) is a special case of

\[ \text{Eq.11} \]

Proof. IM (Eq.5) with (Eq.6), when features are independent and

\[ \text{Eq.5} \]

Zintgraf, L. M., Cohen, T. S., Adel, T., and Welling, M.

Im (Eq.5) with (Eq.6), when features are independent and

\[ \text{Eq.6} \]

Zhang, Y. and Liu, J. S. Bayesian inference of epistatic interactions in case-control studies. Nature Genetics, 39 (9):1167, 2007.

Zhang, Y., Jiang, B., Zhu, J., and Liu, J. S. Bayesian models for detecting epistatic interactions from genetic data. Annals of Human Genetics, 75(1):183–193, 2011.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In European Conference on Computer Vision, pp. 818–833. Springer, 2014.

Wonnacott, T. H. and Wonnacott, R. J. *Introductory statistics*, volume 5. Wiley New York, 1990.

Winer, B. J. Statistical principles in experimental design. 1962.

\[ \text{Eq.11} \]

So Eq.16 is equivalent to Eq.5, if we take:

\[ m_{g,[a_i,b_i] \times [a_j,b_j]}(x_i) = c \int P(x_{i,j}) [g(x_{i,j}, b_i, a_j) - g(x_{i,j}, a_i, a_j)] dx_{i,j}, \]

(18)

and

\[ m_{g,[a_i,b_i] \times [a_j,b_j]}(x_i, x_j) = c \int P(x_{i,j}) [g(x_{i,j}, b_i, b_j) - g(x_{i,j}, a_i, a_j)] dx_{i,j}, \]

(19)

So average Hessian (Eq.11) is a special case of IM (Eq.5), and Eq.18 and Eq.19 take the form in Eq.6.

B. H-statistic may not be 0 for non-interacting pairs

We give an example when H-statistics may not be 0.

Consider one simple function \( y = x_1 + x_2 \), where \( x_1, x_2 \sim U[0, 1] \). Then \( PD_1(x_1) = 0.5 + x_1, PD_2(x_2) = 0.5 + x_2 \), and \( PD_{1,2}(x_1, x_2) = y \). So we can observe that \( PD_{1,2}(x_1, x_2) - PD_1(x_1) - PD_1(x_2) = -1 \) for all \( x_1, x_2 \in [0, 1] \), even \( x_1, x_2 \) does not interact with each other.

C. Relates recovery error to modeling error

We show that if marginal effect \( m_g \) follows Eq.6, the recovery error is related to the model prediction error directly. Moreover, the recovery error is upper bounded linearly by the maximum prediction error.

Since \( im_g(x_i, x_j) = m_g(x_i, x_j) - m_g(x_i) - m_g(x_j) \), then \( im_g \) should also take the same form:

\[ im_g = \sum_k \alpha_k g(x_{k,\cdot}) + \sum_p \beta_p \int g(x_{p,\cdot}) dP. \]

(20)

Since all terms appear in \( im_g(x_i, x_j) \) will appear in \( im_f(x_i, x_j) \) as long as we are using the same measurement, therefore

\[ L_d = \sum_{i,j} |im_g(x_i, x_j) - im_f(x_i, x_j)| \]

\[ = \sum_{i,j} \sum_k \alpha_k (f(x_{k,\cdot}) - g(x_{k,\cdot})) + \sum_p \beta_p \int (g - f)(x_{p,\cdot}) dP. \]

(21)

This means that the recovery error \( L_d \) is related to the modeling error \( (g - f)(x_{k,\cdot}) = g(x_{k,\cdot}) - f(x_{k,\cdot}) \), and interaction
measures \((\alpha_k, \beta_p)\). Moreover, we have

\[
L_d = \sum_{i,j} |i m_g(x_i, x_j) - i m_f(x_i, x_j)|
\]

\[
\leq \sum_{i,j} \left( \sum_k |\alpha_k'| |(g - f)(x_{k,:})| + \sum_p \int |\beta_p'| |(g - f)(x_{p,:})| dP \right)
\]

\[
\leq \sum_{i,j} \left( \sum_k |\alpha_k'| + \sum_p |\beta_p'| \right) \epsilon = D^2 \gamma \epsilon,
\]

(22)

where \(\epsilon\) is the maximum prediction error, i.e. \(\epsilon = \sup_k (g - f)(x_{k,:})\), \(D\) is the number of features, and \(\gamma = \sum_k |\alpha_k'| + \sum_p |\beta_p'|\), which is only related to the interaction measure that we choose.

D. Unbounded recovery error

We give an example when \(L_d\) cannot be bounded in a similar way.

\[
\text{Figure 7. The gradient cannot produce a bounded recovery error while the averaged gradient can have a bounded recovery error.}
\]

Suppose we use point-wise input Hessian at data \(x_{k,:}\) as an interaction measure: \(i m_g(x_i, x_j) = \frac{\partial^2 g(x)}{\partial x_i \partial x_j} \bigg|_{x=x_{k,:}} = \frac{\partial^2 g(x_{k,:})}{\partial x_i \partial x_j}\). Obviously, it does not follow Eq.6. And the recovery error is:

\[
L_d = \sum_{i,j} \left| \frac{\partial^2 g(x_{k,:})}{\partial x_i \partial x_j} - \frac{\partial^2 f(x_{k,:})}{\partial x_i \partial x_j} \right|
\]

\[
= \sum_{i,j} \lim_{\Delta x_i, \Delta x_j \to 0} \frac{d(x_{k,:}, x_i + \Delta x_i, x_j + \Delta x_j)}{\Delta x_i \Delta x_j}
\]

\[
= \frac{d(x_{k,:}, x_i, x_j + \Delta x_j)}{\Delta x_i \Delta x_j} - \frac{d(x_{k,:}, x_i + \Delta x_i, x_j)}{\Delta x_i \Delta x_j}
\]

\[
+ \frac{d(x_{k,:}, x_i, x_j)}{\Delta x_i \Delta x_j} \leq \sum_{i,j} \lim_{\Delta x_i, \Delta x_j \to 0} \left| \frac{4\epsilon}{\Delta x_i \Delta x_j} \right|
\]

\[
= \lim_{\Delta x_i, \Delta x_j \to 0} \frac{4\epsilon D^2}{\Delta x_i \Delta x_j} = +\infty
\]

(23)