Neural Computing and Applications (2023) 35:9657–9685
https://doi.org/10.1007/s00521-023-08204-w

ORIGINAL ARTICLE

Linear iterative feature embedding: an ensemble framework for an interpretable model

Agus Sudjianto[1] · Jinwen Qiu[1] · Miaoqi Li[2] · Jie Chen[3]

Received: 20 May 2022 / Accepted: 6 January 2023 / Published online: 29 March 2023

Abstract
A new ensemble framework for an interpretable model called linear iterative feature embedding (LIFE) has been developed to achieve high prediction accuracy, easy interpretation, and efficient computation simultaneously. The LIFE algorithm is able to fit a wide single-hidden-layer neural network (NN) accurately with three steps: defining the subsets of a dataset by the linear projections of neural nodes, creating the features from multiple narrow single-hidden-layer NNs trained on the different subsets of the data, combining the features with a linear model. The theoretical rationale behind LIFE is also provided by the connection to the loss ambiguity decomposition of stack ensemble methods. Both simulation and empirical experiments confirm that LIFE consistently outperforms directly trained single-hidden-layer NNs and also outperforms many other benchmark models, including multilayers feed forward neural network (FFNN), Xgboost, and random forest (RF) in many experiments. As a wide single-hidden-layer NN, LIFE is intrinsically interpretable. Meanwhile, both variable importance and global main and interaction effects can be easily created and visualized. In addition, the parallel nature of the base learner building makes LIFE computationally efficient by leveraging parallel computing.

Keywords Linear iterative feature embedding · Ensemble method · Loss decomposition · Variable importance · Interaction detection

1 Introduction

Ensemble methods have proved successful in the majority of machine learning competitions, as they integrate multiple machine learning algorithms into one predictive model. In particular, there are three main types of ensemble methods, including bootstrap aggregating (bagging), boosting, and stacking. Bagging and stacking induce base learners independently and aggregate them following a deterministic averaging process, while boosting learns sequentially in a very adaptive way and combines learners using a pre-specified strategy. The main goal of bagging is to arrive at an ensemble method with less variance than its base learners, whereas boosting aims to produce a strong model that is less biased than their base learners. Stack ensemble aims at reduce both variance and bias by combining diversified base learners. Compared with a single model, all of these ensemble methods can significantly improve predictive performance either by bias or variance reduction.

However, the final model produced by ensemble method is still regarded as a black-box model, since the

The views expressed in this paper are those of the authors and do not reflect the views of Wells Fargo.

This research was done while the author was at Wells Fargo.

✉ Jie Chen
Jie.Chen@wellsfargo.com

Agus Sudjianto
Agus.Sudjianto@wellsfargo.com

Jinwen Qiu
qjwsnow_ctw@hotmail.com

Miaoqi Li
Miaoqi.Li@wellsfargo.com

[1] Wells Fargo, 401 S Tryon St, Charlotte 28202, NC, USA

[2] Wells Fargo, 11625 N Community House Rd, Charlotte 28277, NC, USA

[3] Well Fargo, 3440 Walnut Ave, Fremont 94538, CA, USA

combination of base learners leads to a complicated model structure and makes inner decision-making process not transparent for human beings. The ability to explain the rationale behind one's decisions to others is an important aspect of human intelligence in either social interaction or educational context. The interpretability of the results to enable business owners or regulators to better understand risk management decision processes and compel companies to meet regulatory requirements. Some commonly used interpretable models, such as linear model and general additive model, cannot compete with ensemble models including Xgboost [1] and random forest [2] since its simple structure cannot capture complicated dynamic data patterns.

In recent decades, some research works focus on enhancing interpretability by developing tools to 'open up the black box.' There are, broadly speaking, three interrelated model-based areas of research: (a) global diagnostics (Sobol & Kucherenko (2009) [3], Kucherenko (2010) [4]); (b) local diagnostics (Sundararajan et al. (2017) [5], Ancona et al. (2018) [6]); and (c) development of approximate or surrogate models that may be easier to understand and explain. However, (Rudin and Cynthia (2019) [7]) suggests to avoid using explainable black-box models in high-stakes decisions since they are sometimes problematic with several reasons such as unreliable presentation or lack of details of what the original model delivers. Therefore, other researchers made efforts to build inherently interpretable models, such as explainable neural network (xNN) proposed by (Vaughan et al. (2018) [8]), adaptive explainable neural networks (AxNNs) by (Chen et al. (2020) [9]), and explainable neural networks with constraint (Yang et al. (2020) [10]). Following this research direction, we try to build an inherently interpretable with a predictive performance as strong as some black-box models or an even better performance.

In this paper, our LIFE algorithm fulfills three main goals: competitive predictive performance, boosted computation efficiency, and an interpretable model. We know that the single-hidden-layer NN has universal approximation property in theory and is easy to be interpreted due to simple architecture. However, we need to resort to a wide single-hidden-layer NN with a large number of neural nodes to obtain a strong predictive performance, which is numerically difficult to estimate in practice. Therefore, by leveraging the ensemble method and a simple structure of a single-hidden-layer NN, we developed an innovative and flexible framework called LIFE to train wide single-hidden-layer NNs, which can achieve both high accuracy and easy interpretability in both regression and classification settings.

In this algorithm, we first use a special hierarchical structure of multiple single-layer NNs to perform data sampling based on the linear projection of neurons, and then train multiple narrow single-hidden-layer NNs with ReLU activation as base learners on different subsets of the dataset; finally, we aggregate neural nodes as features from multiple base learners into a wide single-layer NN and do a join estimation via a linear model. Compared with the traditional training strategy, LIFE effectively avoids directly training a wide single-layer NN by extracting features from multiple narrow single-hidden-layer NNs trained on different subsets of dataset instead. In this way, we can introduce diversity among the base learners, which tends to decrease the total uncertainty after ensemble and, thus, yields better results empirically. To achieve good diversity among base learners, we build a hierarchical structure of multiple single neural networks, and leverage the linear projections of the neurons to define the subset of sampling. In addition, the algorithm combines the features defined by neurons from the single-layer NN base learner, for which we called neurons flattening, and this can further improve results compared with traditional ensemble methods. This technique also allows LIFE to take advantage of parallel computing to improve computational efficiency through training multiple narrow single-hidden-layer NNs simultaneously.

Extensive analyses on both simulated data and public real data verify its effectiveness in both predictive and computational performance. Furthermore, we provide theoretical foundation for LIFE and prove the importance of the diversity of base learners by exploring the relationship with two-stage ensemble stacking and the ambiguity loss decomposition for two-stage ensemble stacking. The final single-hidden-layer NN architecture obtained from LIFE allows to visualize the neural network weights and bias and understand the input and output relationship easily. In particular, a single-hidden-layer NN with rectified linear unit (ReLU) activation function [8] is equivalent to an additive index model with linear splines on linear projections. Moreover, it can be considered a local linear model, where all predictors are easily visualized by a parallel coordinates plot [11]. The main and interaction effects can also be identified by aggregating local linear model coefficients.

In general, our main contributions are summarized below:

1. LIFE is an innovative and flexible framework for ensemble methods, which allows different kinds of variants.
2. LIFE can achieve a better predictive performance than traditional single-hidden-layer NN training methods, as demonstrated by the theoretical background and empirical experiments.

3. LIFE can still keep model interpretable, and a new interpretation tool is introduced to detect main and interaction effects.
4. LIFE can improve the computation efficiency via easy parallelization, rendering wide single-layer NN training faster.
5. An theoretical foundation for LIFE based on ambiguity loss decomposition and diversity of base learners is provided.

The rest of the paper is organized as follows: In Sect. 2, we introduce LIFE algorithm and theoretical rationale behind LIFE through loss decomposition. Extensive experiments on simulated and real data are conducted in Sect. 3 to test the performance of LIFE under various conditions in comparison with other benchmark algorithms. In Sect. 4, we explored interpretation of model such as main or interaction effect detection. In Sect. 5, we discuss the model pruning and the extension of LIFE algorithm in more depth. In Sect. 6, we provide our conclusions.

2 Methodology

In this section, we introduce a new proposed ensemble method called LIFE. The LIFE algorithm is mainly used to train a wide single-hidden-layer NNs in an iterative way. First, the general framework for LIFE including three steps is introduced in Sect. 2.1. Then, details of LIFE algorithms are provided in Sect. 2.2. Finally, the theoretical foundation is discussed in Sect. 2.3.

2.1 General framework for LIFE

As shown in Fig. 1, the LIFE algorithm consists of three steps: data sampling, base learner training and feature extraction, model aggregation and pruning. Figure 1 presents several options in the white box for each step, which allow for various combinations of those steps to achieve pre-specified goals.

1. The first step consists in defining subsets of data via active functions of NN neurons, which are obtained by training multiple single-hidden-layer NNs in a hierarchal structure as illustrated in Fig. 2. The diversified base learners can be generated by training based on these subsets for datasets. More theoretical explanation on ensemble with diversified base learners will be provided in Sect. 2.3. There are other alternative ways to generate sampling for base learner training, e.g., bootstrapping in the traditional method or data splitting via random projection. Through leveraging linear projections from trained NNs, our sampling method in the supervised setting can more effectively generate the diversity among base learners, which is demonstrated by empirical experiments in Sect. 5.1.
2. The second step consists in training base learners on different subsets of data sampled during the first step. Various options can be considered, e.g., single-layer NN, multiple layer NN, regression or decision tree, etc. In this paper, the single-hidden-layer NN is used as the base learner given its interpretability. After estimating multiple NN base learners, all activation functions of the neurons from NN base learners are extracted as new features.
3. The third step consists in combining all new extracted features from different base learners in the second step to construct the final predictive model. In addition, new features can be pruned through regularization or other methods to generate a more parsimonious model. We use linear model and elastic net in our LIFE algorithm which is simple and straightforward in this paper, but some other more complicated model aggregating methods, e.g., adaptive regression model screen, and pruning methods, e.g., base learner selection algorithm 3, will be discussed in Sect. 5.2.

2.2 LIFE algorithm

LIFE algorithm is an iterative process with multiple single-layer NN base learners trained in each iteration. Assume

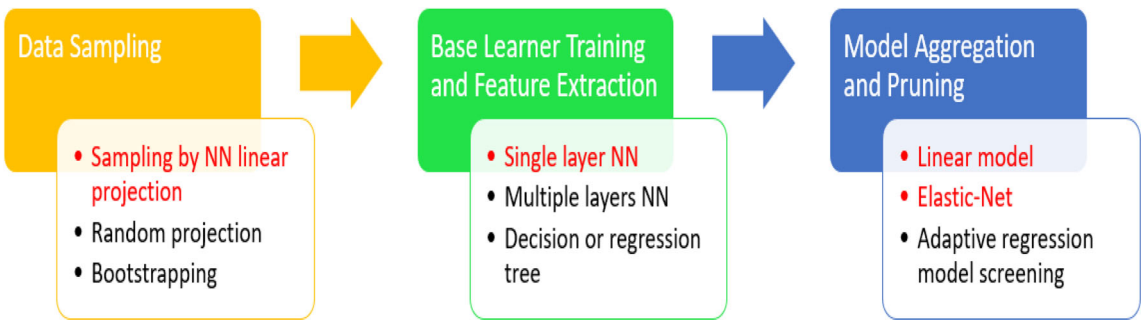


Fig. 1 It is the general framework of LIFE algorithm and options colored in red in each step are used in the paper "colour figure online"

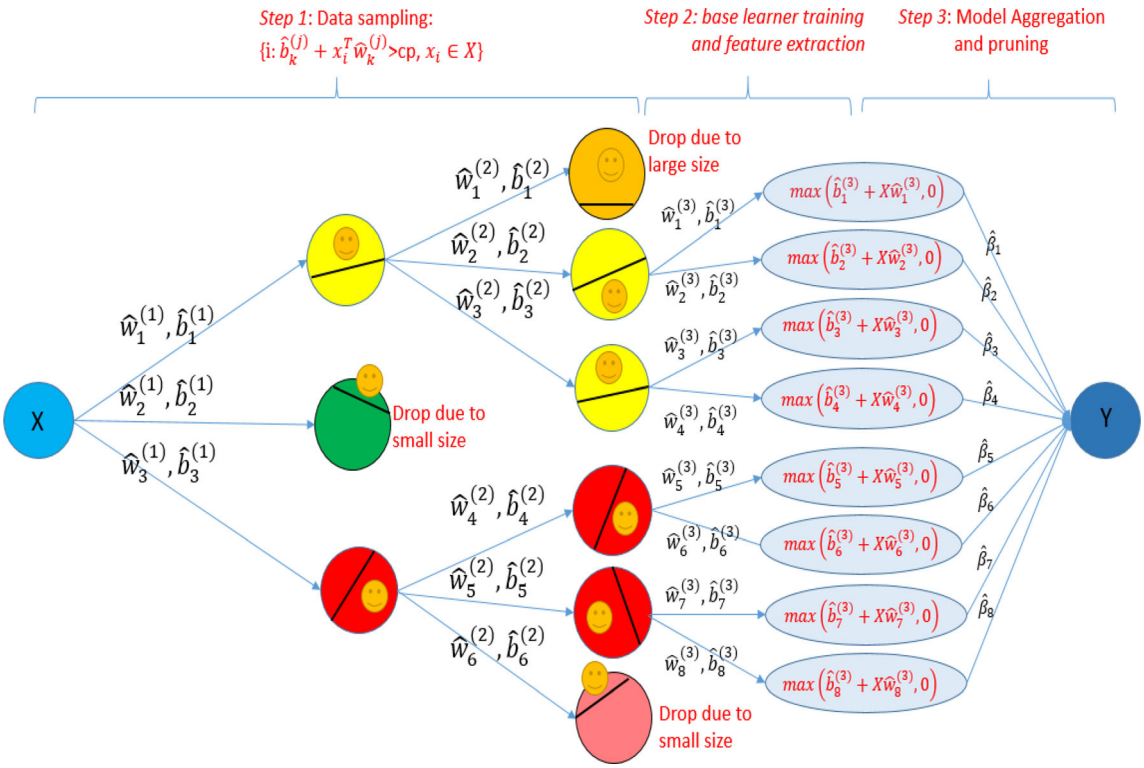


Fig. 2 An illustration of LIFE framework with $[K_1, K_2, K_3] = [3, 3, 2]$. Regions with a smiling emoji face indicate that the subsets satisfying the sampling conditions and are used for NN models training in the next iteration

there are J iterations. The first $J-1$ iteration is used to define the data sampling through a hierarchical structure, and the last J iteration is used to build the features from single-layer NN base learners. Moreover, let $[K_1, \cdots, K_J]$ denote the collection of the number of hidden neurons for single-hidden-layer NNs in all iterations, where K_j is the number of hidden neural nodes for single-hidden-layer NNs which are trained in the $j^{(th)}$ iteration. Figure 2 gives an illustration to LIFE framework, with $[K_1, K_2, K_3] = [3, 3, 2]$, where $\hat{b}_k^{(j)}$'s and $\hat{w}_k^{(j)}$'s represent biases and weights from single-hidden-layer NNs respectively, $\hat{\beta}_k$'s are coefficients used to linearly combine all new neurons in the final step, and the cp is the cutoff point to define subsets by controlling subset size.

The LIFE framework illustration in Fig. 2 can be separated into three steps as displayed in Fig. 1. The two iterations in the first step are used to perform data sampling, in which we first fit a single-hidden-layer NN with three hidden neurons, then define subsets by $b_k^{(1)} + x_i^T w_k^{(1)} > cp$, where $k = 1, 2, 3$; $i = 1, \cdots, N$, given estimated bias and weight in each neuron. Further, the entire neuron is dropped and no longer be used for next iteration if the defined subset in this neuron is either too large or too small based on pre-specified criteria. For example, the green neuron in the middle is dropped due to small size of subset. If the subset size is close to the full data size, the sample is almost identical to original training data, which is not beneficial for generating diversified samples. On the other hand, if the subset size is too small, the sample is not representative of the original data, and the base learner built on this sample does not have good performance on the entire training data. We will show in Sect. 2.3, the diversity and accuracy of the base learners are the two key elements for the final ensemble performance. As the result of the first iteration in the first step, we leverage NN linear hyperplane in the neurons to perform data partition as shown in Fig. 3, of which the idea is similar to oblique trees [12]. Plots (a), (b), and (c) in Fig. 3 have shown how data are partitioned in different ways for these three neurons based on $w_k^{(1)}$'s and $b_k^{(1)}$'s, $k = 1, 2, 3$, obtained from the first iteration. The non-white regions (yellow, green or red) indicate the subsets, where all observation satisfy $b_k^{(j)} + x_i^T w_k^{(j)} > cp$, and will be used to fit single-hidden-layer NNs in the second iteration. Notice that the non-white region in plot (b) is so small, which corresponds to the dropped green neuron in Fig. 2. Plot (d) in Fig. 3 shows the combination of plot (a) and (c). As we can see, the two subsets are overlapped, which is different from the subsets defined in the traditional regression or decision tree structure, which uses the exclusive partition.

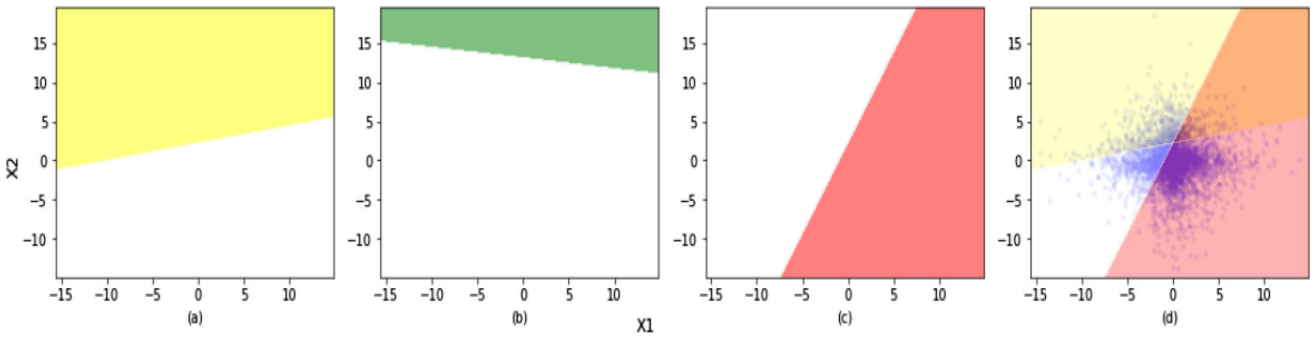


Fig. 3 That is data partition by NN linear projection after the first iteration and observations in the colored area from (**a**), (**b**), (**c**) will be selected in the each subset. The (**d**) shows two kept data partitions in one plot

In the second iteration of the first step, single-hidden-layer NNs with three hidden neurons are fitted independently using the subsets defined from the first iteration. After that, the entire training set is evaluated for data sampling through NN linear projection $b_k^{(2)} + x_i^T w_k^{(2)}$, where $k = 1, \cdots, 6; i = 1, \cdots, N$, and new subsets are defined satisfying $b_k^{(2)} + x_i^T w_k^{(2)} > cp$. Note that, the new subsets are defined on the entire training data, not the samples from the previous iteration, which is also very different from traditional regression or decision tree structure. Again, neurons with too small or too large subset are dropped forever, as shown in the first node (in brown) and the last node (in pink) of the second hidden layer from Fig. 2, corresponding to (a1) and (c3) in Fig. 4. In addition, Fig. 4 shows data partition obtained from the second iteration, where plots (a1), (a2), and (a3) represent the partitions generated from NN trained on the subset of the first node (yellow) in the first iteration, while plots (c1), (c2), and (c3) correspond to the third node (red) in the first iteration. Plot (d2) shows that each training data point as least belongs to one of the six subsets in (a1–a3) and (c1–c2). We do expect all or most data points are covered by different subsets. The reason is that the neurons with different active regions in NN are representing different features and patterns from the data, LIFE trains the same type of base learner model on part of the dataset but evaluates it on the entire dataset. This leads to small errors appearing in the region of sampled data and large errors outside region. Therefore, data sampling with these active regions can effectively define subsets for generating more diverse representation of data and producing less correlated prediction errors. The sampling in this supervised manner is

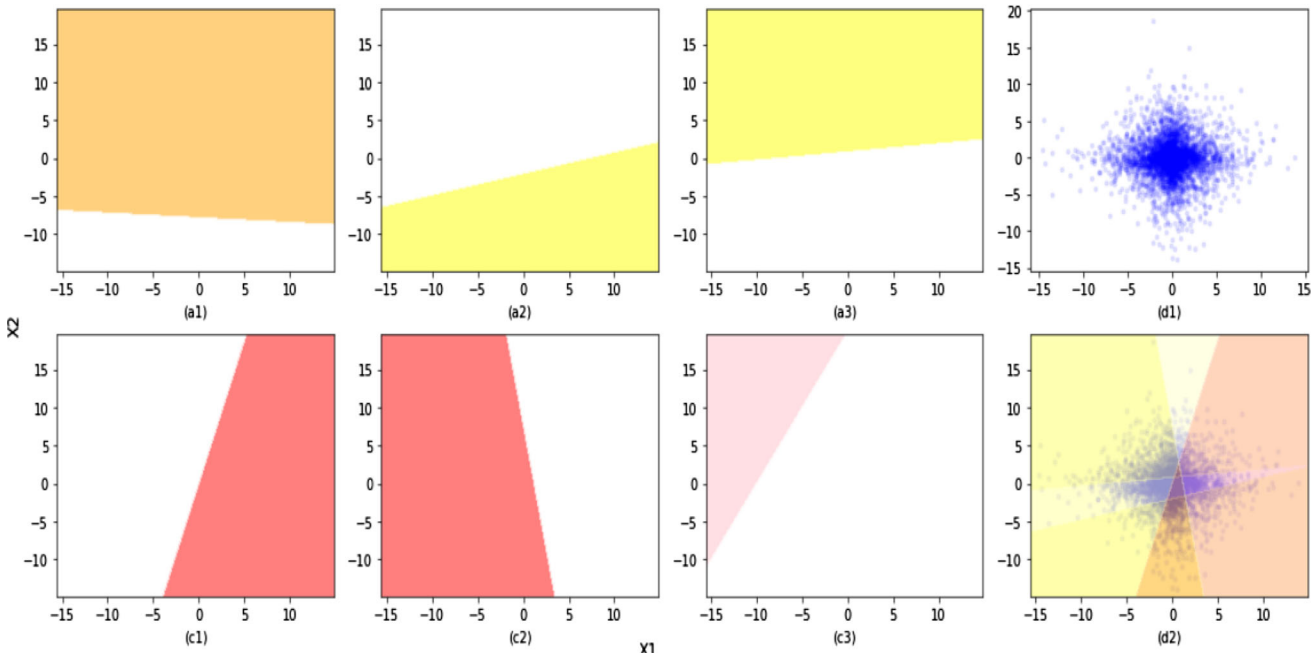


Fig. 4 It is data partition by NN linear projection after the second iteration and observations in the colored area from (**a1–a3**), (**c1–c3**) will be selected in the each subset. The (**d1**) displays all training data points in blue, and the (**d2**) shows four kept data partitions in one plot

better than sampling in a random way and this is further discussed in Sect. 5. In practice, the first step can be reduced to one iteration or have more than two iterations, displayed in Algorithm 1.

Algorithm 1 LIFE

Input: $[K_1, \cdots, K_J]$, upper bound: u, lower bound: l, cutoff point: cp
Output: $\hat{\beta}_k, \hat{b}_k^{(J)}, \hat{w}_k^{(J)}$, $k = 1, \cdots, m_j$, where m_j is the total number of neural nodes left in the j^{th} iteration.
Data: Training set $\{x_i, y_i\}_{i=1,\cdots,N}$
for $j = 1, \cdots, J$ **do**
  if $j = 1$ **then**
    i .Train a single-hidden-layer NN regressor or classifier with the number of hidden neurons equal to K_j on training set x, y by specified optimizer.
    ii. Collect estimated parameters $\hat{b}_k^{(j)}, \hat{w}_k^{(j)}$, where $k = 1, \cdots, K_j$.
    iii. $m_j = K_j$
  else
    for $k = 1, \cdots, m_{j-1}$ **do**
      i. Initialization: $M = 0$, where M records the number of NNs trained in the current iteration.
      ii. Sample observations from training set $\{x_i, y_i\}_{i=1,\cdots,n}$ based on $\hat{b}_k^{(j-1)} + x_i^T \hat{w}_k^{(j-1)} > cp$, where $x_i \subseteq \{x_i\}_{i=1,\cdots,n}$.
      iii. Record size of sampled observations as s.
      iv.
      if $l < s/N < u$ **then**
        i. Train a single-hidden-layer NN regressor or classifier with the number of hidden neurons equal to K_j on sampled subset of training set x, y by specified optimizer.
        ii. Collect estimated parameters $\hat{b}_q^{(j)}, \hat{w}_q^{(j)}$, where $q = 1, \cdots, K_j$.
        iii. $M = M + 1$
      else
        Skip this step.
      end if
      $m_j = M \times K_j$
    end for
  end if
end for
From M trained NNs, collect $\hat{b}_k^{(J)}, \hat{w}_k^{(J)}$, where $k = 1, \cdots, m_J$.
Calculate values of each neural nodes $\sigma(\hat{b}_k^{(J)} + x^T \hat{w}_k^{(J)})$, which are called new features.
Fit a linear regression or logistic model with or without regularization using new features as input to estimate $\hat{\beta}_k$, where $k = 1, \cdots, m_J$.

In the second step, four single-hidden-layer NNs, each with two hidden neurons, are trained as base learners independently on the subsets defined at the end of the first step, and then the features are generated by the ReLU activation functions. Please note that the features are evaluated on the entire training dataset. In the third step, we combine all these new features together. Since all features are obtained based on the entire training set, which technically forms a design matrix with dimension $N \times m_J$, where N is the size of training set and m_J is the total number of extracted features from step 2 with $J = 8$ shown in Fig. 2. Then, a linear model is fitted on these features

and $\beta_i, i = 1, \cdots, m_J$, is the coefficient of each feature. In the default setting, the linear regression or logistic model is applied to combine neural nodes extracted from different base learners and make a final prediction. However, there may be too many features and some of them can even be highly correlated, leading to overfitting problem. Therefore, we need to prune some redundant neural nodes through adding regularization or removing some base learners. Both methods can not only prevent overfitting, but also produce a more parsimonious NN model with fewer nodes that is beneficial for interpretation. As a regularized regression method that linearly combines the L_1 and L_2 penalties, the elastic net is one of the options, and it is used in the paper for the third step of model aggregation and pruning. LASSO and ridge regression are treated as special cases of elastic net.

By combining multiple relatively narrow but diversified single-hidden-layer NNs with K_J hidden neurons for each, the LIFE algorithm finally constructs a wide single-hidden-layer NN with m_J hidden neurons, which is 8 in Fig. 2. This trained process is completely different from the traditional methods, with which a NN is optimized as a whole by stochastic gradient based optimizers. In most cases, it is numerically difficult and computational expensive to train a single-layer NN and achieve good performance. LIFE can help overcome this difficulty and achieve decent performance. In addition, LIFE can leverage parallel computation to significantly reduce the training time. In the end, we provide the detailed pseudo-code of LIFE in J iterations, described in Algorithm 1, where m_j indicates the number of remaining neural nodes after the j^{th} iteration in the first step, where $j = 1, \cdots, J$. Both u and l are maximal and minimal proportions of training set size, which provides the upper and lower bound for subset size, respectively. The neuron will be dropped if the proportion of subset size is beyond the range.

2.3 Theoretical foundation

The ensemble method, such as stacking [13], bagging [2], boosting [14] or Bayesian model averaging [15, 16] is composed of a multiple independently or sequentially trained regressors or classifiers whose predictions are combined or sequentially derived to make final predictions. Empirically, ensembles tend to yield better results than a single model when there is a significant diversity among the models [17]. For the past few decades, many studies have been focusing on accuracy and diversity of ensemble methods in either regression [18] or classification case [19–21]. (Krogh and Vedelsby (1994) [22]) proposed ambiguity decomposition and a computable approach to minimize the quadratic error of the ensemble estimator, while (Ueda and Nakano (1996) [23]) derived a general expression of bias-variance-covariance decomposition. (Brown et al. (2005) [24]) and (Hansen (2000) [25]) investigated the connections between ambiguity decomposition and bias-variance-covariance, and have shown they are identical. Based on ambiguity decomposition, we establish the theoretical foundation for LIFE and extend loss decomposition for both regression and classification setting, which will be discussed in Sects. 2.3.1 and 2.3.2.

2.3.1 Connection to stacking

Stacking is a type of ensemble method, by which a final model is trained from the combined predictions of another models. In stacking, the predictions from different machine learning models are used as new inputs and are combined to generate a new set of predictions. Those predictions can be used on additional layers, or the process can stop here with a final result. One important assumption behind stacking is that different base learners can produce weakly correlated prediction errors that are complementary. If we use weighted averages, we might believe that some of the base learner are better or more accurate and can be assigned higher weights. In the framework of stacking, an even better approach might be to estimate these weights more intelligently by using another layer of the learning algorithm, such as the linear model.

Some major differences between stacking method and LIFE algorithm are shown in Table 1. Without neural nodes flattening, LIFE is very similar to stacking. Despite the differences between the two methods, the minimization of loss function of LIFE is approximately equivalent to the minimization of loss function for the two-stage stacking method, which firstly fits multiple base learners (NN trained on different subsets sampling by LIFE algorithm) and use predictions from base learners as input to train a model averaging model. Figures 5 and 6 indicate a strong linear relationship between LIFE and the stacking method with different hyper-parameter setup in terms of MSE loss or cross-entropy loss in both regression and classification cases. This linear relationship has be verified by both simulated data (MIM) and real data (California Housing for regression and Gamma Telescope for classification). Due to the BLUE (Best linear unbiased prediction) property of OLS estimator in linear regression, the joint estimation of the coefficients of all the combined features in the three step makes LIFE always outperform two-stage stack ensemble method with the same setting. This can be verified in Fig. 5 that all the points are below the green diagonal line. For classification case, LIFE also performs better than stacking method with smaller minimum loss as shown in the white box of Fig. 6.

Table 1 Difference between LIFE and stacking

Aspect	Stacking	Life
Base learner	Different models or same model with different settings	Same model with same setting trained on different subsets of data
Model averaging	Linear or nonlinear combination of prediction from base learners	Linear or nonlinear combination of features flatten from base learners

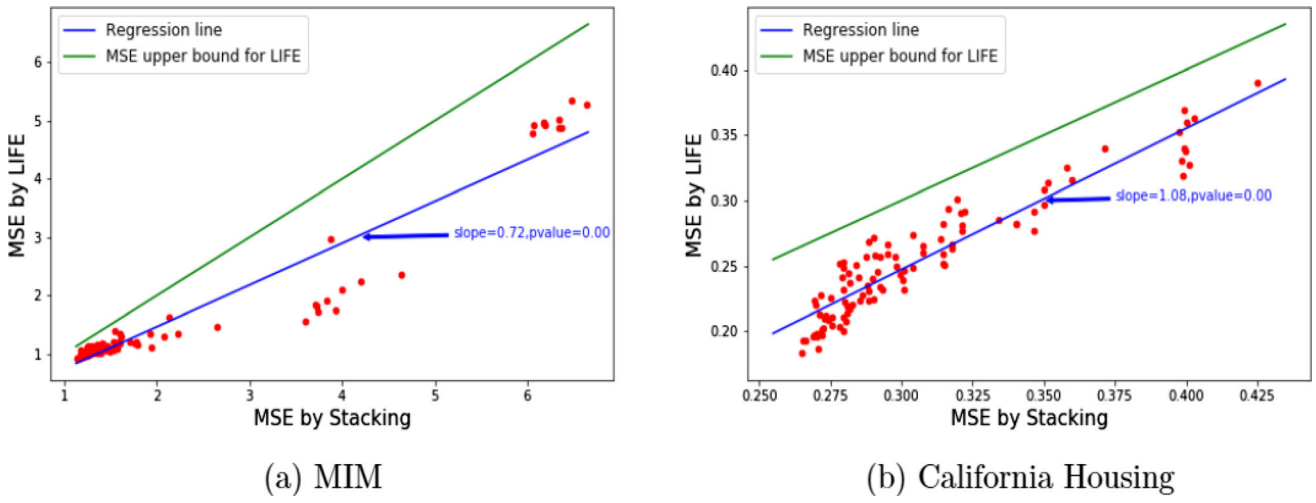


Fig. 5 Relationship between LIFE and stacking (regression)

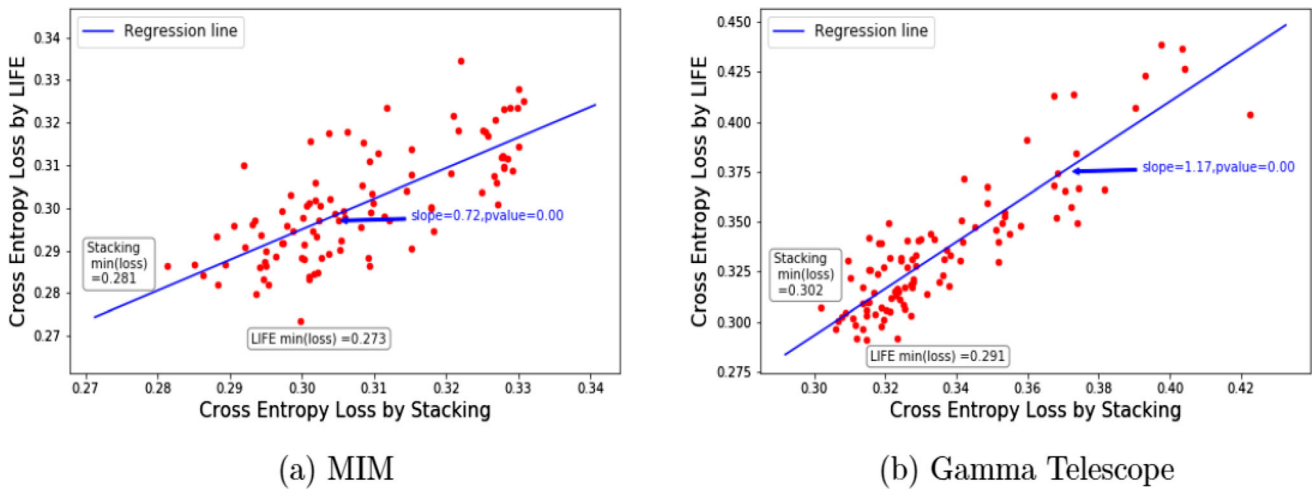


Fig. 6 Relationship between LIFE and stacking (classification)

2.3.2 Loss function decomposition

(Krogh and Vedelsby (1994) [22]) proposed ambiguity decomposition for quadratic error of the ensemble estimator which is the sum of the quadratic loss of individual base learners and the ambiguity measure for diversity. We extend the ambiguity decomposition to both mean square error and cross-entropy error via Taylor expansion, where the two loss functions corresponds to regression and classification, respectively. Let an ensemble model with M base learners be expressed as $f_{ens} = \sum_{j=1}^{M} \beta_j f^{(j)}$, where $\sum_{j=1}^{M} \beta_j = 1$ and $\beta_j \geq 0$.

For any loss function that is twice differentiable, we can expand the loss function of j^{th} base learner around output of an ensemble model based on Taylor's theorem with Peano form of the remainder as follows:

$$l(y,f^{(j)}) = l(y,f_{ens}) + l'(y,f_{ens})(f^{(j)} - f_{ens}) + \frac{1}{2}l''(y,f^{(j)\star})(f^{(j)} - f_{ens})^2, \tag{1}$$

where the value of $f^{(j)\star}$ is between f_{ens} and $f^{(j)}$. Multiplying both sides of Eq. (1) by w_j and taking a sum yield:

$$\begin{aligned}\sum_{j=1}^{M}\beta_j l(y,f^{(j)}) &= \sum_{j=1}^{M}\beta_j l(y,f_{ens}) + \sum_{j=1}^{M}\beta_j l'(y,f_{ens})(f^{(j)} - f_{ens}) \\ &\quad + \frac{1}{2}\sum_{j=1}^{M}\beta_j l''(y,f^{(j)\star})(f^{(j)} - f_{ens})^2.\end{aligned} \tag{2}$$

The second term on the right side of (2) is expressed by:

$$\begin{aligned}&\sum_{j=1}^{M}\beta_j l'(y,f_{ens})(f^{(j)} - f_{ens}) \\ &= l'(y,f_{ens})\left\{\sum_{j=1}^{M}\beta_j f^{(j)} - f_{ens}\sum_{j=1}^{M}\beta_j\right\} \\ &= l'(y,f_{ens})\{f_{ens} - f_{ens}\} = 0.\end{aligned} \tag{3}$$

Since this term is zero, the loss function $l(y,f_{ens})$ of the ensemble can be decomposed into:

$$l(y,f_{ens}) = \sum_{j=1}^{M}\beta_j l(y,f^{(j)}) - \frac{1}{2}\sum_{j=1}^{M}\beta_j l''(y,f^{(j)\star})(f_{ens} - f^{(j)})^2. \tag{4}$$

In regression case, let $f_i = \sum_{j=1}^{M}\beta_j f_i^{(j)}$ be individual predicted value of an ensemble model for i^{th} observation, where M is the number of base learners, $f_i^{(j)}$ represents individual predicted value of j^{th} base learner for i^{th} observation and β_j denotes regression coefficient for j^{th} base learner. Basically, the mean squared error (MSE) is commonly used loss function $l(y,f) = \frac{1}{N}\sum_i^N (y_i - f_i)^2$ for regression problems, where N is the total number of observations in the entire dataset. Based on Eq. (5), MSE of an ensemble model can be written in terms of the ambiguity decomposition given $x_i, i = 1,\cdots,N$:

$$\begin{aligned}MSE &= \frac{1}{N}\sum_{i=1}^{N}(y_i - f_i)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{M}\beta_j f_i^{(j)}\right)^2 \\ &= \underbrace{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\beta_j (y_i - f_i^{(j)})^2}_{\text{accuracy}} - \underbrace{\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\beta_j (f_i - f_i^{(j)})^2}_{\text{diversity}}.\end{aligned} \tag{5}$$

On the right-hand side of Eq. (5), the first term of this decomposition is to measure average prediction accuracy of base learners, while the second term is called ambiguity (hence the name of the decomposition) and can be easily interpreted in terms of diversity between individual base learners. Unlike the bias-variance-covariance decomposition, the ambiguity decomposition highlights a trade-off between the average accuracy of base learners, and their deviation from the ensemble output.

Regarding LIFE, the base learner is a single-hidden-layer neural network trained on a subset of all observations. A stronger base learner indicates a better performance of model, which is reflected by first term of Eq. (5). If the subset size is small, the base learner is also weak, which deteriorates performance. Thus, a lower bound is set up. The power of LIFE framework comes from second term diversity, which is due to data sampling during iterations. Creating different subsets through sampling allows the model to be trained on different aspects of data, which produces diversity deliberately without resorting to other machine learning algorithms. In general, the more diverse the subsets, the better the predictive performance of LIFE. Hence, the upper bound is necessary to ensure diversity of subset since subset contains almost all observations and its size is very large, making subsets loss diversity. Another parameter cutoff point is set up to balance accuracy and diversity as well.

For binary classification purposes, let $f_i = \sum_{j=1}^{M}\beta_j f_i^{(j)}$ be individual predicted probability of an ensemble model for i^{th} observation, which is weighted average of predicted probability or log-odds of base learner $f_i^{(j)}$, where $\sum_{j=1}^{M}\beta_j = 1$ and $\beta_j \geq 0$. The cross-entropy loss is widely used for classification and it can be written as follows for single observation:

$$l(y_i,f_i) = -y_i log(f_i) - (1 - y_i) log(1 - f_i). \tag{6}$$

By plugging the loss function (14) into Eq. (4), we can write average cross-entropy loss of an ensemble method on training set $\{x_i, y_i\}_{i=1,\cdots,N}$ in the probability space as follows:

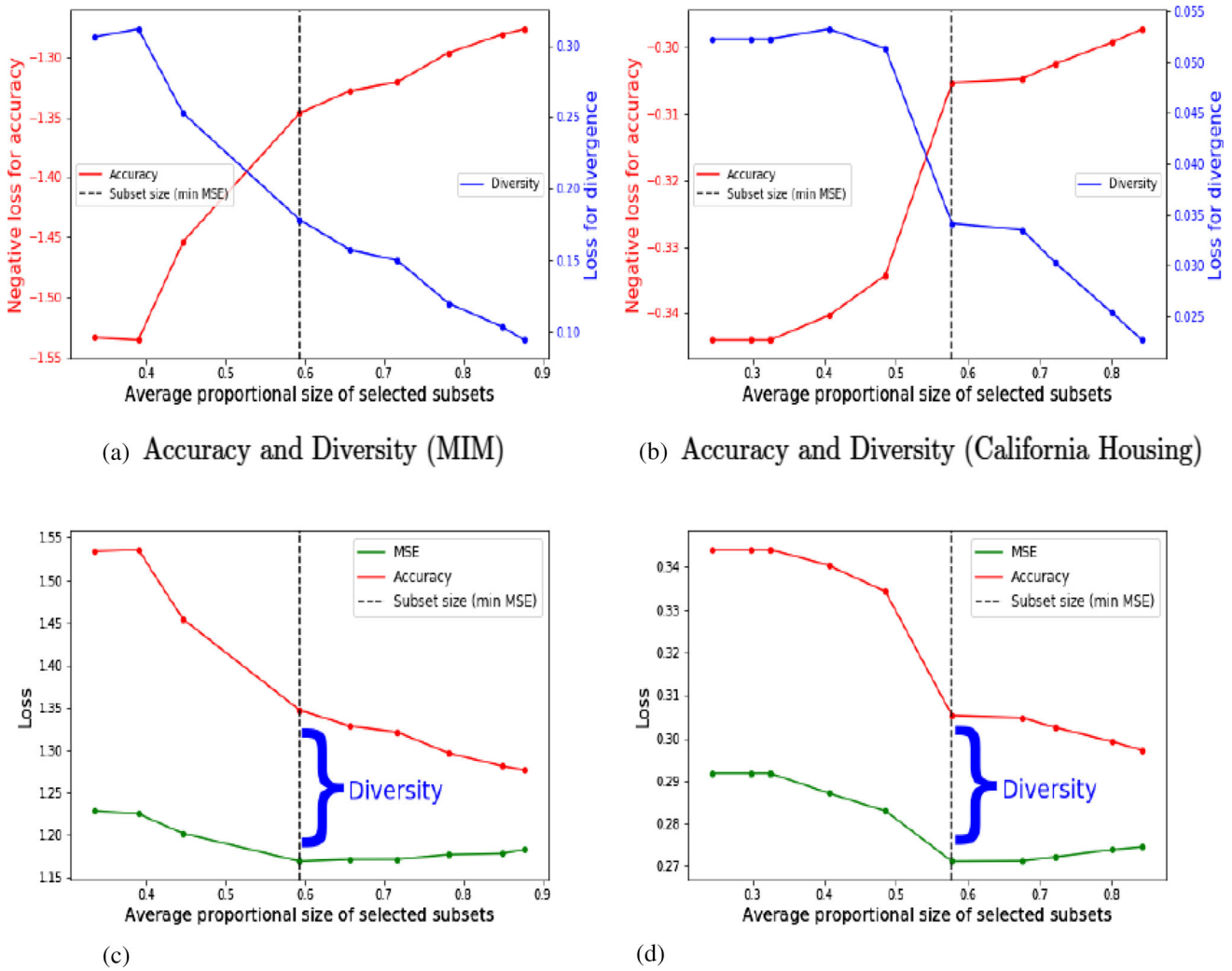


Fig. 7 Loss decomposition for regression

$$\sum_{i=1}^{N}[-y_i log(f_i) - (1-y_i)log(1-f_i)]$$
$$= \underbrace{\sum_{i=1}^{N}\sum_{j=1}^{M}\beta_j[-y_i log(f_i^{(j)}) - (1-y_i)log(1-f_i^{(j)})]}_{\text{accuracy}}$$
$$\underbrace{-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\beta_j\left\{\frac{y_i - 2f_i^{(j)\star}y_i + \left(f_i^{(j)\star}\right)^2}{[f_i^{(j)\star}(1-f_i^{(j)\star})]^2}\right\}(f_i - f_i^{(j)})^2}_{\text{diversity}}, \quad (7)$$

where $f_i^{(j)\star}$ takes value between $f_i^{(j)}$ and f_i. The term $(f_i - f_i^{(j)})^2$ is a measure of the differences in value between base learner and the ensemble. The cross-entropy loss and its decomposition in the log-odds space is provided in the "Appendix". Unlike diversity term in the regression case, the second term (diversity) in the right-hand side of Eq. (7) also includes the true class label y_i and parameter with unknown value $f_i^{(j)\star}$. However, the interpretation of decomposition is also clear. It shows that a lower average accuracy of individual base learner can be compensated by a higher disagreement with the ensemble, scaled by $\frac{1}{(f_i^{(j)\star})^2}$ if $y_i = 1$ or $\frac{1}{(1-f_i^{(j)\star})^2}$ if $y_i = 0$ in the probability space. Since $\frac{\beta_j}{(f_i^{(j)\star})^2}$ or $\frac{\beta_j}{(1-f_i^{(j)\star})^2}$ is positive, the more deviance of predicted probability between base learner and an ensemble model implies more diversity.

We implement the loss decomposition on simulated data (MIM) and real data (California Housing for regression and Gamma Telescope for classification) to LIFE without neural nodes flattening. It ensembles the predictions from single-hidden-layer NN base learner directly which is the two-stage stacking model averaging method discussed above. As subset size in the sampling step of LIFE impact

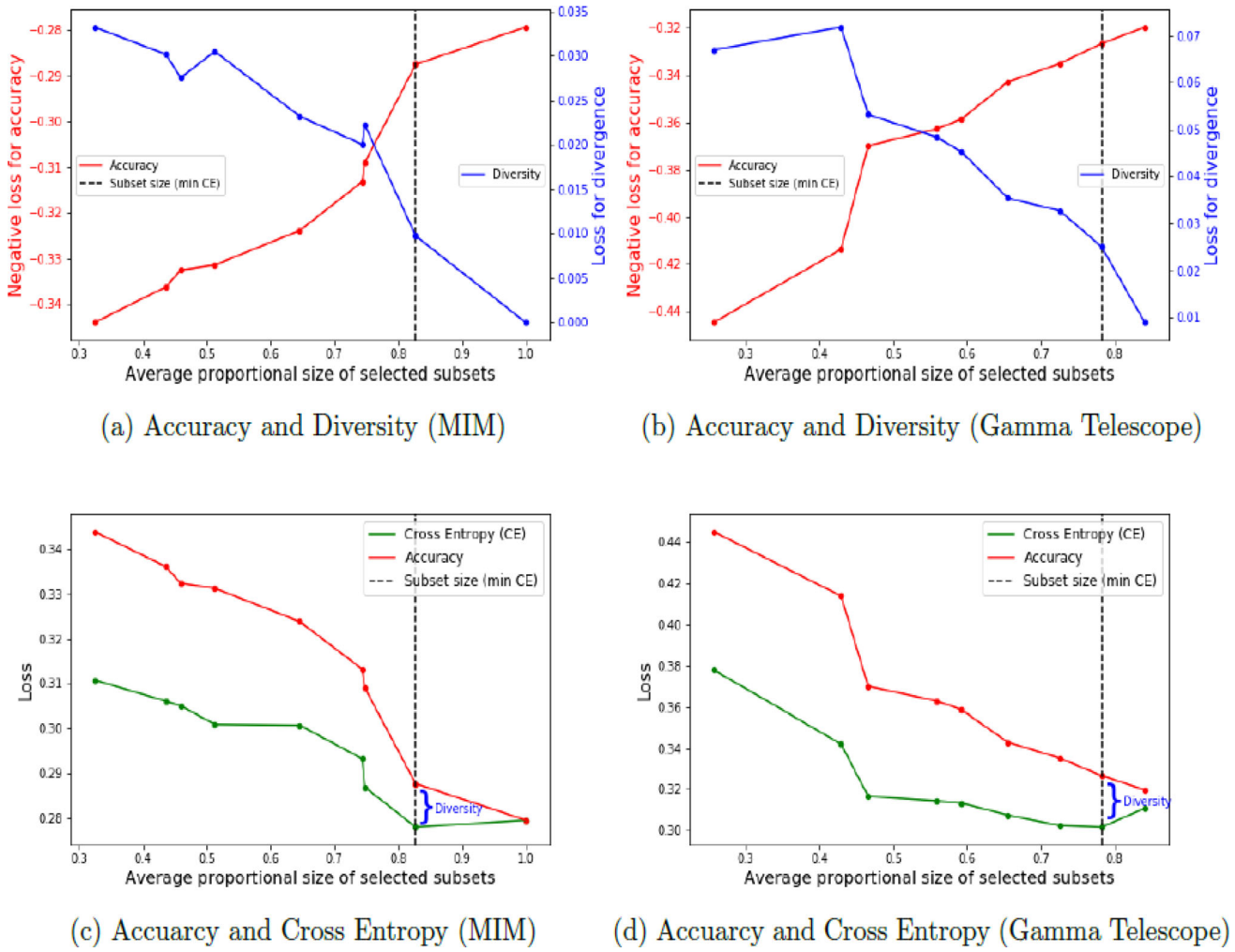


Fig. 8 Loss decomposition for classification

the strength of diversity, we explore the overall loss, the weighted sum individual base learner accuracy, and the ambiguity measure against the average subset size over all the single-hidden-layer NNs in the first step of LIFE. Here, we vary the subset size by controlling the cutoff point *cp* for linear project. Figures 7 and 8 show the relationship between average subset size (in terms of the proportion of original train data size) and loss for accuracy, loss for diversity, and MSE loss in both regression and classification cases. In plot (a) and (b), the blue curves indicate the ambiguity diversity measure have a decreasing trend when the subset size increases. This is consistent with the intuition that the larger overlapping the subsets are, the less diverse the base learners are. On the other hand, the accuracy of the individual base learners is higher when subset size is larger, as the sample is more representative of the whole training set. Similarly, training on smaller subsets has stronger diversity but leads to lower accuracy. Plots (c) and (d) illustrate the trade-off between average accuracy and diversity to minimize total loss, where the dash line shows the optimal subset size achieving the minimum loss. Combining the results from Sects. 2.3.1 and 2.3.2, we conclude that competitive predictive performance of LIFE benefits from diversity due to data sampling in the first step and the feature 'flattening' and joint estimation in the third step.

3 Empirical experiment

In this section, we conduct multiple empirical experiments via both simulated and real data for regression and classification cases to confirm the competitive performance of LIFE. We have generated multiple datasets with Normal distribution and heavy-tailed predictor distribution (Laplace distribution), as well as different function forms to analyze predictive performance and computational efficiency. All datasets are split into 80% training data and 20% testing data. Other benchmark models including single-hidden-layer NN trained by different optimizer (local

Table 2 Regression on simulated data (normal distribution)

Model	Metric	Oracle	LIFE (LLA)	LLA	LIFE (Adam)	Adam	FFNN	Xgboost	RF
GAM	*RMSE*	1.000	**1.046**	1.062	1.063	1.149	1.104	1.065	1.906
			(0.012)	(0.040)	(0.016)	(0.023)	(0.025)	(0.020)	(0.037)
	R^2	0.965	**0.962**	0.961	0.961	0.954	0.957	0.960	0.873
			(0.001)	(0.003)	(0.001)	(0.002)	(0.002)	(0.002)	(0.005)
	T	N/A	75s	120s	46s	55s	81s	**8s**	44s
AIM	*RMSE*	1.000	**1.077**	1.092	1.153	1.202	1.159	2.606	3.285
			(0.018)	(0.032)	(0.029)	(0.040)	(0.033)	(0.173)	(0.161)
	R^2	0.984	**0.981**	0.980	0.978	0.975	0.978	0.886	0.828
			(0.001)	(0.001)	(0.001)	(0.004)	(0.001)	(0.010)	(0.010)
	T	N/A	**7s**	51s	17s	40s	46s	27s	48s
MIM	*RMSE*	1.000	**1.028**	1.043	1.036	1.047	1.037	1.057	1.139
			(0.005)	(0.003)	(0.006)	(0.004)	(0.009)	(0.004)	(0.011)
	R^2	0.937	**0.933**	0.931	0.931	0.930	0.931	0.929	0.918
			(0.003)	(0.003)	(0.003)	(0.003)	(0.007)	(0.003)	(0.008)
	T	N/A	29s	313s	81s	45s	51s	**14s**	44s

Columns 4–7 represent just single-hidden-layer NNs optimized by LIFE (LLA), LLA, LIFE (Adam), and Adam, while there are three state-of-art machine learning methods including FFNN, Xgboost, and random forest (RF) on the right side. FFNN is multi-hidden-layer feed forward NN, where range for number of hidden layers is from two to four. In addition, Xgboost and RF are two tree-based ensemble methods. The figures inside parenthesis indicate the standard deviation of metrics, and time represents training time of one replication. The numbers in bold represent the optimal results of this metric

Table 3 Regression on simulated data (laplace distribution)

Model	Metric	Oracle	LIFE (LLA)	LLA	LIFE (Adam)	Adam	FFNN	Xgboost	RF
GAM	*RMSE*	1.000	**1.166**	1.258	1.427	1.629	1.460	1.439	3.242
			(0.079)	(0.091)	(0.192)	(0.100)	(0.205)	(0.256)	(0.226)
	R^2	0.992	**0.989**	0.987	0.984	0.979	0.983	0.983	0.920
			(0.001)	(0.001)	(0.003)	(0.002)	(0.004)	(0.005)	(0.005)
	T	N/A	48s	84s	19s	96s	75s	**5s**	20s
AIM	*RMSE*	1.000	**1.057**	1.084	1.122	1.192	1.166	2.031	2.491
			(0.026)	(0.040)	(0.055)	(0.084)	(0.077)	(0.099)	(0.204)
	R^2	0.968	**0.966**	0.964	0.962	0.957	0.959	0.876	0.804
			(0.004)	(0.005)	(0.003)	(0.004)	(0.004)	(0.012)	(0.035)
	T	N/A	10s	61s	22s	59s	26s	**8s**	56s
MIM	*RMSE*	1.000	**1.060**	1.110	1.060	1.096	1.088	1.099	1.555
			(0.017)	(0.065)	(0.018)	(0.026)	(0.018)	(0.013)	(0.081)
	R^2	0.969	**0.965**	0.961	0.965	0.962	0.963	0.962	0.925
			(0.001)	(0.005)	(0.001)	(0.002)	(0.001)	(0.001)	(0.008)
	T	N/A	20s	195s	39s	42s	32s	**6s**	43s

Columns 4–7 represent just single-hidden-layer NNs optimized by LIFE (LLA), LLA, LIFE (Adam), and Adam, while there are three state-of-art machine learning methods including FFNN, Xgboost, and random forest (RF) on the right side. FFNN is multi-hidden-layer feed forward NN, where range for number of hidden layers is from two to four. In addition, Xgboost and RF are two tree-based ensemble methods. The figures inside parenthesis indicate the standard deviation of metrics, and time represents training time of one replication. The numbers in bold represent the optimal results of this metric

linear approximation and Adam algorithm), and other machine learning algorithms including multilayer FFNN, Xgboost, and random forest are tested on the same data for comparison after extensive hyper-parameter tuning. Local linear approximation (LLA) algorithm is a recently proposed method to estimate the weights and biases of single-

Table 4 Classification on simulated data (normal distribution)

Model	Metric	Oracle	LIFE (LLA)	LLA	LIFE (Adam)	Adam	FFNN	Xgboost	RF
GAM	*AUC*	0.984	0.974	0.968	0.973	0.945	0.973	**0.975**	0.964
		(0.002)	(0.002)	(0.003)	(0.002)	(0.003)	(0.002)	(0.002)	(0.002)
	logloss	0.113	0.146	0.158	0.148	0.247	0.152	**0.141**	0.166
		(0.005)	(0.004)	(0.010)	(0.005)	(0.029)	(0.005)	(0.006)	0.006
AIM	*AUC*	0.836	0.751	0.535	**0.752**	0.742	**0.752**	0.687	0.715
		(0.008)	(0.010)	(0.037)	(0.010)	(0.010)	(0.010)	(0.010)	(0.011)
	logloss	0.359	**0.422**	0.485	0.428	0.448	0.431	0.457	0.447
		(0.005)	(0.003)	(0.005)	(0.006)	(0.010)	(0.004)	(0.001)	(0.002)
MIM	*AUC*	0.888	**0.839**	0.838	0.838	0.830	**0.839**	0.837	0.834
		(0.005)	(0.008)	(0.007)	(0.008)	(0.008)	(0.008)	(0.008)	(0.007)
	logloss	0.331	**0.386**	0.388	0.387	0.399	0.387	0.388	0.391
		(0.010)	(0.010)	(0.009)	(0.010)	(0.011)	(0.009)	(0.010)	(0.009)

Columns 4–7 represent just single-hidden-layer NNs optimized by LIFE (LLA), LLA, LIFE (Adam), and Adam, while there are three state-of-art machine learning methods including FFNN, Xgboost, and random forest (RF) on the right side. FFNN is multi-hidden-layer feed forward NN, where range for number of hidden layers is from two to four. In addition, Xgboost and RF are two tree-based ensemble methods. The figures inside parenthesis indicate the standard deviation of metrics. The numbers in bold represent the optimal results of this metric

Table 5 Classification on simulated data (laplace distribution)

Model	Metric	Oracle	LIFE (LLA)	LLA	LIFE (Adam)	Adam	FFNN	Xgboost	RF
GAM	*AUC*	0.996	0.990	0.984	0.990	0.857	0.976	**0.991**	0.979
		(0.000)	(0.001)	(0.003)	(0.001)	(0.012)	(0.002)	(0.001)	(0.002)
	logloss	0.051	0.081	0.100	0.081	0.252	0.133	**0.079**	0.120
		(0.002)	(0.003)	(0.006)	(0.003)	(0.012)	(0.004)	(0.004)	(0.003)
AIM	*AUC*	0.866	**0.802**	0.797	**0.802**	0.795	**0.802**	0.737	0.743
		(0.012)	(0.018)	(0.020)	(0.018)	(0.021)	(0.017)	(0.014)	(0.015)
	logloss	0.230	**0.270**	0.274	0.271	0.281	0.273	0.298	0.296
		(0.006)	(0.008)	(0.012)	(0.008)	(0.007)	(0.007)	(0.004)	(0.004)
raisebox1.5exMIM	*AUC*	0.958	0.939	0.938	**0.940**	0.920	0.937	0.938	0.934
		(0.003)	(0.004)	(0.003)	(0.004)	(0.003)	(0.002)	(0.004)	(0.004)
	logloss	0.225	**0.268**	0.272	**0.268**	0.421	0.276	0.271	0.279
		(0.011)	(0.012)	(0.011)	(0.012)	(0.075)	(0.010)	(0.011)	(0.010)

Columns 4–7 represent just single-hidden-layer NNs optimized by LIFE (LLA), LLA, LIFE (Adam), and Adam, while there are three state-of-art machine learning methods including FFNN, Xgboost, and random forest (RF) on the right side. FFNN is multi-hidden-layer feed forward NN, where range for number of hidden layers is from two to four. In addition, Xgboost and RF are two tree-based ensemble methods. The figures inside parenthesis indicate the standard deviation of metrics. The numbers in bold represent the optimal results of this metric

hidden-layer NN by iterative linear regression and linear approximation of the ReLU activation function [26]. The LLA algorithm is distinguished from existing gradient descent algorithms in that it utilizes the Hessian matrix in the same spirit of Fisher scoring algorithm for nonlinear regression models with normal error. The outline of the LLA algorithm is included in the "Appendix".

3.1 Simulated data

3.1.1 Regression

For the regression scenario, there are three different function forms including generalized additive model (GAM), additive index model (AIM), and multiple index model (MIM), which are expressed, as follows:

$$
\begin{aligned}
GAM: y_i &= \beta_1 x_{1i} + \beta_2\sqrt{|x_{2i}|} + \beta_3|x_{3i}| \\
&\quad + \beta_4 exp(x_{4i}) + \beta_5 log(|x_{5i}|) \\
&\quad + \beta_6 max(1, x_{6i}) + \epsilon_i, \\
\beta &= \{\beta_1, \cdots, \beta_6\} \\
&= \{1.5, \sqrt{5}, 2, 4e^{(\frac{-1.5}{7})}, 4log(1.5), -4\}, \\
\epsilon_i &\sim N(0,1), i = 1, \ldots, N,
\end{aligned} \tag{8}
$$

$$
\begin{aligned}
AIM: y_i &= 2log(|\beta_1 x_{1i} + \cdots + \beta_4 x_{4i}|) \\
&\quad + exp(\frac{\beta_3 x_{3i} + \cdots + \beta_6 x_{6i}}{9}) \\
&\quad + max(0, \beta_5 x_{5i} + \beta_6 x_{6i}) + \epsilon_i, \\
\beta &= \{\beta_1, \cdots, \beta_6\} \\
&= \{3, -2.5, 2, -1.5, 1.5, -1\},
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
MIM: y_i &= exp(\beta_1 x_{1i} + \beta_2 x_{2i})\beta_3 x_{3i} \\
&\quad + \frac{\beta_4 x_{4i}}{1 + \beta_5|x_{5i}|} + max(2, \beta_6 x_{6i}) + \epsilon_i, \\
\beta &= \{\beta_1, \cdots, \beta_6\} \\
&= \{0.03, -0.025, 1, -3, 1.5, -2\},
\end{aligned} \tag{10}
$$

where $N = 20k$ and all predictors $\{x_{ji}\}_{j=1,\cdots,6;i=1,\cdots,N}$ are drawn from Normal or Laplace distribution. For regression, the experimental results show mean and standard deviation of *RMSE*, R^2 and training time T over five replications in Tables 2 and 3, while logloss and AUC are used as a performance metric in the classification case as shown in Tables 4 and 5. Bayesian optimization allows us to jointly tune more parameters with fewer experiments and find better values, so we implement it to perform extensive hyper-parameter tuning on all the algorithm. The important hyper-parameters for LIFE include the number of iterations, the number of neurons in each iteration, upper and

Table 6 Regression on real data

Data	Metric	LIFE (LLA)	LLA	LIFE (Adam)	Adam	FFNN	Xgboost	RF
Abalone	RMSE	**2.112 ± 0.077**	2.162 ± 0.074	2.137 ± 0.071	2.179 ± 0.108	2.195 ± 0.133	2.169 ± 0.068	2.170 ± 0.071
	R^2	**0.570 ± 0.031**	0.549 ± 0.031	0.560 ± 0.029	0.543 ± 0.057	0.534 ± 0.047	0.546 ± 0.028	0.546 ± 0.029
	T	2s	289s	**1s**	35s	17s	**1s**	**1s**
Airfoil	RMSE	**0.225 ± 0.020**	0.288 ± 0.021	0.276 ± 0.026	0.293 ± 0.018	0.270 ± 0.034	0.238 ± 0.016	0.343 ± 0.023
	R^2	**0.949 ± 0.009**	0.917 ± 0.012	0.923 ± 0.014	0.913 ± 0.011	0.925 ± 0.019	0.943 ± 0.008	0.881 ± 0.016
	T	**1s**	14s	**1s**	8s	**1s**	**1s**	**1s**
Aquatic	RMSE	**1.193 ± 0.066**	1.220 ± 0.045	1.210 ± 0.068	1.223 ± 0.096	1.255 ± 0.125	1.214 ± 0.101	1.204 ± 0.082
Toxicity	R^2	**0.484 ± 0.058**	0.461 ± 0.039	0.469 ± 0.060	0.455 ± 0.088	0.425 ± 0.117	0.464 ± 0.089	0.473 ± 0.071
	T	**1s**	41s	**1s**	2s	**1s**	**1s**	**1s**
Bike	RMSE	41.98 ± 1.299	47.66 ± 0.946	45.34 ± 1.103	50.74 ± 1.601	45.99 ± 1.737	**41.71 ± 1.184**	66.38 ± 1.257
Sharing	R^2	0.946 ± 0.003	0.930 ± 0.003	0.937 ± 0.003	0.922 ± 0.005	0.935 ± 0.005	**0.947 ± 0.003**	0.866 ± 0.005
	T	85s	2194s	81s	266s	114	13s	**12s**
California	RMSE	0.500 ± 0.012	0.527 ± 0.011	0.521 ± 0.009	0.535 ± 0.008	0.519 ± 0.011	**0.479 ± 0.007**	0.486 ± 0.007
Housing	R^2	0.812 ± 0.009	0.791 ± 0.008	0.796 ± 0.007	0.784 ± 0.007	0.797 ± 0.008	**0.825 ± 0.005**	0.822 ± 0.005
	T	13s	108s	**8s**	74s	40s	12s	40s
CASP	RMSE	3.865 ± 0.034	4.111 ± 0.049	4.081 ± 0.047	4.133 ± 0.071	4.067 ± 0.037	**3.786 ± 0.022**	3.951 ± 0.016
	R^2	0.601 ± 0.007	0.548 ± 0.011	0.555 ± 0.010	0.543 ± 0.016	0.558 ± 0.008	**0.617 ± 0.004**	0.583 ± 0.004
	T	216s	2881s	**54s**	339s	83s	60s	56s
Electrical	*RMSE*	**0.007 ± 0.000**	0.011 ± 0.001	0.008 ± 0.000	0.009 ± 0.001	0.009 ± 0.001	0.011 ± 0.000	0.013 ± 0.000
Grid	R^2	**0.959 ± 0.003**	0.908 ± 0.016	0.949 ± 0.003	0.944 ± 0.009	0.945 ± 0.011	0.908 ± 0.003	0.877 ± 0.004)
	T	25s	27s	**4s**	27s	17s	16s	18s

Columns 3–6 represent just single-hidden-layer NNs optimized by LIFE (LLA), LLA, LIFE (Adam), and Adam, while there are three state-of-art machine learning methods including FFNN, Xgboost, and random forest (RF) on the right side. FFNN is multi-hidden-layer feed forward NN, where range for number of hidden layers is from two to four. In addition, Xgboost and RF are two tree-based ensemble methods. The figures after '±' sign indicate the standard deviation of metrics, and time represents training time of one replication. The numbers in bold represent the optimal results of this metric.

lower bound. We marked optimal results that have won the campaign in bold.

As illustrated in Tables 2 and 3, the result is predictable regardless of the distribution predictors drawn. LIFE algorithm with LLA optimizer achieves higher accuracy among all methods in terms of predictive performance on the test set. The values of two metrics from LIFE (LLA) are close to oracle values, which implies LIFE performs well in the data with a smoothing response surface. If we compare results from one-hidden-layer FFNNs trained by LIFE algorithm with either LLA or Adam base learners and non-ensemble algorithms of LLA or Adam, LIFE always outperforms the relevant optimization methods used to train single-hidden-layer FFNN as a whole due to the generated diversity of data sampling. In addition, the performance of LIFE also depends on the strength of individual NN base learner, which can be easily spotted in Tables 2 and 3 that LIFE (LLA) outperforms LIFE (Adam). From the perspective of computational efficiency, LIFE algorithm also shows some advantages over other single-hidden-layer NN training algorithms. In general, LIFE algorithm can not only boost predictive performance of one-hidden-layer NN, but also speed up training, especially with respect to wide NN with large hidden-layer dimension.

3.1.2 Classification

For classification case, the functional forms in simulation setup are similar to the ones in regression case except that the coefficients are a little bit different. Detailed information on formulas can be found in "Appendix". Similar to the setup in regression case, we choose $N = 20k$ and all predictors are drawn from either Normal or Laplace distribution. The response variable is sampled from Bernoulli distribution with probability calculated using the logit link function.

Tables 4 and 5 show the simulation results from binary scenario, where data are generated from Normal distribution and Laplace distribution, respectively. Similar to the results in regression case, LIFE (LLA) has won the campaign in four out of six functional forms. LIFE (Adam) also performs pretty well especially in Table 5 for Laplace distribution. For data drawn from Normal distribution shown in Table 4, LIFE (Adam) is also quite close to the optimal result. Furthermore, there is a strong evidence to show LIFE algorithm does improve the performance of base learners with larger AUC, smaller logloss and smaller standard errors of both metrics. Even if the base learner is not strong enough, like Adam for GAM and MIM in Laplace distribution (Table 5), with which AUC or logloss or both has large standard error, the ensemble approach in

Table 7 Classification on real data

Data	Metric	LIFE (LLA)	LLA	LIFE (Adam)	Adam	FFNN	Xgboost	RF
Bank	AUC	0.796 ± 0.006	0.790 ± 0.007	0.796 ± 0.006	0.793 ± 0.006	0.794 ± 0.006	0.799 ± 0.008	$\mathbf{0.800 \pm 0.005}$
Market	Logloss	0.288 ± 0.003	0.292 ± 0.003	0.289 ± 0.004	0.294 ± 0.003	0.288 ± 0.003	0.288 ± 0.003	$\mathbf{0.286 \pm 0.003}$
Breast	AUC	$\mathbf{0.997 \pm 0.002}$	0.991 ± 0.009	0.996 ± 0.004	0.994 ± 0.005	0.996 ± 0.004	0.992 ± 0.006	0.993 ± 0.006
Cancer	Logloss	0.096 ± 0.015	0.143 ± 0.088	$\mathbf{0.084 \pm 0.027}$	0.340 ± 0.207	0.099 ± 0.014	0.114 ± 0.042	0.090 ± 0.019
Wisc.								
Higgs	AUC	0.912 ± 0.001	0.904 ± 0.002	$\mathbf{0.914 \pm 0.001}$	0.891 ± 0.003	0.912 ± 0.001	0.911 ± 0.001	0.911 ± 0.001
Boson	Logloss	0.353 ± 0.001	0.368 ± 0.004	$\mathbf{0.348 \pm 0.001}$	0.399 ± 0.005	0.353 ± 0.001	0.355 ± 0.001	0.357 ± 0.001
Home	AUC	0.853 ± 0.005	0.841 ± 0.003	$\mathbf{0.860 \pm 0.003}$	0.848 ± 0.006	0.858 ± 0.003	0.858 ± 0.003	0.854 ± 0.003
Lend	Logloss	$\mathbf{0.046 \pm 0.001}$	0.047 ± 0.001	$\mathbf{0.046 \pm 0.001}$	0.047 ± 0.001	$\mathbf{0.046 \pm 0.001}$	$\mathbf{0.046 \pm 0.001}$	$\mathbf{0.046 \pm 0.001}$
MAGIC	AUC	$\mathbf{0.943 \pm 0.002}$	0.928 ± 0.003	0.938 ± 0.003	0.909 ± 0.008	0.938 ± 0.003	0.936 ± 0.002	0.937 ± 0.003
Gamma	Logloss	$\mathbf{0.275 \pm 0.007}$	0.311 ± 0.007	0.288 ± 0.007	0.371 ± 0.019	0.287 ± 0.006	0.291 ± 0.005	0.296 ± 0.005
Telescope								
Mush	AUC	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$	$\mathbf{1 \pm 0}$
Room	Logloss	0.001 ± 0	$\mathbf{0.000 \pm 0}$	0.001 ± 0.001	0.024 ± 0.036	0.004 ± 0.003	0.004 ± 0.001	0.005 ± 0.000
Ring	AUC	$\mathbf{0.998 \pm 0.001}$	0.997 ± 0.001	$\mathbf{0.998 \pm 0.001}$	0.995 ± 0.002	0.997 ± 0.001	$\mathbf{0.998 \pm 0.000}$	0.995 ± 0.001
Rorm	Logloss	$\mathbf{0.049 \pm 0.006}$	0.086 ± 0.020	0.051 ± 0.007	0.147 ± 0.009	0.075 ± 0.015	0.057 ± 0.004	0.162 ± 0.006

Columns 3–6 represent just single-hidden-layer NNs optimized by LIFE (LLA), LLA, LIFE (Adam), and Adam, while there are three state-of-art machine learning methods including FFNN, Xgboost, and random forest (RF) on the right side. FFNN is multi-hidden-layer feed forward NN, where range for number of hidden layers is from two to four. In addition, Xgboost and RF are two tree-based ensemble methods. The figures after '±' sign indicate the standard deviation of metrics. The numbers in bold represent the optimal results of this metric

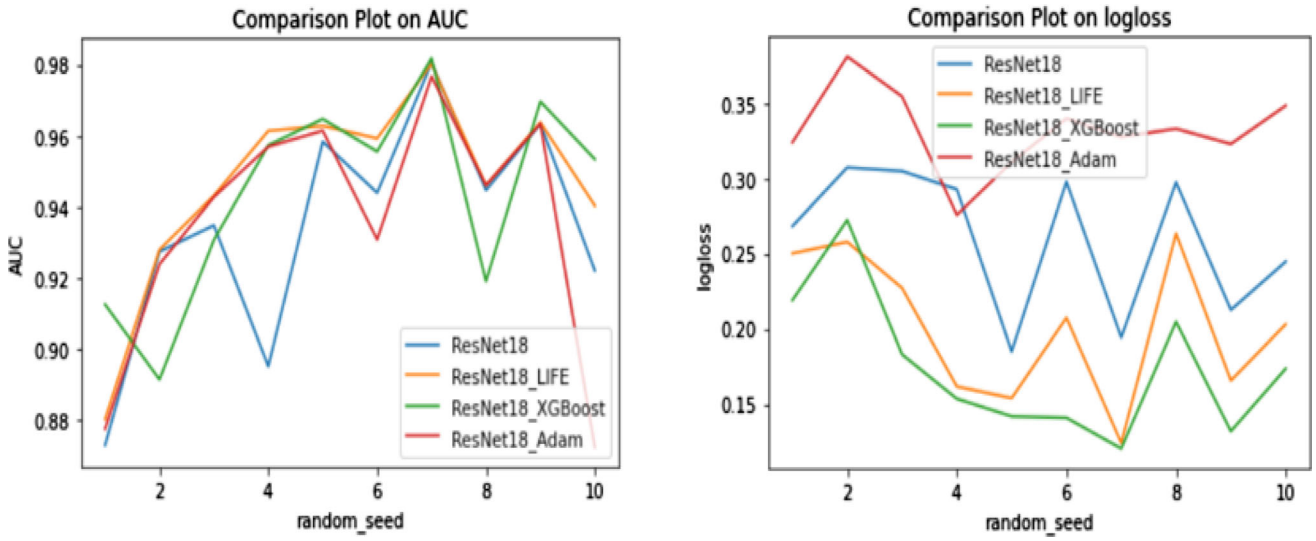


Fig. 9 ResNet18 vs. its potential improvers

Table 8 ResNet18 vs. its potential improvers

Data	Metric	ResNet18	ResNet18_LIFE	ResNet18_Xgboost	ResNet18_Adam
MNIST	*AUC*	0.934	**0.947**	0.944	0.935
		(0.030)	(0.026)	(0.027)	(0.034)
	Logloss	0.261	0.202	**0.174**	0.332
		(0.046)	(0.046)	(0.045)	(0.027)

LIFE (Adam) can dramatically reduce the variance. Xgboost ranks at top for GAM in both distributions, however, the differences between LIFE and Xgboost are negligible with only 0.1% of difference in AUC and 3% of difference in logloss.

3.2 Real data

Besides implementing LIFE algorithm on simulated data, we also tested it on seven public datasets for regression and eight datasets for classification and compared it with other benchmark models including single-hidden-layer FFNN and Xgboost. All datasets are split into 80% training data and 20% testing data with 10 different random seeds, which yield results over 10 replications. For all the datasets, we transformed categorical variables into dummy variables and standardized the continuous variables, so that the mean and the variance of each continuous variable are equal to 0 and 1, respectively. A detailed description of all datasets and corresponding data preprocessing steps are outlined in the "Appendix".

3.2.1 Regression

The experiment results averaged over ten replications are reported in Table 6, including root mean squared error (*RMSE*), R-squared (R^2), and training time (T).

As observed in Table 6, LIFE (LLA) is ranked as the best algorithm in the four datasets. For the remaining three datasets, LIFE (LLA) is still the second or third best algorithm among all models with a close or slightly worse predictive performance than optimal one (Xgboost or random forest), which implies that the LIFE algorithm is competitive with other state-of-art machine learning algorithms. In addition, there is an average 4.6% or 1.8% improvement in R-square of all real datasets when single-hidden-layer NN is trained by LIFE (LLA or Adam) instead of other optimization methods (LLA or Adam), which is consistent with the conclusion made from experiment in the simulated data. It is also worth mentioning that computation efficiency of NN training has been significantly boosted for almost all dataset via LIFE compared with traditional NN training methods. In particular, if we take a look at the largest real dataset CASP, training time of NN via LIFE reduces to 216 seconds from 2881 seconds or to 54 seconds from 339 seconds when we use LLA as optimizer or Adam respectively, which is almost more than six times faster. Although tree-based ensemble methods

such as Xgboost and random forest show strong predictive power in some datasets, they are still black-box models, and they are hard to interpret. The biggest advantage of our proposed algorithm LIFE is that it preserves the interpretability of model, which is still single-hidden-layer NN with very strong predictive performance and boosted computation efficiency.

3.2.2 Classification

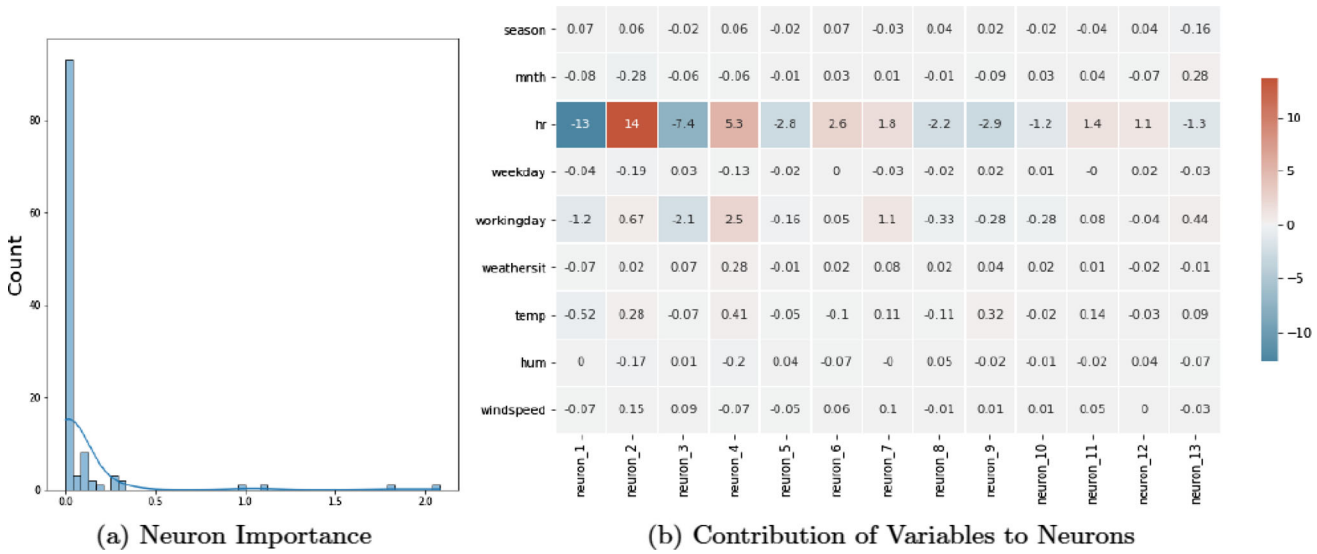
In the classification case, original LLA algorithm is not stable, since it involves matrix inversion. We added a ridge parameter into the matrix inversion and treat it as a hyperparameter in LLA algorithm. Experiments have shown adding ridge parameter in LLA can give better and more stable prediction than not adding ridge parameter. After testing LLA and LIFE (LLA) with or without ridge parameter, we further choose the best one for the performance.

Table 7 presents similar patterns in real data analyses as in simulation studies with LIFE (LLA) and LIFE (Adam) taking turns to occupy the dominant position for most of the datasets. LIFE performs much better than Xgboost and random forest (RF) in most experiments. For example, with Breast Cancer Wisconsin data, logloss in LIFE (Adam) is 26.3% lower than that in Xgboost, and with MAGIC Gamma Telescope data, logloss has dropped by 7% from random forest to LIFE (LLA). The performance of some datasets, such as Bank Marketing data, where LIFE cannot outperform Xgboost or RF, however, the performance is competitive, with only 0.5% and 0.7% of difference in AUC and logloss, respectively. Another aspect worth mentioning is that Higgs Boson data contains quite a few highly correlated variables. The results show that LIFE algorithm outperforms all the rest of models, which indicates LIFE really does an excellent job in predicting on highly correlated structures.

We have also investigated whether LIFE algorithm can further improve the performance of trained deep neural network on image data. We here use ResNet18 proposed by (He et al. 2016 [27]) as an example and apply the algorithms on MNIST data [28]. The detailed information of data preprocessing can be found in Case 8 from the description lists of real datasets 6. After training MNIST data using ResNet18, the output of final convolutional layer has been extracted, which has size 8000×512, and it is also the input of feed forward neural network (FFNN) in the final step of ResNet18. This 8000×512 data is then treated as the input of LIFE (Adam). Further, we also attach Xgboost, Adam to ResNet18 and compare the results.

3.2.3 Classification on image data

Figure 9 shows the performance of LIFE (orange line) is better than ResNet18 (blue line) with consistently larger AUC and smaller logloss in all of the 10 replications. LIFE is comparable to Xgboost in terms of logloss in general with all 10 values below ResNet18. However, in terms of AUC, Xgboost is worse than ResNet18 with significantly lower AUC in two replications (seed = 2 and seed = 8). On the other hand, Adam (red line) is not able to further improve the performance of ResNet18, which is consistent with what we have discovered in the previous empirical



(a) Neuron Importance (b) Contribution of Variables to Neurons

Fig. 10 Variable importance detection (LIFE)

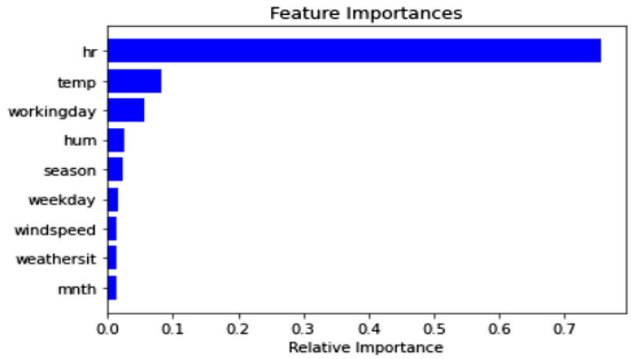


Fig. 11 Variable importance detection (random forest)

studies. Table 8 also indicates LIFE and Xgboost are both capable of remarkably enhancing a trained deep NN with similar performance. One last discovery is since the input of LIFE contains 512 columns, it also indicates that LIFE can handle high dimensionality quite well in terms of prediction.

4 Interpretation

Interpretability is the degree to which one human being can understand the cause of a decision or predict the result of a model. The higher the interpretability of a machine learning or deep learning model, the easier it is for someone to comprehend why certain decisions or predictions have been made. A key advantage of LIFE is that it is still an inherently interpretable model. From the perspective of the NN structure, the model is a single-hidden-layer NN with ReLU activation function where all the weights and bias can be easily extracted and visualized. Moreover, the single-layer NN with ReLU activation function can be rewritten in the form of local linear model representation, and be interpreted by exploring the patterns of local linear model coefficients. Finally, the main and interaction effects can be identified by exploring and aggregating the local linear coefficients.

We use the bike sharing data result as an example to illustrate the intrinsic interpretability of LIFE. Bike sharing data is a public dataset hosted on UCI machine learning repository, where there are around 17, 000 observations on hourly (and daily) bike rental counts along with weather and time information between 2011 and 2012 in the Capital Bikeshare system. Out of the original 17 predictors, we removed some non-meaningful and highly correlated ones, leaving us with 9 predictors to predict hourly rental counts. At the tiny expense of predictive performance, we applied both the base learner selection method shown in Algorithm 3 in Sect. 5.2 and elastic net to reduce the number of base learners and features so that the final single-hidden-layer NN has a small number of significant neurons and is easier for interpretation.

4.1 Explore the weights and bias of single-layer NN

As LIFE finally generates a single-hidden-layer NN in the third step, we can explore the weights $\hat{w}_k$s and bias $\hat{b}_k$s of the NN directly and identify which variable is important. For bike sharing data, there are finally 116 new features (or neurons) after base learner selection and elastic net regularization. We measure the neuron importance by $std(\hat{\beta}_k \sigma(\hat{b}_k + x^T \hat{w}_k))/std(\hat{f})$, and $\hat{f}$, where std is the standard deviation, $\hat{f}$ is the predicted value of response variable for regression or log-odds for classification, $\hat{w}_k$s is the neuron weight, and $\hat{b}_k$ is the coefficient for neuron. This quantity measures the importance of neurons/feature by comparing the variation of each feature to the total variance. The histogram on the neuron importance for the 116 features in Fig. 10 shows that there are only 13 neurons whose importance values are greater than 2% of the maximum importance.

Then, we can detect how each variable contributes to each neuron by applying the following measurement:

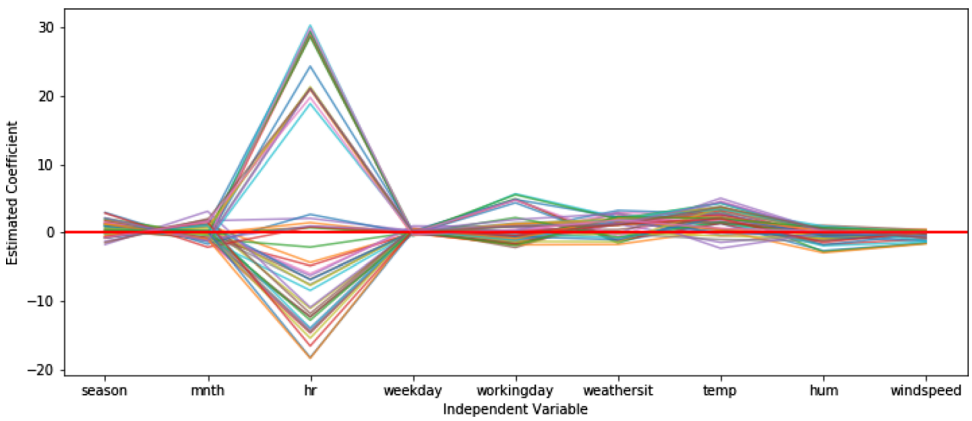


Fig. 12 Parallel coordinates plot for bikesharing data

$$\hat{w}_k \hat{\beta}_k \frac{std(\sigma(\hat{b_k} + x^T \hat{w}_k))}{std(\hat{f})}$$

where we allocate neuron importance to each variable by multiplying $\hat{w}_k$. This contribution measurement can be simply visualized by heatmap between neurons and original variables in Fig. 10. It shows that hour (*hr*) and working day (*workingday*) are top significant variables with darker colors for almost each important neuron compared with other variables. Another variable temperature (*temp*) can also be considered to relatively important except *hr* and *workingday*.

Variable importance detection using random forest in Fig. 11 shows similar findings, with hour (*hr*) having the highest relative importance score, followed by temperature (*temp*) and working day (*workingday*).

4.2 Treat a single-layer NN as a local linear model

As we may have many features in the final wide single-layer NN, it is difficult to visualize and explore the weights and bias of all the neurons. Hence, we also propose to interpret single-layer NN from local linear model perspective. Single-layer NN with ReLU function can be considered a type of local linear model. Each linear projection would determine the active or inactive states of the ReLU neurons at hidden layers, which define the layered pattern. The activation region is constructed as a combination of those distinct patterns. Those activation regions are mutually exclusive and regarded as convex polytopes with closed-form boundaries [29]. A linear equation can be used in all data points inside the activation region to represent the relationships between response and independent variables. After defining the region each observation belongs to, we can easily extract a linear equation for each region based on estimated weights in the hidden and output layers. The detailed algorithm that performs a linear equation extraction the following:

Algorithm 2 Local Linear Equation Extraction

Input: Estimated weights and biases $\{\hat{w}_k, \hat{b}_k\hat{\beta}_k\}_{k=1,\cdots,m_J}$; τ: threshold for the number of observations
Output: Coefficients of local linear equations $\{\hat{E}_t\}_{t=1,\cdots,M}$
Data: Independent variables $\{x_i\}_{i=1,\cdots,N}$
Determine if i^{th} observation is active or not in each neuron based on linear projection $\hat{b}_k + x_i^T \hat{w}_k > 0$
Combine observations into M homogenous activation local regions $R_t, t = 1, \cdots, M$, depending on if they have the same set of active neurons (boundary condition)
for $t = 1, \cdots, M$ **do**
  if $\sum_{x_i \in R_t} i > \tau$ **then**
    i. Construct activation set A_t of the local region R_t relying on its boundary condition
    ii. Calculate coefficients: $\hat{E}_t = \sum_{k \in R_t} \hat{w}_k * \hat{\beta}_k$
  end if
end for

We can visualize those linear equations by a parallel coordinate plot, which allows comparing the estimated coefficients of all predictors for different local linear regions. Through the visualization of local linear equations, we can not only have an overview of the importance of each predictor in each region by comparing the magnitude of coefficients, but also check the validity for effect of each predictor on the response variables. It is worth mentioning that those coefficients are comparable after standardizing all the predictors. There are three scenario for a particular independent variable:

1. Relatively large coefficients of the variable, compared with others in terms of absolute values and have the same signs, imply that this variable has a significant positive or negative effect on the response variable if all coefficients are positive or negative.
2. Relatively large coefficients of the variable with both positive and negative signs strongly imply that this variable has inconsistent slopes across local activation regions, which might be due to either its own nonlinear

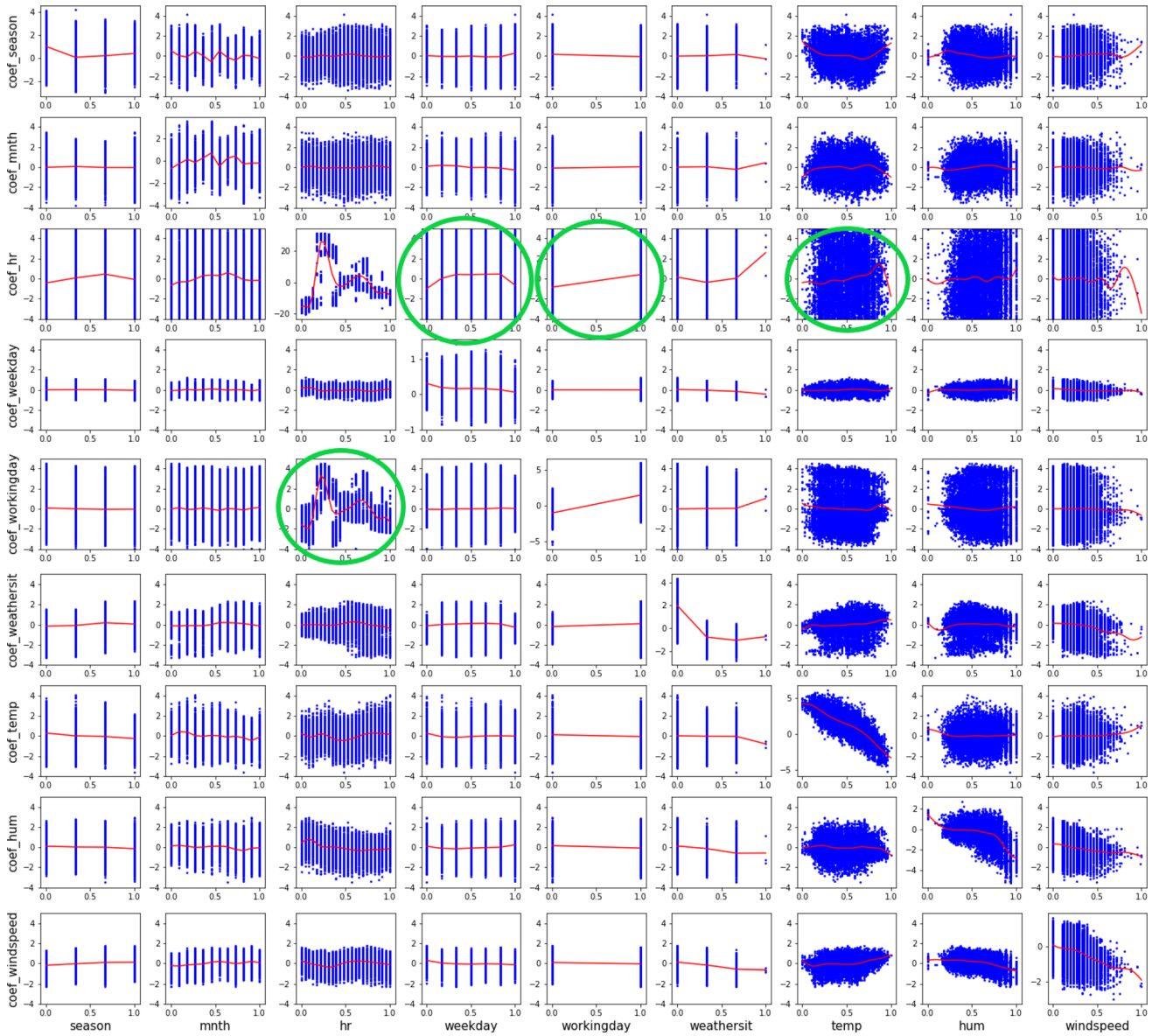


Fig. 13 Plot matrix between α_m and x_m

main effect or the interaction effects with other variables.

3. Small and close-to-zero coefficients indicate that this feature is not important to explain the variation of the response variable and can be removed from the model.

Furthermore, we were able to verify if the sign of estimated coefficients of predictor in all regions is consistent with domain knowledge or business sense. Figure 12 displays the estimated coefficients of all predictors in the local activation regions for bike sharing dataset. There are 116 neurons extracted from NN base learners and 47 local regions created by Algorithm 2 and each local region has at least three data points.

It clearly indicates that hour (*hr*), working day (*workingday*) and temperature (*temp*) are the three most important predictors with relatively higher absolute values of their corresponding coefficients in several local regions, which is pretty consistent with result from Fig. 11. Their estimated coefficients present different directions across local activation regions, which is consistent with our second scenario.

This gives us a hint of interactions between those variables. Other variables such as humidity (*hum*) and wind speed (*windspeed*) are insignificant based on their absolute values of estimation coefficients from the plot. Sometimes there are too many local regions and (Sudjianto et al.(2020)) [29] provides two approaches to simplify and

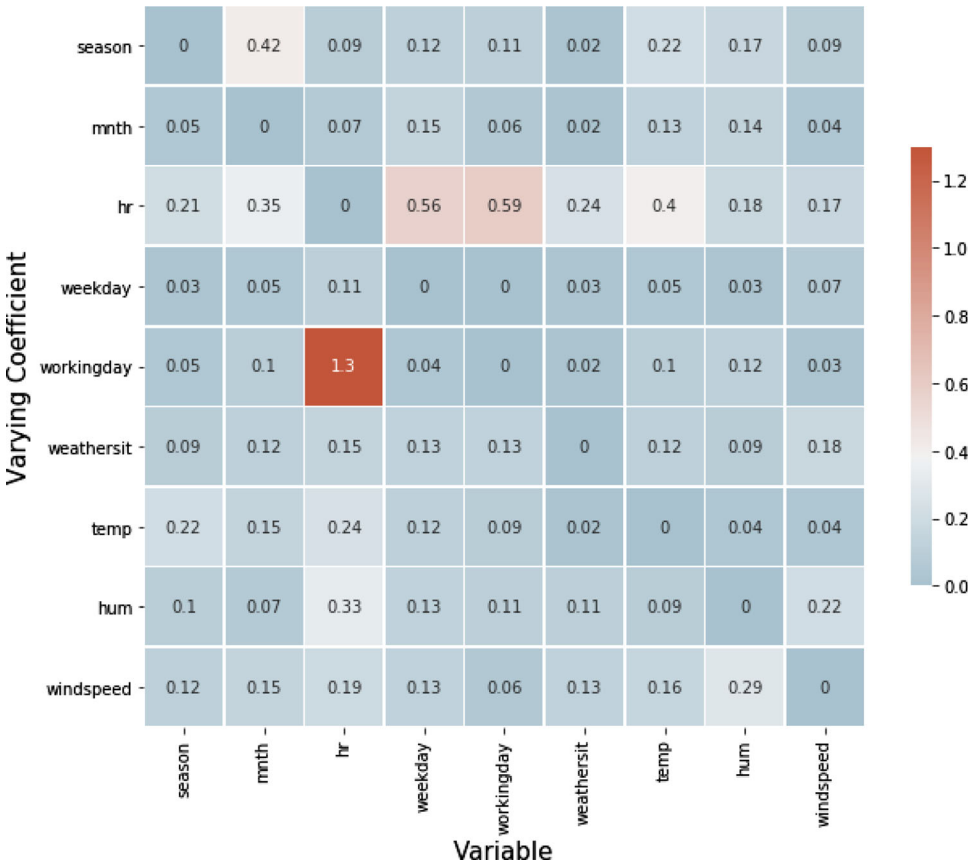


Fig. 14 Heatmap for interaction measures

reduce the number of local linear equations-merging and flattening in their paper, where a variety of other diagnostic tools and plots for local linear model have also been provided.

4.3 Main and interaction effect detection

Even though the parallel coordinates plot provides a guideline about the variable importance in each local region, we still need a solid technique to detect nonlinear main effects and interaction effects. To achieve this purpose, we can treat single-hidden-layer NN as a varying coefficient model through linear equation extraction shown in Algorithm 2. As all the local linear equation coefficients are varying over local regions, and region definition depends on predictors, so the coefficients can be treated as a function of predictors in Eq. 11.

$$\begin{aligned} &\hat{f}_i = \alpha_{0i} + \alpha_{1i}x_{1i} + \cdots + \alpha_{mi}x_{mi} + \cdots + \alpha_{pi}x_{pi}, \\ &i = 1, \cdots, n, \\ &\text{where } \alpha_{mi} = g(x_{1i}, \cdots, x_{pi}), \end{aligned} \tag{11}$$

where p is the number of predictors and n is number of observations. $\hat{f}_i$ is predicted value for regression and predicted log-odds for classification. α_{mi} is the coefficient for m^{th} variable at i^{th} observation, and could also be a function of all predictors, varying by different observations. Our goal is to investigate what the functional forms of the estimated coefficients are. Therefore, we separate α_{mi} into two components representing main and interaction effects in Eq. 12:

$$\alpha_{mi} = g(x_{1i}, \cdots, x_{pi}) = \underbrace{g_{main}(x_{mi})}_{\text{main effect}} + \underbrace{g_{int}(x_{1i}, \cdots, x_{pi})}_{\text{interaction effect}}. \tag{12}$$

The first term in Eq. 12 is a function of x_{mi} , including the intercept of α_{mi}, and this term captures the main effect of x_{mi}. If α_{mi} has a significant intercept, then linear main effect can be identified; while a strong relationship with x_{mi} indicates a nonlinear main effect. The remaining second term is the function of other predictors and it may or may not contain x_{mi}. This term can be used to detect interactions between x_{mi} and other predictors. For an illustration, let us look at a simple example with all estimated coefficients constant except $\alpha_{1i} = \theta_0 + \theta_1 x_{1i} + \theta_2 x_{2i}$, then the varying coefficient model can be expressed as follows:

$$\begin{aligned} y_i = \alpha_{0i} + \theta_0 x_{1i} + \theta_1 x_{1i}^2 + \theta_2(x_{2i})x_{1i} + \alpha_2 x_{2i} + \\ \cdots + \alpha_p x_{pi} + \epsilon_i. \end{aligned} \tag{13}$$

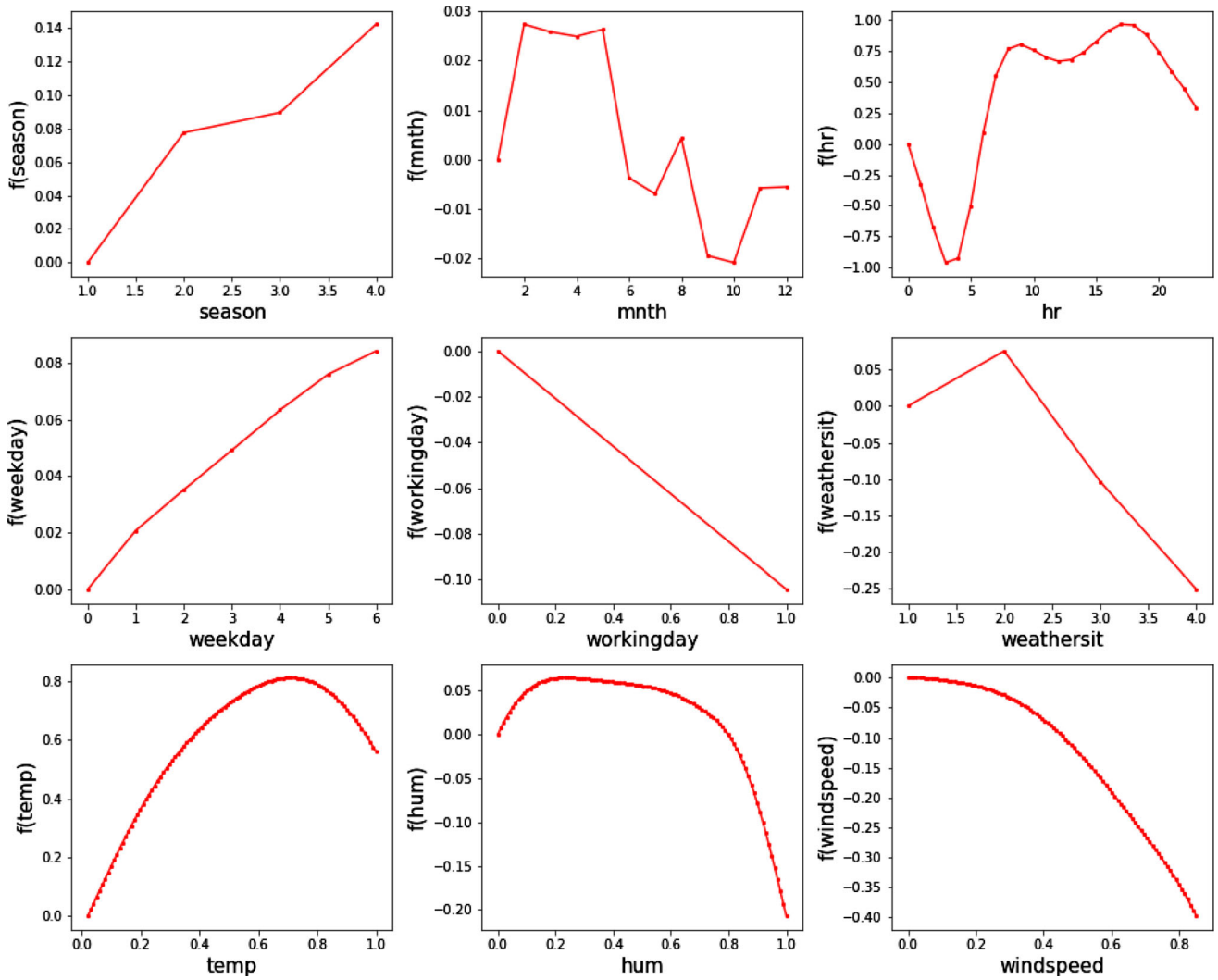


Fig. 15 ALE plots for predictors based on LIFE

where $g_{main}(x_{1i}) = \theta_0 + \theta_1 x_{1i}$ and $g_{int}(x_{2i}) = \theta_2 x_{2i}$. We can easily identify the interaction term between x_{1i} and x_{2i}, and x_{1i} shows a nonlinear main effect via its quadratic term. To detect main effects and interaction effects from α_{mi}, we propose the two-stage process below:

1. Check nonlinearity: Calculate conditional expectation $E(\hat{\alpha}_{mi}|x_{mi})$ by smoothing estimated coefficients of predictors against itself[1]. $\hat{\alpha}_{mi} \sim g_{main}(x_{mi}) \quad m = 1, \cdots, p.$
2. Check interactions: Remove main effect from α_{mi}, and calculate conditional expectation $E(\hat{\alpha}_{mi} - \hat{g}_{main}(x_{mi})|x_{ki})$ by smoothing estimated coefficients of predictors against each other variable. $\hat{\alpha}_{mi} - \hat{g}_{main}(x_{mi}) \sim g_k^m(x_{ki}) \quad k \neq m$

[1] The smoothing spline is used for illustration and other estimation methods can also be applied.

We choose to use a two-stage process instead of a one-stage process, as we can estimate its main and interaction effect more accurately in the correlated predictor case and split two effects effectively. Note that some special interaction effects may not be identified by one-stage process such as $y = \alpha_0 + \alpha_1 x_1$ as an example, where $\alpha_1 = x_1 x_2$. In this case, $g_2^1(x_{2i})$ is zero curve. Fortunately, most common interaction patterns can be identified by our two-stage process. As long as $g_k^m(x_{ki})$ has a significant pattern on x_{ki}, an interaction effect can be identified.

For the case of bike sharing data, we visualized all pairs of varying coefficients and variables (α_{mi} *vs* x_{ki}) with scatter plots in Fig. 13. Due to $\frac{\partial \hat{f}(\mathbf{x})}{\partial x_m} = \alpha_m$, this is also scattered partial derivative plot for $\hat{f}(\mathbf{x})$. On top of the scatter plot, we also draw $g_{main}(x_{mi})$ against x_{mi} in the diagonal plots show and $g_k^m(x_{ki})$ agaist x_{ki} in the (m, k) off-diagonal plots. To further quantify the magnitude of the

Table 9 Sampling Method Comparison

Data	Metric	NN projection	Random projection	Bootstrapping	LLA or Adam
MIM (Regression)	*RMSE*	**1.060**	1.095	1.138	1.110
		(0.017)	(0.023)	(0.034)	(0.065)
	R^2	**0.965**	0.962	0.960	0.961
		(0.001)	(0.002)	(0.003)	(0.005)
California Housing	*RMSE*	**0.500**	0.525	0.519	0.527
		(0.012)	(0.020)	(0.018)	(0.011)
	R^2	**0.812**	0.792	0.797	0.791
		(0.009)	(0.016)	(0.014)	(0.008)
MIM (Classification)	*AUC*	**0.940**	0.935	0.936	0.920
		(0.004)	(0.004)	(0.003)	(0.003)
	Logloss	**0.268**	0.277	0.275	0.421
		(0.012)	(0.009)	(0.007)	(0.075)
Gamma telescope	*AUC*	**0.938**	0.929	0.927	0.909
		(0.002)	(0.003)	(0.003)	(0.008)
	lLgloss	**0.288**	0.307	0.314	0.371
		(0.007)	(0.007)	(0.007)	(0.019)

The different sampling methods in the first step of the general framework of LIFE are tested on two simulated data and two real data. NN projection performs data partition by linear projection inside neurons of NNs and then samples data from the selected region, which is the main part of LIFE algorithm used in the paper. Random projection performs data partition by random linear projection, where weights and biases are randomly drawn from standard normal. The bootstrapping selects observations randomly from training set. The number in bold represents optimal result for this metric. LLA is used to optimize single-hidden-layer NN base learner in the regression case, which Adam is used for classification case

interaction effects, we calculated weighted standard deviation of $g_{main}(x_{mi})$ and $g_k^m(x_{ki})$ with population density as weight. The heatmap of the interaction measures for bike sharing data is provided in Fig. 14 where diagonals are masked by zero.

The nonlinear patterns of variables can be clearly spotted in the diagonal plots in Fig. 13. The most important variable hour (*hr*) displays the drastic fluctuation compared with others, indicating its nonlinear effect on response. As evidenced in both Figs. 13 and 14, the top three interaction pairs including *hr* vs *workingday*, *hr* vs *weekday*, and *hr* vs *temp* can be easily identified.

In addition to interaction detection, we can obtain and visualize the main effect of each predictor directly by aggregating the local linear coefficients. Due to $\frac{\partial \hat{f}(\mathbf{x})}{\partial x_m} = \alpha_m$ in the varying coefficient setting, we compute the exact main effect of x_m by constructing a relationship between $f(x_j)$ and x_j based on formula $\int_0^x E(\frac{\partial \hat{f}(\mathbf{x})}{\partial x_m}|x_m)dx_m$ from Accumulated Local Effects (ALE) plot, where the variable is transformed back to original scale as seen in Fig. 15. This ALE formulation can be simplified as $\int_0^x E(\alpha_m|x_m)dx_m = \int_0^x g_{main}(x_m)dx_m$ and its numerical implementation of ALE is achieved by the Midpoint Rule.

The main effect for hr has two peaks and one trough, which is similar to partial dependence plot from other machine learning algorithms, while the main effect of *temp* and *hum* shows a quadratic relationship. More specifically, the peak of bike rentals happen around 7 am and 5–6 pm, while very few people will rent bikes around 3–4 am. People usually prefer to rent bikes in a nice day with moderate temperature and humidity. Both of them are pretty consistent with common sense.

5 Discussion

5.1 Different sampling schemes

The LIFE algorithm can be considered a general framework with three steps, as discussed in the methodology section. LIFE is very flexible and allows users to try different combinations of three steps. The first step presents several data sampling options. In the paper, we use linear projection inside NN neurons to split data and select data points from active region for base learner training, as shown in Fig. 2. Those linear projections are obtained with trained NNs in a supervised setting. Given a fixed hyper-parameter setup for LIFE, we have implemented our method with different sampling choices including NN projection, Random projection, Bootstrapping. In Table 9, we can easily see that all the ensemble methods outperform a single single-hidden-layer NN model optimized by LLA

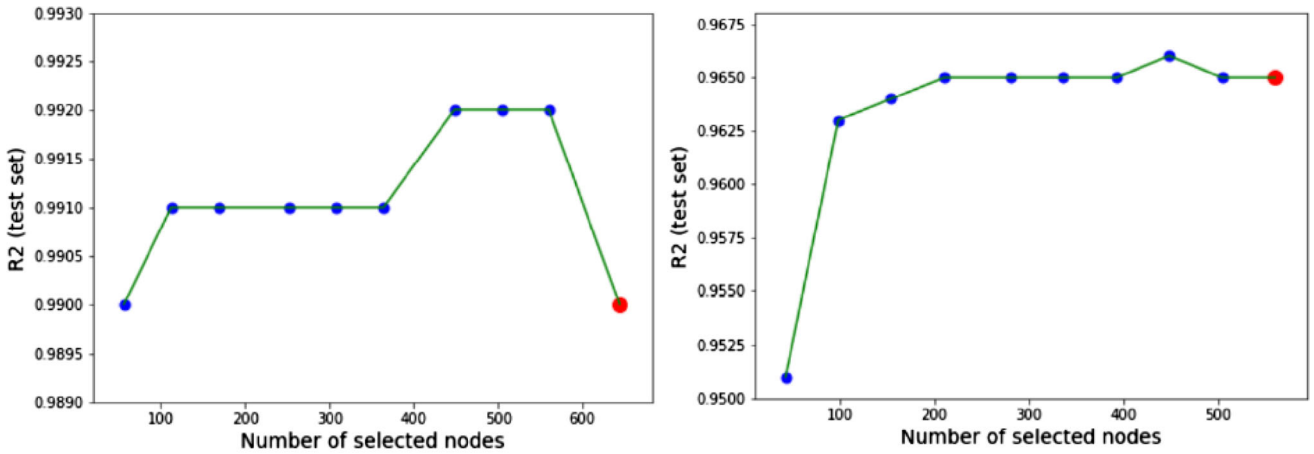


Fig. 16 Relationship between R^2 and number of neurons. The red point indicates one using linear regression as a final step without base learner selection, while blue points indicates model aggregation by base learner selection with different threshold "colour figure online"

or Adam. Most importantly, sampling by NN linear projection is better than other sampling methods that create subsets in a random way.

5.2 Base learner selection

For model aggregation and pruning, we can prune neurons to have a single-layer NN with fewer neurons. We used elastic net for pruning due to its simplicity, but pruning methods besides elastic net can also be considered. Based on properties of LIFE, we also developed an alternative pruning method called base learner selection to reduce the number of nodes in the final step. It is assumed that LIFE works well because the correlations of prediction errors are not strong among different base learners. Therefore, we can remove base learner one by one, according to the correlation between its prediction errors and prediction errors from other base learners. In this way, we can still maintain diversity and solve overfitting issue by keeping fewer necessary base learners without sacrificing predictive performance a lot. This method is thoroughly described by Algorithm 3.

Algorithm 3 Base learner selection

Input: B_j: trained base learner $j = 1, \cdots, M$; τ: threshold
Output: B_j: selected base learner $j \in \{1, \cdots, M\}$
Training set $\{x_i, y_i\}_{i=1,\cdots,N}$
for $j = 1, \cdots, M$ **do**
  i. $\hat{y}_i^{(j)} = B_j(x_i)$
  ii. $r_i^{(j)} = y_i - \hat{y}_i^{(j)}$
end for
for $j = 1, \cdots, M$ **do**
  i. Train a linear model $r_i^{(j)} \sim c_0 + c_1 r_i^{(1)} + \cdots + c_{j-1} r_i^{(j-1)} + c_{j+1} r_i^{(j+1)} + \cdots + c_M r_i^{(M)}$
  ii. Collect R_j^2 from this linear model
end for
Remove B_j based on value of R_j^2 in descending order until τ is achieved (the one with highest R^2 will be removed first).

In Algorithm 3, the threshold τ is the percentage of base learners you want to retrieved from a pool of candidates. When there is a large number of neurons in LIFE setting,

the elastic net is usually computational expensive and the base learner selection is a good alternative by parallel computation. Moreover, we can combine these two pruning methods to achieve a simpler NN model from wide NN faster. We also illustrate it using two simulated model (GAM and MIM). The plots (a) and (b) in Fig. 16 show the relationship between R^2 and number of hidden neurons for the feature extraction. Setting different thresholds in the base learner selection Algorithm 3, we can construct single-hidden-layer NN with different number of neurons. In general, base learner selection algorithms can effectively reduce number of neurons to produce a simpler model without sacrificing predictive performance or obtain even better results.

6 Conclusion

In this paper, we have proposed a novel algorithm that fits single-hidden-layer NN to achieve three goals: ensuring competitive predictive performance, boosting computational efficiency, and preserving the interpretability of the model. Unlike traditional NN training methods, we train it in an iterative way through multiple NNs layer-by-layer training and then effectively combine them via neural nodes flattening. We have evaluated the performance of our approach using simulated and empirical data in terms of predictive accuracy and computational efficiency and found that it consistently outperforms single-hidden-layer NN trained directly by LLA or Adam optimizer and achieves competitive results as those of Xgboost.

This superior performance lies in three reasons: First, as an ensemble method, the LIFE algorithm performs data sampling through linear projection inside neural nodes, which creates diversity among the models and contributes to bias and variance reduction of prediction from combined models. Second, the LIFE algorithm takes advantage of single-hidden-layer NN structure to combine multiple narrow single-hidden-layer NNs into a wide one via neural nodes flattening Third, LIFE algorithm benefits from leveraging parallel computing to train multiple NNs on subsets of data simultaneously. Moreover, the base learner selection method is introduced in the paper to help us prune redundant neural nodes and produce a more parsimonious model after several iterations of the LIFE algorithm. We have also proposed a new method for main and interaction detection from the perspective of interpretation.

Appendix

Loss decomposition in the log-odds space

The cross-entropy loss for one observation in the log-odds space can be written as:

$$l(y_i,f_i) = y_i log(1+e^{-f_i}) + (1-y_i) log(1+e^{f_i}) \tag{14}$$

Therefore, the average cross-entropy loss is expressed as:

$$\begin{aligned}
&\sum_{i=1}^{N}[y_i log(1+e^{-f_i}) + (1-y_i) log(1+e^{f_i})] \\
&= \underbrace{\sum_{i=1}^{N}\sum_{j=1}^{M}\beta_j[y_i log(1+e^{-f_i^{(j)}}) + (1-y_i) log(1+e^{f_i^{(j)}})]}_{\text{accuracy}} \\
&\quad \underbrace{- \sum_{i=1}^{N}\sum_{j=1}^{M}\beta_j\{\frac{e^{f_i^{(j)\star}} + e^{-f_i^{(j)\star}} + 2}{[(1+e^{-f_i^{(j)\star}})(1+e^{f_i^{(j)\star}})]^2}\}(f_i - f_i^{(j)})^2}_{\text{diversity}}
\end{aligned} \tag{15}$$

Local linear approximation (LLA) algorithm

The algorithm is extracted from Zeng, et al (2020) [26]. Note that $h^{(0)} = x$. Then, one-hidden-layer FNN with J_1 nodes can be expressed as follows

$$h^{(1)} = m\sigma(Wh^{(0)} + b) = (\sigma(w_1^T x + b_1), \cdots, \sigma(w_{J_1}^T x + b_{J_1}))^T,$$

where we drop the subscripts of W and b for ease of presentation. We set $\sigma(z) = \max\{z, 0\}$, the ReLU activation function. To estimate the weights and biases, we minimize a nonlinear LS function

$$l_1(\theta) = \sum_{i=1}^{n}\left\{y_i - \beta_0 - \sum_{j=1}^{J_1}\beta_j\sigma(w_j^T x_i + b_j)\right\}^2$$

Given $w_j^{(c)}$ and $b_j^{(c)}$ in the current step, we propose to approximate $\sigma(x^T w_j + b_j)$ by a linear function based on the first-order Taylor expansion of $\sigma(z)$:

$$\begin{aligned}
\sigma(x^T w_j + b_j) &\approx \sigma(x^T w_j^{(c)} + b_j^{(c)}) \\
&\quad + \{(x^T w_j + b_j) - (x^T w_j^{(c)} + b_j^{(c)})\} I(x^T w_j^{(c)} + b_j^{(c)} > 0)
\end{aligned}$$

for $j = 1, \cdots, J_1$. Thus,

$$\beta_j\sigma(x^Tw_j+b_j)\approx\beta_j\sigma(x^Tw_j^{(c)}+b_j^{(c)})+\gamma_jI(x^Tw_j^{(c)}+b_j^{(c)}>0)+m\eta_j^Txi(x^Tw_j^{(c)}+b_j^{(c)}>0)$$

where $\gamma_j=\beta_j(b_j-b_j^{(c)})$ and $m\eta_j=\beta_j(w_j-w_j^{(c)})$. Define $z_{1ij}=\sigma(x_i^Tw_j^{(c)}+b_j^{(c)})$, $z_{2ij}=I(x_i^Tw_j^{(c)}+b_j^{(c)}>0)$, and $z_{3ij}=x_iI(x_i^Tw_j^{(c)}+b_j^{(c)}>0)\}$, $j=1\cdots,J_1$. Further define $z_{1i}=[z_{1i1},\cdots,z_{1iJ_1}]$, $z_{2i}=[z_{2i1},\cdots,z_{2iJ_1}]$, $z_{3i}=[z_{3i1}^T,\cdots,z_{3iJ_1}^T]$, and $z_i=[z_{1i},z_{2i},z_{3i}]^T$, which is a $J_1(p+2)$-dimensional vector. the objective function is approximated by

$$\sum_{i=1}^{n}\{y_i-\beta_0-\sum_{j=1}^{J_1}\{\beta_jz_{1ij}+\gamma_jz_{2ij}+m\eta_j^Tz_{3ij}\}\}^2,$$

which is the LS function of linear regression with the response y_i and predictors z_i. Denote the resulting LS estimate of β_j, γ_j and $m\eta_j$ by $\hat{\beta}_j$, $\hat{\gamma}_j$ and $\hat{m\eta}_j$, respectively. By the definition of γ_j and $m\eta_j$, we can update b_j and w_j as shown in the step 2 of algorithm If $|\hat{\beta}_j|$ is very close to zero, one may simply set $b_j^{(c+1)}=b_j^{(c)}$ and $w_j^{(c+1)}=w_j^{(c)}$. Thus, we may estimate W and b by iteratively and regressing y_i on the updated z_i. The procedure can be summarized as the following algorithm.

1. Set initial value for $W^{(0)}=[w_1^{(0)},\cdots,w_{J_1}^{(0)}]^T$ and $b_j^{(0)}$, and let $c=0$.
2. Calculate z_i defined in the text based on $w_j^{(c)}$ and $b_j^{(c)}$, obtain the least squares estimate (LSE) $\hat{\beta}_j$s, $\hat{\gamma}_j$s and $\hat{\eta}_j$s by running a linear regression y_i on covariate z_i, and update the biases and weights by
   $b_j^{(c+1)}=b_j^{(c)}+\hat{\gamma}_j/\hat{\beta}_j$, and $w_j^{(c+1)}=w_j^{(c)}+\hat{\eta}_j/\hat{\beta}_j$.
   if $|\hat{\beta}_j|\geq\varepsilon$, where ε is a constant for numerical stability and is set to be 10^{-3} in our numerical experiment, and keep the corresponding biases and weights unchanged if $|\hat{\beta}_j|<\varepsilon$.
3. Set $c=1,2,\cdots$, and repeat Step 2 until the criterion of algorithm convergence meets.

Functional forms for classification case in simulation study

In classification case, we use three function forms including generalized additive model (GAM), additive index model (AIM), and multiple index model (MIM) expressed as follows, where $\epsilon_i\sim N(0,1), i=1,\ldots,N$.

$$GAM: f(\boldsymbol{x}_i)=1.5x_{1i}+4\sqrt{|-2.5x_{2i}|}+2|x_{3i}|+4\exp(-\frac{3}{14}x_{4i})+4\log(1.5|x_{5i}|)-4\max(1,x_{6i})+\epsilon_i. \quad (16)$$

$$AIM: f(\boldsymbol{x}_i)=\log(|3x_{1i}-2.5x_{2i}+2x_{3i}-1.5x_{4i}|)+\exp\{(-1.5x_{4i}+1.5x_{5i}-x_{6i})/11\}+\epsilon_i. \quad (17)$$

$$MIM: f(\boldsymbol{x}_i)=\exp(0.03x_{1i}-0.025x_{2i})x_{3i}-\frac{3x_{4i}}{1+1.5|x_{5i}|}+2\max(1,x_{6i})+\epsilon_i. \quad (18)$$

Real datasets

Regression

Case 1: Abalone. Original data come from a Marine Resources Division Marine Research Laboratories in Australia. The goal is to predict the age of abalone from physical measurements, which is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope. From the original data, examples with missing values were removed, and the ranges of the continuous values have been scaled for use. Therefore, there are 4177 observations left and nine variables including an integer response variable (number of rings), one categorical predictor (sex), and seven continuous predictors (length, diameter, height, whole weight, shucked weight, viscera weight, shell weight).

Case 2: Airfoil. It is NASA dataset, obtained from a series of aerodynamic and acoustic tests of two- and three-dimensional airfoil blade sections conducted in an anechoic wind tunnel. It comprises different size NACA 0012 airfoils at various wind tunnel speeds and angles of attack. There are 1503 observations and six variables, where the response variable is scaled sound pressure level in decibels and there are five independent variables containing frequency in Hertzs, angle of attack in degrees, chord length in meters, free-stream velocity, in meters per second, suction side displacement thickness in meters.

Case 3: Aquatic toxicity. This dataset was used to develop quantitative regression QSAR models to predict acute aquatic toxicity toward the fish Pimephales promelas (fathead minnow) on a set of 908 chemicals. To predict acute aquatic toxicity toward Daphnia Magna as a response variable called LC50. Without missing values, it contains values for 8 predictors (molecular descriptors) of 546

chemicals (TPSA, SAacc, H-050, MLOGP, RDCHI, GATS1p, nN, C-040).

Case 4: Bike sharing. This dataset contains the hourly and daily count of rental bikes between years 2011 and 2012 in Capital bike share system with the corresponding weather and seasonal information. Specifically, the bike sharing dataset contains 17379 data points of 16 predictors. Out of the 16 independent variables, we removed two non-meaningful information as well as two response-related information. This leaves us with 12 variables including hour, temperature, feeling temperature, humidity, wind speed, season, workingday weekday and weather situation, year and month.

Case 5: California housing. This dataset was derived from the 1990 US census, using one row per census block group and the target variable is the median house value for California districts. A block group is the smallest geographical unit for which the US Census Bureau publishes sample data (a block group typically has a population of 600 to 3000 people). It consists of 20640 observations and the independent variables include longitude, latitude, housing median age, medium income, population, total rooms, total bedrooms and households.

Case 6: CASP. This is a dataset of Physicochemical Properties of Protein Tertiary Structure, which is taken from CASP 5-9. The goal is to predict RMSD-size of residue given other physical attributes. There are 45730 observations and 9 predictors including total surface area, nonpolar exposed area, fractional area of exposed nonpolar residue, fractional area of exposed nonpolar part of residue, molecular mass weighted exposed area, average deviation from standard exposed area of residue, Euclidian distance, secondary structure penalty, special Distribution constraints (N,K Value).

Case 7: Electrical grid. The local stability analysis of the 4-node star system (electricity producer is in the center) implementing Decentral Smart Grid Control concept. There are 10000 observations and 11 predictors including $tau[x], x = 1, 2, 3, 4$ which are reaction time of participant, $p[x], x = 2, 3, 4$ which are nominal power consumed (negative) divided by produced(positive)(real), and $g[x], x = 1, 2, 3, 4$ that are coefficient (gamma) proportional to price elasticity. The continuous response variable is the maximal real part of the characteristic equation root.

Classification

Case 1: Bank marketing. The data provide information regarding direct marketing campaigns of a Portuguese banking institution. The classification goal is to predict if the clients, who were contacted based on at least two phone calls in general, would like to subscribe a term deposit or not. The data contain 45211 examples and 16 variables including 6 numerical variables (age, balance, day, campaign, pdays, previous) and nine categorical variables (job, marital, education, default, housing, loan, contact, month, poutcome). Notice that input variable 'duration' is not included in the analyses based on the suggestions from the provider of this dataset since it highly affects the output target and thus the variable should be discarded due to the intention of building a realistic predictive model.

Case 2: Breast cancer wisconsin. Data come from original Wisconsin Breast Cancer Database, with purpose of detecting if the tissue is benign or malignant. The original dataset contains 699 instances and 9 attributes. After deleting rows with missing values, finally 683 instances are used in our analyses. All predictive variables are numerical with a scale of 1 to 10, including clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses.

Case 3: Higgs boson. This dataset was built from official ATLAS full-detector simulation in 2014 that mixes 'Higgs to tautau' events with different backgrounds, where the events in which Higgs bosons were produced are comprised in the signal sample, while other known processes mimicking the signal are considered as background noise. The dataset contains 818238 events and 63 variables, among which a few are highly correlated. The objective is to detect signal from background based on characteristics of events such as mass (estimated, transverse or invariant), transverse momentum, centrality of the pseudo rapidity of the lepton, azimuth angle, etc.

Case 4: Home lending. This is an internal First Lien modeling data from Wells Fargo, which was constructed from the entire panel dataset of loans provided from the Consumer Lending Group's (CLG) data modeling team. The goal is to predict the probability of troubled loans based on fico score, unemployment rate, delinquency status, loan to value ratio (LTV), total employment, etc. The toy dataset used in this paper contains 1 million observations and 43 variables, which was selected from 210 million observations obtained by stacking. Training and testing data are split based on 'account id' to prevent overfitting.

Case 5: MAGIC gamma telescope. The dataset comes from Major Atmospheric Gamma Imaging Cherenkov Telescope project (MAGIC), which is Monte-Carlo generated to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. The objective is to discriminate the images caused by signals (primary

gammas) from the images caused by background (hadronic showers initiated by cosmic rays) based on some characteristics such as the attributes of ellipse (fLength, fWidth, fM3Long, fM3Trans, fAlpha, fDist) and attributes on pixels (fSize, fConc, fConc1, fAsym). The dataset contains 19020 observations and 10 continuous predictive variables.

Case 6: Mushroom. This dataset includes descriptions in terms of physical characteristics of samples related with 23 species of gilled mushrooms. Each species is identified as poisonous or edible. The purpose is to predict the probability of a species being poisonous given the attributes of the species such as cap (shape, surface, color), odor, gill (attachment, spacing, size, color), stalk (surface and color above or below ring), veil (type, color), ring (number, type), spore print color, population and habitat, all of which are categorical variables. The dataset contains 8124 instances and 21 variables.

Case 7: Ringnorm. It is a simulated dataset as an implementation of Leo Breimans ringnorm example with 7400 rows and 21 columns. Two classes contained in the response variable, within each class explanatory variables, are drawn from a multivariate Normal distribution.

Case 8: MNIST data. Data are originally obtained from the MNIST database of handwritten digits collected from hundreds of Census Bureau employees and high-school students. The response variable contains signal digit from 0 to 9, which are further centered into 28×28 gray-scaled images. Then, the images are translated into regular dataset with dimension 60,000 by 784, indicating that there are 60,000 images and 28×28 positions of the field. Digit '1' is considered as signal and all the other nine digits are considered as noise and are re-encoded as '0.' Training, validation, and testing data are randomly selected from these 60,000 observations for 10 times with size 8000, 2000, 2000, respectively. Images in the training dataset are then further rotated by 20 degrees.

Data availability statement The datasets, Abalone, Airfoil, Aquatic Toxicity, Bike Sharing, CASP, Electrical Grid, Bank Marketing, Breast Cancer Wisconsin, MAGIC Gamma Telescope and Mushroom, analyzed during the current study, are available in the UCI Machine Learning repository, https://archive.ics.uci.edu/ml/index.php. The datasets, California Housing, Higgs Boson, Ringnorm and MNIST Data, analyzed during the current study, are available in the Kaggle repository, https://www.kaggle.com/. The Home Lending dataset analyzed during the current study is not publicly available because it is Wells Fargo's internal data but is available from the corresponding author on reasonable request. Detailed logics on generating simulated datasets have been illustrated in Section 3.1 and Appendix in this manuscript, so that users can generate the simulated datasets by themselves.

Declaration

Conflict of interest The authors declare no conflict of interest.

References

1. Chen T, He T, Benesty M, Khotilovich V, Tang Y (2015) Xgboost: extreme gradient boosting. R package version 0.4-2, 1–4
2. Breiman L (2001) Random forests. Mach Learn 45(1):5–32
3. Kucherenko S et al (2009) Derivative based global sensitivity measures and their link with global sensitivity indices. Math Comput Simul 79(10):3009–3017
4. Kucherenko S et al (2010) A new derivative based importance criterion for groups of variables and its link with the global sensitivity indices. Comput Phys Commun 181(7):1212–1217
5. Sundararajan M, Taly A, Yan Q (2017) Axiomatic attribution for deep networks. arXiv preprint arXiv:1703.01365
6. Ancona M, Ceolini E, Öztireli C, Gross M (2017) Towards better understanding of gradient-based attribution methods for deep neural networks. arXiv preprint arXiv:1711.06104
7. Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell 1(5):206–215
8. Vaughan J, Sudjianto A, Brahimi E, Chen J, Nair VN (2018) Explainable neural networks based on additive index models. arXiv preprint arXiv:1806.01933
9. Chen J, Vaughan J, Nair V, Sudjianto A (2020) Adaptive explainable neural networks (axnns). Available at SSRN 3569318
10. Yang Z, Zhang A, Sudjianto A (2020) Enhancing explainability of neural networks through architecture constraints. IEEE Trans Neural Netw Learn Syst 32(6):2610–2621
11. Andrienko G, Andrienko N (2001) Constructing parallel coordinates plot for problem solving. In: 1st International Symposium on Smart Graphics, pp. 9–14
12. Heath D, Kasif S, Salzberg S (1993) Induction of oblique decision trees. In: IJCAI, vol. 1993, pp. 1002–1007
13. Wolpert DH (1992) Stacked generalization. Neural Netw 5(2):241–259
14. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, pp. 785–794
15. Qiu J, Jammalamadaka SR, Ning N (2018) Multivariate bayesian structural time series model. J Mach Learn Res 19:1–33
16. Qiu J, Jammalamadaka SR, Ning N (2020) Multivariate time series analysis from a bayesian machine learning perspective. Ann Math Artif Intell 88(10):1061–1082

17. Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach Learn 51(2):181–207
18. Brown G, Wyatt JL, Ti o P (2005) Managing diversity in regression ensembles. Journal of machine learning research 6(Sep), 1621–1650
19. Aksela M, Laaksonen J (2006) Using diversity of errors for selecting members of a committee classifier. Pattern Recogn 39(4):608–623
20. Gacquer D, Delcroix V, Delmotte F, Piechowiak S (2009) On the effectiveness of diversity when training multiple classifier systems. In: European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, pp. 493–504. Springer
21. Butler HK IV, Friend MA, Bauer KW Jr, Bihl TJ (2018) The effectiveness of using diversity to select multiple classifier systems with varying classification thresholds. J Algorithm Comput Technol 12(3):187–199
22. Krogh A, Vedelsby J (1994) Neural network ensembles, cross validation, and active learning. Adv Neural Inf Process Syst 7:231–238
23. Ueda N, Nakano R (1996) Generalization error of ensemble estimators. In: Proceedings of International Conference on Neural Networks (ICNN'96), vol. 1, pp. 90–95. IEEE
24. Brown G, Wyatt J, Harris R, Yao X (2005) Diversity creation methods: a survey and categorisation. Inf Fusion 6(1):5–20
25. Hansen JV (2000) Combining predictors: Meta machine learning methods and bias/variance & ambiguity decompositions. PhD thesis, Aarhus University, Computer Science Department
26. Zeng M, Liao Y, Li R, Sudjianto A (2020) Local linear approximation algorithm for neural network. Manuscript
27. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778
28. Lecun Y, Cortes C, Burges C (1999) The MNIST Dataset of Handwritten Digits(Images)
29. Sudjianto A, Knauth W, Singh R, Yang Z, Zhang A (2020) Unwrapping the black box of deep relu networks: Interpretability, diagnostics, and simplification. arXiv preprint arXiv:2011.04041