Accumulated Local Effects (ALE) and Package `ALEPlot`

Jingyu Zhu, Daniel W. Apley

November 12, 2017

1 Motivation: Partial Dependence Plots, Marginal Plots, and the Need for ALE Plots

Due to their flexibility, black box supervised learning models (for example, complex trees, neural networks, and support vector machines) have been widely used to capture nonlinear relationships in predictive modeling. However, they often lack interpretability in the sense that it is difficult to study the effect of each predictor on the response and the interactions among different predictors. Understanding these effects and interactions is obviously crucial if the predictive models serve an explanatory purpose. Even if the models are purely predictive, effect visualization is an important model diagnostic tool for the users.

Suppose that we have fit a black box supervised learning model to approximate $E[Y|X = x] \approx f(x)$. Here, Y is a scalar response variable, $X = (X_1, ..., X_d)$ is a d-dimensional vector of predictors, and $f(\cdot)$ is the fitted model that predicts Y (or the probability that Y falls into a particular class in the classification setting) as a function of X. The training data to which we fit the model is $\{y_i, x_i = (x_{i,1}, ..., x_{i,d}) : i = 1, ..., n\}$. For simplicity of notation, we omit the $\hat{}$ symbol and denote our fitted model by f instead of $\hat{f}$. We use upper case X and Y to denote random variables, and lower case to denote specific values of the random variables.

The objective here is to visualize the 'main effect' dependence of $f(x_1, ..., x_d)$ on each predictor $x_1, ..., x_d$ as well as the lower-order 'interaction effects' among different predictors. In the introduction of this vignette, we illustrate concepts for the simple case of $d = 2$. General d is conceptually similar and is considered throughout the rest of the vignette. Details can be found in Apley (2016). One popular visualization approach is the partial dependence (PD) plot proposed in Friedman (2001). To visualize the effect of one predictor, say x_1, on the response $f(\cdot)$, a PD plot plots the function

$$f_{1,PD}(x_1) = E[f(x_1, X_2)] \tag{1}$$

versus x_1. An estimate of (1) is

$$\hat{f}_{1,PD}(x_1) = \frac{1}{n}\sum_{i=1}^{n} f(x_1, x_{i,2}) \tag{2}$$

.

Figure 1(a) illustrates how $f_{1,PD}(x_1)$ is computed at a specific value $x_1 = 0.3$ for a toy example with $n = 200$ observations of (X_1, X_2) following a uniform distribution along the line segment $x_2 = x_1$ but with independent $N(0, 0.05^2)$ variables added to both predictors. The salient point in Figure 1(a), which illustrates the problem with PD plots, is that the expectation in (1) is the weighted average of $f(x_1, X_2)$ as X_2 varies over its marginal distribution. This weighted average

is equivalent to an integral over the entire vertical line segment in Figure 1(a) and requires rather severe extrapolation beyond the envelope of the training data. If one were to fit a simple parametric model (e.g., $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2^2$) of the correct form, then this extrapolation might be reliable. However, by nature of its flexibility, a nonparametric supervised learning model like a regression tree cannot be expected to extrapolate reliably. Hence, a PD plot is not an ideal tool for visualization when the predictors are correlated.

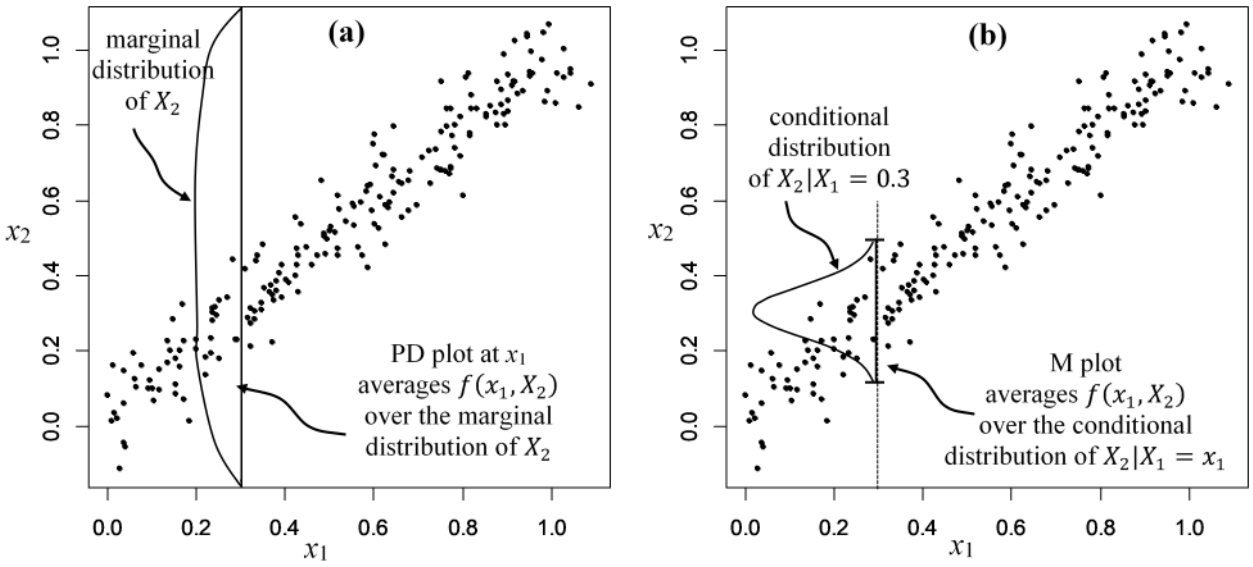

Figure 1: Illustration of the differences between the computation of (a) $f_{1,PD}(x_1)$ and (b) $f_{1,M}(x_1)$ at $x_1 = 0.3$.

The extrapolation in Figure 1(a) that is required to calculate $f_{1,PD}(x_1)$ occurs because the marginal density of X_2 is much less concentrated around the data than the conditional density of X_2 given $X_1 = x_1$, due to the strong dependence between X_2 and X_1. Marginal (M) plots are alternatives to PD plots that avoid such extrapolation by using the conditional density in place of the marginal density. As illustrated in Figure 1(b), an M plot of the effect of x_1 is a plot of the function

$$f_{1,M}(x_1) = E[f(X_1, X_2)|X_1 = x_1] \tag{3}$$

versus x_1. A crude estimate of $f_{1,M}(x_1)$ is

$$\hat{f}_{1,M}(x_1) = \frac{1}{n(x_1)} \sum_{i \in N(x_1)} f(x_1, x_{i,2}) \tag{4}$$

where $N(x_1) \subset \{1, 2, ..., n\}$ is the subset of row indices i for which $x_{i,1}$ falls into some small, appropriately selected neighborhood of x_1, and $n(x_1)$ is the number of observations in the neighborhood. Although more sophisticated kernel smoothing methods are typically used to estimate $f_{1,M}(x_1)$, we do not consider them here, because there is a more serious problem with using $f_{1,M}(x_1)$ to visualize the main effect of x_1 when X_1 and X_2 are dependent. Namely, using $f_{1,M}(x_1)$ is like regressing Y onto X_1 while ignoring (i.e., marginalizing over) the nuisance variable X_2. Consequently, if Y depends on X_1 and X_2, $f_{1,M}(x_1)$ will reflect both of their effects, a consequence of the omitted variable bias (OVB) phenomenon in regression. Viewed another way, if Y depends on X_1 but not X_2, correlation between X_1 and X_2 will result in $f_{1,M}(x_2)$ making it appear as though Y depends on X_2. In machine learning problems using large observational databases, predictor variables are often highly correlated. M Plots will then be severely biased by the OVB problem, rendering them virtually useless for visualizing the effects of the individual predictors.

2 Estimation and Interpretation of ALE Main Effects with the `ALEPlot` Package

The accumulated local effects (ALE) plots proposed in Apley (2016) constitute a visualization approach that avoids both the extrapolation problem in PD plots and the OVB problem in M plots. The ALE main effect of the predictor x_j, $j \in \{1, ..., d\}$ is defined as[1]

$$f_{j,ALE}(x_j) = \int_{z_{0,j}}^{x_j} E[\frac{\partial f(X_1, ..., X_d)}{\partial X_j})|X_j = z_j]dz_j - c_1 \tag{5}$$

Here, $z_{0,j}$ is an approximate lower bound of X_j. The constant c_1 is chosen such that $f_{j,ALE}(X_j)$ has a mean of zero with respect to the marginal distribution of X_j. An ALE plot of the main effect of x_j is a plot of an estimate of $f_{j,ALE}(x_j)$ versus x_j and it visualizes the main effect dependence of $f(\cdot)$ on x_j.

The estimate of the ALE main effect is obtained by replacing the integral in (5) with a summation and the derivative with a finite difference, i.e.,

$$\hat{f}_{j,ALE}(x) = \sum_{k=1}^{k_j(x)} \frac{1}{n_j(k)} \sum_{i:x_{i,j} \in N_j(k)} [f(z_{k,j}, x_{i,\setminus j}) - f(z_{k-1,j}, x_{i,\setminus j})] - \hat{c_1} \tag{6}$$

where the notation is as follows, and the constant $\hat{c_1}$ is chosen so that $\frac{1}{n}\sum_{i=1}^{n} \hat{f}_{j,ALE}(x_{i,j}) = 0$.

Let $x_{i,\setminus j} = (x_{i,l} : l = 1, ..., d; l \neq j)$, where the subscript $\setminus j$ indicates all variables but the jth, and let $\{N_j(k) = (z_{k-1,j}, z_{k,j}] : k = 1, 2, ..., K\}$ be a sufficiently fine partition of the sample range of $\{x_{i,j} : i = 1, 2, ..., n\}$ into K intervals.[2] The `ALEPlot` functions choose $z_{k,j}$ as the $\frac{k}{K}$ quantile of the empirical distribution of $\{x_{i,j} : i = 1, 2, ..., n\}$ with $z_{0,j}$ chosen just below the smallest observation, and $z_{K,j}$ chosen as the largest observation. For $k = 1, 2, ..., K$, let $n_j(k)$ denote the number of observations in $\{x_{i,j} : i = 1, ..., n\}$ that fall into the kth interval $N_j(k)$, so that $\sum_{k=1}^{K} n_j(k) = n$. For a particular value x for the predictor x_j, let $k_j(x)$ denote the index of the interval into which x falls, i.e., $x \in (z_{k_j(x)-1,j}, z_{k_j(x),j}]$. Figure 2 illustrates the computation of the ALE main effect estimator $\hat{f}_{j,ALE}(x_j)$ for the first predictor $j = 1$ for the case of $d = 2$ predictors.

[1] Technically, this definition of the theoretical ALE effect assumes differentiability of $f(\cdot)$, and the definition must be modified for nondifferentiable $f(\cdot)$, as discussed in Apley (2016). However, the estimator below and its implementation in the package remains valid, since the estimators use finite differences and summations, as opposed to differentiation and integration.

[2] K is an input argument in the `ALEPlot` functions, and we typically use K around 100, with larger values often give better results. Note that the algorithm may adjust (reduce) K internally if the predictors are discrete and have fewer than 50 distinct values. K is only used if the predictor is numeric. For factor predictors, the equivalent of K is the number of factor levels, which is automatically determined internally.

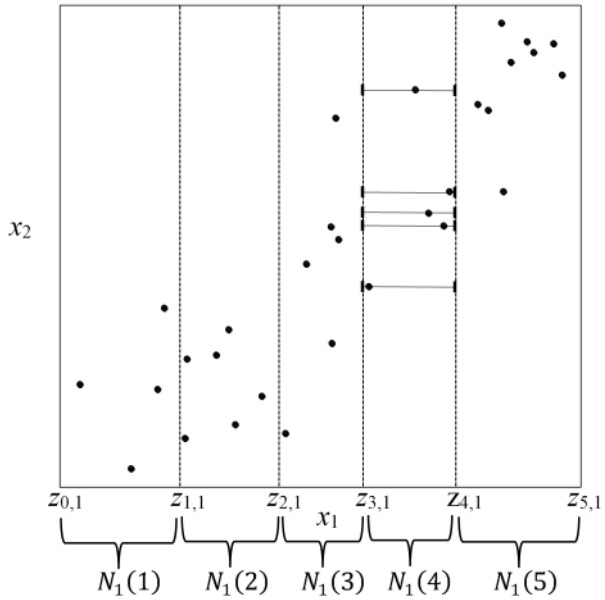

Figure 2: Illustration of the notation and concepts in computing the ALE main effect estimator $\hat{f}_{j,ALE}(x_j)$ for $j = 1$ with $d = 2$ predictors. The bullets are a scatterplot of $\{(x_{i,1}, x_{i,2}) : i = 1, 2, ..., n\}$ for $n = 30$ training observations. The range of $\{x_{i,1} : i = 1, 2, ..., n\}$ is partitioned into $K = 5$ intervals $\{N_1(k) = (z_{k-1,1}, z_{k,1}] : k = 1, 2, ..., 5\}$ (in practice, K should usually be chosen much larger than 5.) The numbers of training observations falling into the 5 intervals are $n_1(1) = 4, n_1(2) = 6, n_1(3) = 6, n_1(4) = 5$, and $n_1(5) = 9$. The horizontal lines segments shown in the $N_1(4)$ region are the segments across which the finite differences $f(z_{4,j}, x_{i,\setminus j}) - f(z_{3,j}, x_{i,\setminus j})$ are calculated and then averaged in the inner summand of (6) corresponding to $k = 4$ and $j = 1$.

The `ALEPlot` package is used to visualize the main effects of individual predictors and their second-order interaction effects (to be discussed in the next section) in black-box supervised learning models. It consists of two primary functions `ALEPlot` and `PDPlot`, which create ALE plots and PD plots respectively, given a fitted supervised learning model and the training data set to which it was fit. Note that the `ALEPlot` package depends on the `yaImpute` package, which is needed for ordering categorical predictors according to a nearest-neighbors type criterion. Hence before installing and loading the `ALEPlot` package, we need to install the `yaImpute` package. The following example illustrates the use of the `ALEPlot` package to visualize the main effects with a simulated example.

Example 1. Visualization of ALE Main Effects with Simulated Data

Suppose $X = \{X_1, X_2, X_3, X_4\}$ is a set of $d = 4$ predictor variables, where each $X_i : i \in \{1, 2, 3, 4\}$ follows a uniform distribution on the interval $[0, 1]$, and all 4 predictors are independent (an example with correlated predictors is given later). Suppose also that the response variable is

$$Y = 4x_1 + 3.87{x_2}^2 + 2.97\frac{exp(-5 + 10x_3)}{1 + exp(-5 + 10x_3)} + \epsilon,$$

where ϵ follows a $N(0, 1^2)$ distribution. The coefficients 4, 3.87, and 2.97 were chosen so that the three terms have approximately the same variance. We generated $n = 5000$ observations $\{y_i, x_i = (x_{i,1}, ..., x_{i,4}) : i = 1, ..., n\}$ from this model as the training data set, to which we fit a neural network model using the nnet package by Venables and Ripley (2002) with 8 nodes in the single hidden layer, a linear output activation function, and a decay parameter of 0.1.

These parameters were chosen as approximately optimal via multiple replicates of 10-fold cross-validation. We then calculated and plotted ALE main effect plots for the four predictors using $K = 100$. The following R code generates the data, fits the neural network, and computes and constructs the ALE plots:

```
## R code for Example 1
## Load relevant packages
library(ALEPlot)
library(nnet)

## Generate some data and fit a neural network supervised learning model
n = 5000
x1 <- runif(n, min = 0, max = 1)
x2 <- runif(n, min = 0, max = 1)
x3 <- runif(n, min = 0, max = 1)
x4 <- runif(n, min = 0, max = 1)
y = 4*x1 + 3.87*x2^2 + 2.97*exp(-5+10*x3)/(1+exp(-5+10*x3))+ rnorm(n, 0, 1)
DAT <- data.frame(y, x1, x2, x3, x4)
nnet.DAT <- nnet(y~., data = DAT, linout = T, skip = F, size = 8,
decay = 0.1, maxit = 1000, trace = F)

## Define the predictive function
yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata,
type = "raw"))

## Calculate and plot the ALE main effects of x1, x2, x3, and x4
par(mfrow = c(2,2), mar = c(4,4,2,2) + 0.1)
ALE.1 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = 1,
K = 100, NA.plot = TRUE)
ALE.2 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = 2,
K = 100, NA.plot = TRUE)
ALE.3 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = 3,
K = 100, NA.plot = TRUE)
ALE.4 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = 4,
K = 100, NA.plot = TRUE)

## Manually plot the ALE main effects on the same scale for easier
## comparison of the relative importance of the four predictor variables
plot(ALE.1$x.values, ALE.1$f.values, type="l", xlab="x1",
ylab="ALE_main_x1", xlim = c(0,1), ylim = c(-2,2), main = "(a)")
plot(ALE.2$x.values, ALE.2$f.values, type="l", xlab="x2",
ylab="ALE_main_x2", xlim = c(0,1), ylim = c(-2,2), main = "(b)")
plot(ALE.3$x.values, ALE.3$f.values, type="l", xlab="x3",
ylab="ALE_main_x3", xlim = c(0,1), ylim = c(-2,2), main = "(c)")
plot(ALE.4$x.values, ALE.4$f.values, type="l", xlab="x4",
ylab="ALE_main_x4", xlim = c(0,1), ylim = c(-2,2), main = "(d)")
```

Note that the first argument of the `ALEPlot` function is the data frame of predictor variables (excluding the response variables) to which the supervised learning model is fit. The second argument of `ALEPlot` is the supervised learning model object. The argument pred.fun is a user-supplied function that will be used to predict the response for the supervised learning model object. For most supervised learning model objects, pred.fun can simply call the predict function that was written as part of that modeling object package, assuming the package contains a predict function. The argument J indicates whether the ALE main effect (J = 1) or the ALE second-order effect (J = 2) estimate (to be discussed in the next section) will be plotted. K specifies the number of intervals into which the space of the predictor of interest is divided. `ALEPlot` has three output values: K, x.values, and f.values. In particular, f.values stores the ALEPlot estimates evaluated at the break points of the predictor space. We used these values in

the manual plotting commands. More details of the function arguments and output values can be found in the package help file.

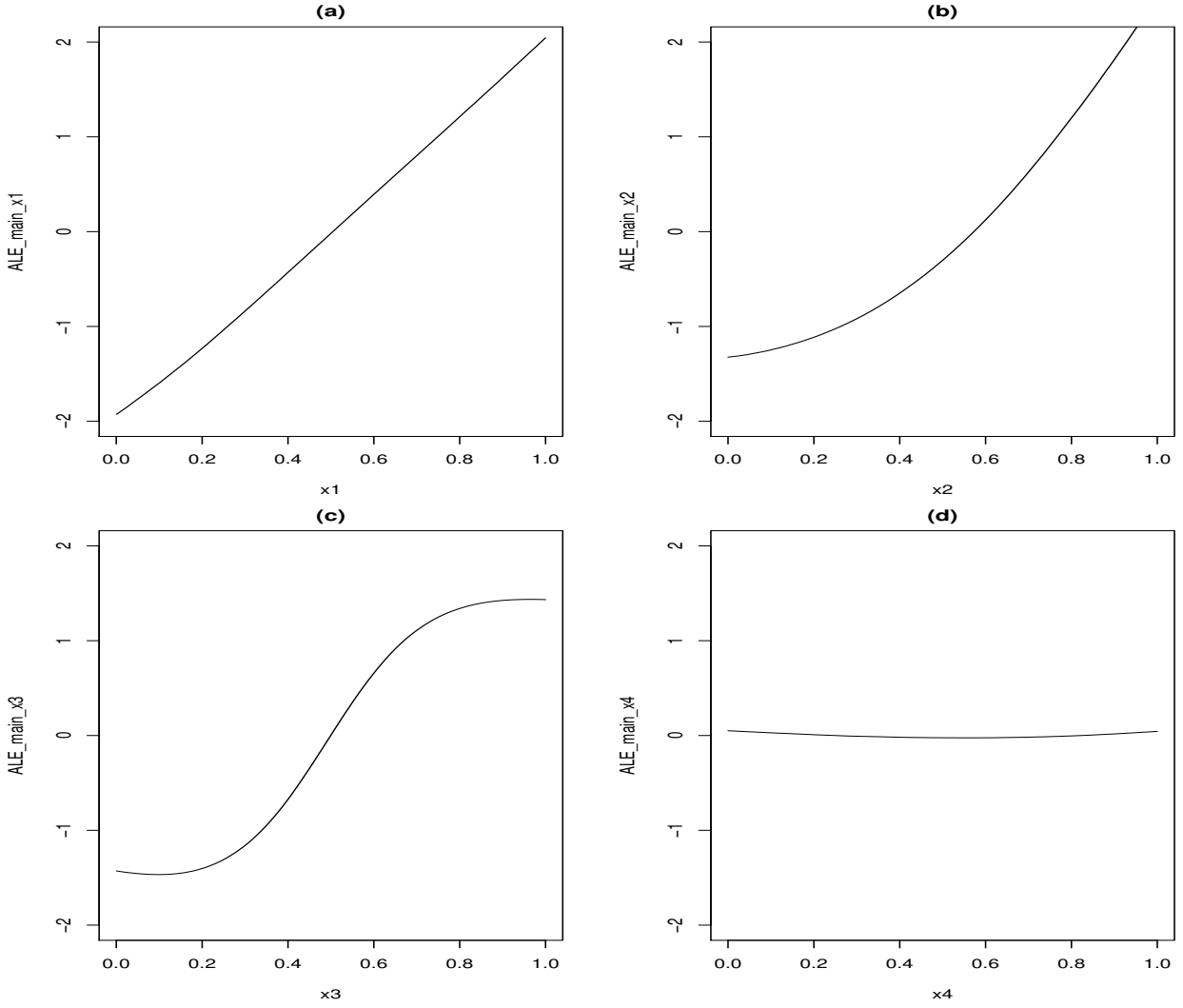

Figure 3: ALE main effect plots for the fitted neural network model in Example 1: (a) $\hat{f}_{1,ALE}(x_1)$ vs. x_1, (b) $\hat{f}_{2,ALE}(x_2)$ vs. x_2, (c) $\hat{f}_{3,ALE}(x_3)$ vs. x_3, and (d) $\hat{f}_{4,ALE}(x_4)$ vs. x_4. The four estimated ALE main effects accurately capture the correct linear, quadratic, and sigmoidal relationships, respectively, for x_1, x_2, and x_3. Moreover, $\hat{f}_{4,ALE}(x_4)$ in panel (d) correctly indicates that x_4 has no effect on Y.

Figure 3 shows the ALE main effect plots for x_1, x_2, x_3, and x_4 for Example 1. The estimates of $f_{1,ALE}(x_1)$, $f_{2,ALE}(x_2)$, and $f_{3,ALE}(x_3)$ clearly capture the correct linear, quadratic, and sigmoidal relationships quite well. Notice also that the estimate of $f_{4,ALE}(x_4)$ in panel (d) is almost zero, which agrees with the fact that Y did not depend at all on X_4 in the model for generating the data, so that there is no functional dependence of Y on X_4. In this manner, ALE plots can be used to visually assess variable importance (main effect plots for main effect importance).

3 Estimation and Interpretation of ALE Second-Order Effects with the `ALEPlot` Package

The ALE second-order effect of the predictors $\{x_j, x_l\}$, $\{j,l\} \subseteq \{1,...,d\}$ is defined as

$$f_{\{j,l\},ALE}(x_j,x_l) = \int_{z_{0,l}}^{x_l}\int_{z_{0,j}}^{x_j} E[\frac{\partial^2 f(X_1,...,X_d)}{\partial X_j \partial X_l}|X_j = z_j, X_l = z_l]dz_j dz_l - g_j(x_j) - g_l(x_l) - c_2 \quad (7)$$

Here, $z_{0,j}$ and $z_{0,l}$ are approximate lower bounds of X_j and X_l, respectively. The functions $g_j(x_j)$ and $g_l(x_l)$ (functions of the single variables x_j and x_l, respectively) and the constant c_2 are calculated so that $f_{\{j,l\},ALE}(x_j,x_l)$ is 'doubly centered' in the sense that $f_{\{j,l\},ALE}(X_j,X_l)$ has a mean of zero with respect to the marginal distribution of $\{X_j,X_l\}$, and the ALE main effects of x_j and x_l on $f_{\{j,l\},ALE}(X_j,X_l)$ are both zero. A contour plot of an estimate of $f_{\{j,l\},ALE}(x_j,x_l)$ vs. $\{x_j,x_l\}$ shows the interaction effect between the two predictors.

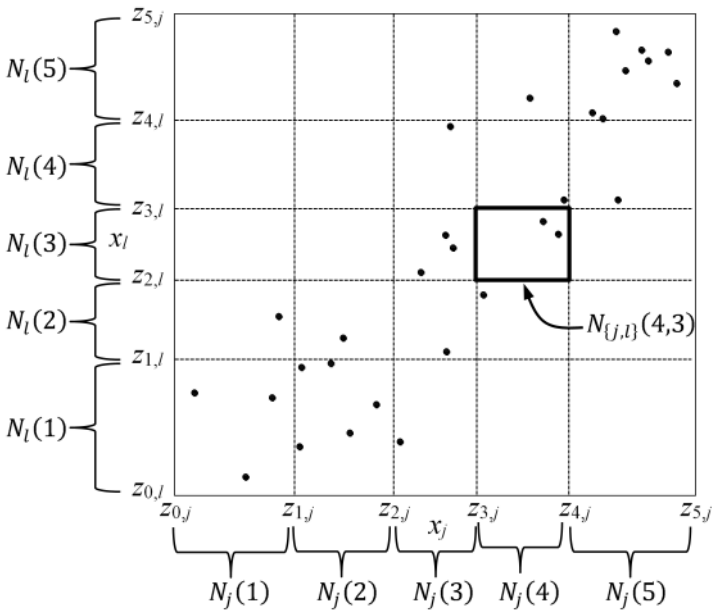

Figure 4: Illustration of the notation used in computing the ALE second-order effect estimator $\hat{f}_{\{j,l\},ALE}(x_j,x_l)$ for $K = 5$. The ranges of $\{x_{i,j} : i = 1,2,...,n\}$ and $\{x_{i,l} : i = 1,2,...,n\}$ are each partitioned into 5 intervals, and their Cartesian product forms the grid of rectangular cells $\{N_{\{j,l\}}(k,m) = N_j(k) \times N_l(m) : k = 1,2,...,5; m = 1,2,...,5\}$. The cell with bold borders is the region $N_{\{j,l\}}(4,3)$. The second-order finite differences in Eq.(8) for $(k,m) = (4,3)$ are calculated across the corners of this cell. In the inner summation of Eq. (8), these differences are then averaged over the $n_{\{j,l\}}(4,3) = 2$ observations in region $N_{\{j,l\}}(4,3)$.

An estimate of the ALE second-order effect of $\{X_j,X_l\}$ at any $(x_j,x_l) \in (z_{0,j},z_{K,j}] \times (z_{0,l},z_{K,l}]$ is

$$\hat{f}_{\{j,l\},ALE}(x_j,x_l) = \sum_{k=1}^{k_j(x_j)}\sum_{m=1}^{k_l(x_l)} \frac{1}{n_{\{j,l\}}(k,m)} \sum_{i:x_{i,\{j,l\}} \in N_{\{j,l\}}(k,m)} [[f(z_{k,j},z_{m,l},x_{i,\setminus\{j,l\}}) - f(z_{k-1,j},z_{m,l},x_{i,\setminus\{j,l\}})]$$
$$- [f(z_{k,j},z_{m-1,l},x_{i,\setminus\{j,l\}}) - f(z_{k-1,j},z_{m-1,l},x_{i,\setminus\{j,l\}})]] - \hat{g}_j(x_j) - \hat{g}_l(x_l) - \hat{c}_2 \quad (8)$$

, where the functions $\hat{g}_j(x_j)$, $\hat{g}_l(x_l)$ and the constant $\hat{c}_2$ are calculated to 'doubly center' the ALE second order effect estimate, and the notation is as follows. As illustrated in Figure 4, we

partition the $\{X_j, X_l\}$ space into a grid of K^2 rectangular cells $\{N_{\{j,l\}}(k,m) = N_j(k) \times N_l(m) : k = 1, 2, ..., K; m = 1, 2, ..., K\}$ obtained as the cross product of the intervals in the individual one-dimensional partitions defined in Section 2. Let $n_{\{j,l\}}(k,m)$ denote the number of training observations that fall into cell $N_{\{j,l\}}(k,m)$, so that $\sum_{k=1}^{K}\sum_{m=1}^{K} n_{\{j,l\}}(k,m) = n$. For a specific point (x_j, x_l) in the $\{X_j, X_l\}$ space, let $k_j(x_j)$ and $k_l(x_l)$ denote the indices of the intervals into which x_j and x_l fall, respectively, i.e., $(x_j, x_l) \in N_{\{j,l\}}(k_j(x_j), k_l(x_l))$.

Example 2. We now modify Example 1 by adding an interaction term to the true model and demonstrate the calculation and visualization ALE second-order interaction effects using the `ALEPlot` package. Again, $X = \{X_1, X_2, X_3, X_4\}$ are four independent predictor variables with each $X_j \sim U[0,1]$. Now, the true response is generated as

$$Y = 4x_1 + 3.87{x_2}^2 + 2.97\frac{exp(-5+10x_3)}{1+exp(-5+10x_3)} + 13.86(x_1 - 0.5)(x_2 - 0.5) + \epsilon,$$

where $\epsilon \sim N(0,1)$, and the coefficients 4, 3.87, 2.97, and 13.86 were chosen so that the four terms have approximately the same variance. We generated $n = 5000$ observations from the preceding model and fit a neural network model using the nnet package by Venables and Ripley (2002) with 6 nodes in the single hidden layer, a linear output activation function, and a decay parameter of 0.1, which were approximately optimal according to 10-fold cross-validation. We use $K = 500$ for the main effect plots and $K = 100$ for the second-order effect plots and used the following R code to generate them:

```
## R code for Example 2
## Load relevant packages
library(ALEPlot)
library(nnet)

## Generate some data and fit a neural network supervised learning model
n = 5000
x1 <- runif(n, min = 0, max = 1)
x2 <- runif(n, min = 0, max = 1)
x3 <- runif(n, min = 0, max = 1)
x4 <- runif(n, min = 0, max = 1)
y = 4*x1 + 3.87*x2^2 + 2.97*exp(-5+10*x3)/(1+exp(-5+10*x3))+
13.86*(x1-0.5)*(x2-0.5)+ rnorm(n, 0, 1)
DAT <- data.frame(y, x1, x2, x3, x4)
nnet.DAT <- nnet(y~., data = DAT, linout = T, skip = F, size = 6,
decay = 0.1, maxit = 1000, trace = F)

## Define the predictive function
yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata,
type = "raw"))

## Calculate and plot the ALE main effects of x1, x2, x3, and x4
ALE.1 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = 1, K = 500,
NA.plot = TRUE)
ALE.2 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = 2, K = 500,
NA.plot = TRUE)
ALE.3 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = 3, K = 500,
NA.plot = TRUE)
ALE.4 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = 4, K = 500,
NA.plot = TRUE)

## Calculate and plot the ALE second-order effects of {x1, x2} and {x1, x4}
ALE.12 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = c(1,2), K = 100,
NA.plot = TRUE)
ALE.14 = ALEPlot(DAT[,2:5], nnet.DAT, pred.fun = yhat, J = c(1,4), K = 100,
NA.plot = TRUE)
```

```
## Manually plot the ALE main effects on the same scale for easier comparison
## of the relative importance of the four predictor variables
par(mfrow = c(3,2))
plot(ALE.1$x.values, ALE.1$f.values, type="l", xlab="x1",
ylab="ALE_main_x1", xlim = c(0,1), ylim = c(-2,2), main = "(a)")
plot(ALE.2$x.values, ALE.2$f.values, type="l", xlab="x2",
ylab="ALE_main_x2", xlim = c(0,1), ylim = c(-2,2), main = "(b)")
plot(ALE.3$x.values, ALE.3$f.values, type="l", xlab="x3",
ylab="ALE_main_x3", xlim = c(0,1), ylim = c(-2,2), main = "(c)")
plot(ALE.4$x.values, ALE.4$f.values, type="l", xlab="x4",
ylab="ALE_main_x4", xlim = c(0,1), ylim = c(-2,2), main = "(d)")
## Manually plot the ALE second-order effects of {x1, x2} and {x1, x4}
image(ALE.12$x.values[[1]], ALE.12$x.values[[2]], ALE.12$f.values, xlab = "x1",
ylab = "x2", main = "(e)")
contour(ALE.12$x.values[[1]], ALE.12$x.values[[2]], ALE.12$f.values, add=TRUE,
drawlabels=TRUE)
image(ALE.14$x.values[[1]], ALE.14$x.values[[2]], ALE.14$f.values, xlab = "x1",
ylab = "x4", main = "(f)")
contour(ALE.14$x.values[[1]], ALE.14$x.values[[2]], ALE.14$f.values, add=TRUE,
drawlabels=TRUE)
```

Figures 5(a) - (d) show the ALE main effect plots of x_1 - x_4. Figures 5(e) and (f) show the ALE second-order effect plots for $\{x_1, x_2\}$ and $\{x_1, x_4\}$ respectively. From Figures 5(a) - (d), we see that with the addition of the interaction term, the ALE main effect plots still capture the correct linear, quadratic, sigmoidal, and constant (zero) relationships very well. From Figures 5(e) and 5(f), we see that the interaction between x_1 and x_2 is significant while the interaction between x_1 and x_4 is almost negligible. Specifically, the plot of $\hat{f}_{\{1,2\},ALE}(x_1, x_2)$ looks similar to the contour plot of a hyperbolic parabola, and the plot of $\hat{f}_{\{1,4\},ALE}(x_1, x_4)$ is close to the contour plot of the zero function (taking into account the scale of the contour values shown in the figures). These results agree with the fact that the true model has strong interaction between x_1 and x_2 and no interaction between x_1 and x_4.

Regarding interpretation of the $\{x_1, x_2\}$ interaction effect, consider the dependence of f on x_2 when x_1 is fixed. From Figure 5(e), when we fix (say) $x_1 = 0.2$, increasing x_2 decreases $\hat{f}_{\{1,2\},ALE}(x_1, x_2)$, and when we fix (say) $x_1 = 0.8$, increasing x_2 increases $\hat{f}_{\{1,2\},ALE}(x_1, x_2)$. Thus, Figure 5(e) indicates that $\{x_1, x_2\}$ have positive (reinforcement) interaction. Users should keep in mind that, by definition, ALE second-order effects have zero ALE main effects, since the latter is subtracted from the former when we do the 'centering'. Hence, in order to understand the dependence of f on x_2 for different fixed values of x_1, one should look at the function $\hat{f}_{2,ALE}(x_2) + \hat{f}_{\{1,2\},ALE}(x_1, x_2)$ versus x_2, i.e., the ALE main effect of x_2 added to the ALE interaction effect of $\{x_1, x_2\}$. For fixed $x_1 = 0.2$, the true effect of x_2 on Y is a milder $3.87{x_2}^2 - 4.16x_2 + constant$; and for fixed $x_1 = 0.8$, the true effect of x_2 on Y is a stronger $3.87{x_2}^2 + 4.16x_2 + constant$, which agrees quite closely with the ALE estimate $\hat{f}_{2,ALE}(x_2) + \hat{f}_{\{1,2\},ALE}(x_1, x_2)$ with $x_1 = 0.2$ and $x_1 = 0.8$ plugged in.

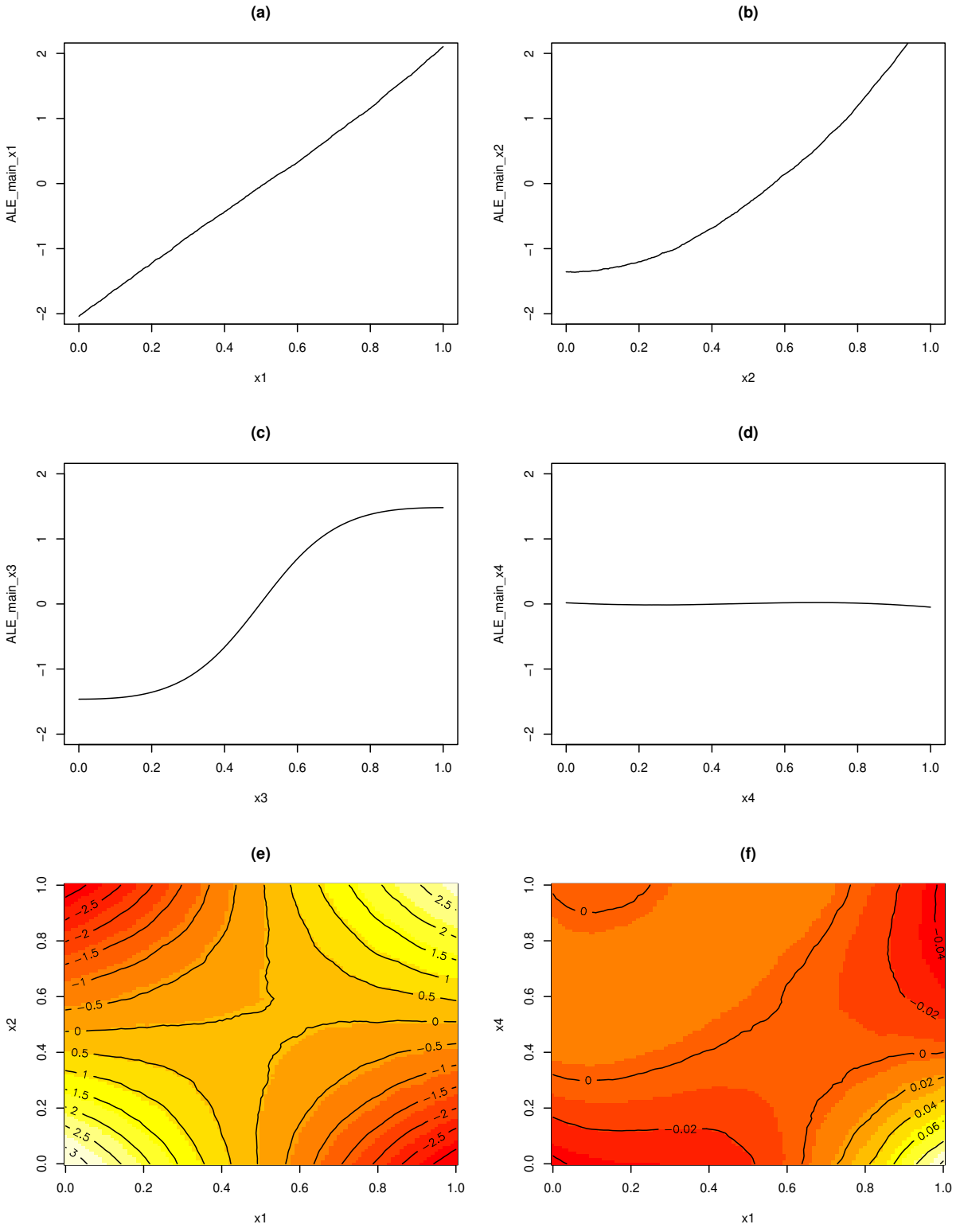

Figure 5: ALE main effect plots: (a) $\hat{f}_{1,ALE}(x_1)$, (b) $\hat{f}_{2,ALE}(x_2)$, (c) $\hat{f}_{3,ALE}(x_3)$, and (d) $\hat{f}_{4,ALE}(x_4)$ and ALE second-order effect plots: (e) $\hat{f}_{\{1,2\},ALE}(x_1, x_2)$, and (f) $\hat{f}_{\{1,4\},ALE}(x_1, x_4)$ for the fitted neural network model in Example 2. Figures 5(a) - (d) capture the correct linear, quadratic, sigmoidal, and constant (zero) relationships for x_1 - x_4 well. Figure 5(e) reveals significant interaction between x_1 and x_2. Figure 5(f) shows that the interaction between x_1 and x_4 is negligible.

Apley (2016) discusses additional properties and characteristics of ALE effects. By using a conditional expectation, as opposed to a marginal expectation, in the definitions (5) and (7) of ALE effects, ALE plots avoid the extrapolation problem of PD plots depicted in Figure 1(a). Moreover, by averaging the local effect (i.e., the partial derivative) rather than the function f itself, ALE plots do not suffer from the OVB problem that renders M Plots of little use for understanding the effects of the individual predictors. In addition to overcoming the extrapolation and OVB problems, ALE plots enjoy a type of additive unbiasedness property for dependent or independent predictors and a multiplicative unbiasedness property for independent subsets of predictors, just as PD plots do. ALE plots are also far less computationally expensive than PD plots, and the computational expense does not depend on the choice of K. Finally, third- and higher-order ALE interaction effects can be defined and estimated in a similar way as ALE main effects and second-order interaction effects. However, the `ALEPlot` does not consider them, because they are less prevalent and much more difficult to interpret than main effects and second-order interaction effects. See Apley (2016) for details.

4 A More Complex Example with Real Data

We now walk through an example considered in Apley (2016), in which a binarized version of household income (above or below a threshold) is predicted as a function of a number of other demographic predictor variables. The data are a compilation of the 1994 US Census data from the University of California Irvine Machine learning repository at

http://archive.ics.uci.edu/ml/datasets/Census+Income.

Example 3. Income Data Example

To reproduce these results using the code below, readers should first obtain the 'adult.data' file from the above link and save it as a csv file. There are $n = 30,162$ cases in the training data set (after removing cases with missing data), and each case represents a person. The response is the binary categorical variable indicating whether a person earned more than \$50k income in 1994. The $d = 12$ predictor variables that we use below are: age (x_1, numerical); working class (x_2, categorical with 8 categories); education level (x_3, treated as numerical:1 = preschool, 2 = 1st-4th grade, 3 = 5th-6th grade, 4 = 7th-8th grade, 5 = 9th grade, 6 = 10th grade, 7 = 11th grade, 8 = 12th grade, 9 = high school graduate, 10 = some college, 11 = vocational associates degree, 12 = academic associates degree, 13 = bachelor's degree, 14 = master's degree, 15 = professional degree, 16 = doctorate degree); marital status (x_4, categorical with 7 categories); occupation (x_5, categorical with 13 categories); relationship status (x_6, categorical with 6 categories); race (x_7, categorical with 5 categories); sex (x_8, categorical with 2 categories); capital gains (x_9, numerical); capital loss (x_{10}, numerical); hours-per-week spent working (x_{11}, numerical); and native country (x_{12}, categorical with 41 categories). Note that the code below removes the original third and fourth predictors, which we do not use. We fit a boosted tree using the R gbm package by Ridgeway and with contributions from others (2015) with parameters shrinkage = 0.02 and interaction.depth = 3, for which the optimal number of trees (determined via 10-fold cross-validation) was 6,000. As $f(x)$ for constructing ALE plots, we used the log-odds of the predicted probability that a person makes over \$50k dollars from the fitted boosted tree. Figure 6 shows the ALE main effect plots for the age, education level, and hours-per-week predictors and the ALE second-order interaction plot $f_{\{1,11\},ALE}(x_1, x_{11})$ for {age, hours-per-week}. We used $K = 500$ for the main effects plots and $K = 50$ for the interaction plot. The following R code can be used to generate the plots:

```
## R code for Example 3
## Load relevant packages
library(ALEPlot)
library(gbm)

## Read data and fit a boosted tree supervised learning model
data = read.csv("adult_data.csv", header = TRUE, strip.white = TRUE,
na.strings = "?")
data = na.omit(data)
gbm.data <- gbm(income==">50K" ~ ., data= data[,-c(3,4)],
distribution = "bernoulli", n.trees=6000,  shrinkage=0.02,
interaction.depth=3)

## Define the predictive function; note the additional arguments for the
## predict function in gbm
yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata,
n.trees = 6000, type="link"))

## Calculate and plot the ALE main and interaction effects for x_1, x_3,
## x_11, and {x_1, x_11}
par(mfrow = c(2,2), mar = c(4,4,2,2)+ 0.1)
ALE.1=ALEPlot(data[,-c(3,4,15)], gbm.data, pred.fun=yhat, J=1, K=500,
NA.plot = TRUE)
ALE.3=ALEPlot(data[,-c(3,4,15)], gbm.data, pred.fun=yhat, J=3, K=500,
NA.plot = TRUE)
ALE.11=ALEPlot(data[,-c(3,4,15)], gbm.data, pred.fun=yhat, J=11, K=500,
NA.plot = TRUE)
ALE.1and11=ALEPlot(data[,-c(3,4,15)], gbm.data, pred.fun=yhat, J=c(1,11),
K=50, NA.plot = FALSE)
```

Regarding interpretation of the results, the ALE main effects plots in Figure 6 have clear interpretations. The probability of earning more than \$50k (i) gradually increases with age until it peaks around 50 years and then gradually declines; (ii) monotonically increases with education level, with the largest jumps occurring when going from Associates to Bachelor's, from Bachelor's to Master's, and from Master's to Ph.D./Professional degrees; and (iii) monotonically increases with hours per week worked up until about 50 hours per week, with the steepest increases between roughly $30 - 50$ hours per week.

The ALE second-order {age, hours-per-week} interaction plot in Figure 6 also reveals an interesting relationship. Consider the increased probability of earning more than \$50k that is associated with increasing hours-per-week from 35 to 80. From the interaction plot, the amount that this probability increases depends on age. For 25-year-olds, the increase in probability when going from 35 to 80 hours-per-week is larger than for 75-year-olds, because $\hat{f}_{\{1,11\},ALE}(x_1, x_{11})$ increases by 0.3 units for 25-year-olds but decreases by 0.5 units for 75-year-olds when going from 35 to 80 hours-per-week. Perhaps this is because 75-year-olds who work so many hours may be more compelled to do so for financial reasons than 25-year-olds (or perhaps there are other explanations).

A word of caution regarding interpreting the ALE interaction plots is again in order. By definition, $\hat{f}_{\{1,11\},ALE}(x_1, x_{11})$ has no x_1 or x_{11} ALE main effects, because they are subtracted from it. Thus, the fact that $\hat{f}_{\{1,11\},ALE}(x_1, x_{11})$ decreases by 0.5 for 75-year-olds going from 35 to 80 hours-per-week does not imply that such an increase in hours-per-week is associated with a decrease in the probability of a 75-year-old earning more than \$50k. To gauge this, we must look at whether $\hat{f}_{11,ALE}(x_{11}) + \hat{f}_{\{1,11\},ALE}(75, x_{11})$ increases or decreases when going from $x_{11} = 35$ to $x_{11} = 80$ hours per week. From Figure 6 this still increases by about 0.6 units for

75-year-olds, so at any age, increasing hours per week worked is associated with an increase in the probability of earning more than \$50k.

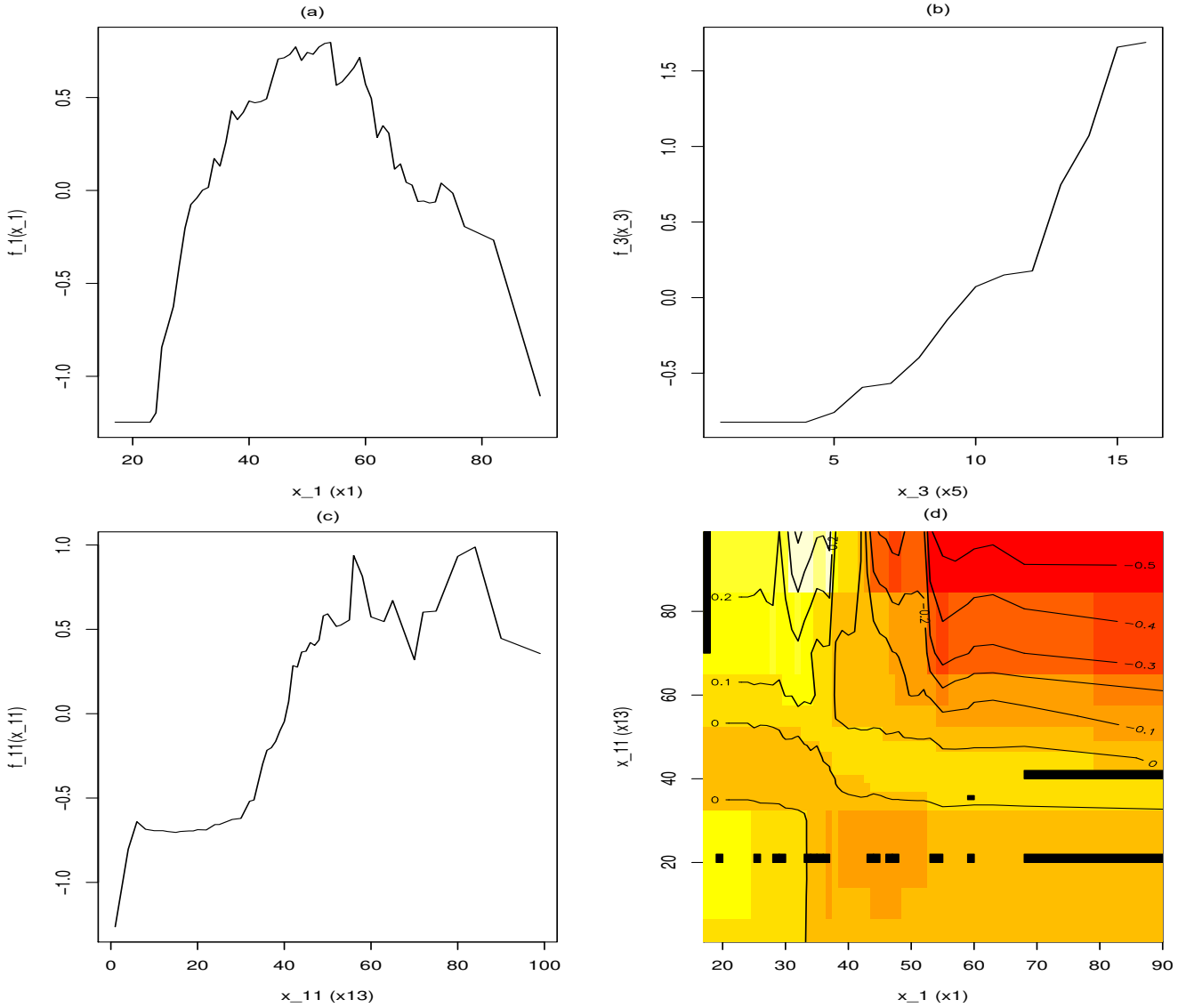

Figure 6: For the income data example using the boosted tree log-odds as $f(x)$, ALE main effect plots for age, education level, and hours-per-week (top panels and bottom left panel) and ALE second-order interaction plot for {age, hours-per-week} (bottom right panel). The black rectangles in the interaction plot indicate empty cells, into which none of the training observations fell.

5 Comparing ALE and PD Plots: An Example where PD Plots break down

We have discussed the advantages of ALE plots throughout the vignette (i.e., they do not suffer from the extrapolation or OVB problems and are computationally simpler than existing visualization methods). In this section, we show an example where ALE plots are reliable, while PD plots break down, because of the extrapolation problem.

Example 4. Simulated example with dependent predictors

Suppose $X = \{X_1, X_2\}$ follows a uniform distribution along the line $x_2 = x_1$ with independent $N(0, 0.05^2)$ variables added to both predictors. The true response is generated as $Y = X_1 + X_2^2 + \epsilon$ with $\epsilon \sim N(0, 0.1^2)$. We generated $n = 200$ observations from this model and fit a neural network model using the nnet package by Venables and Ripley (2002) with 10 nodes in the single hidden layer, a linear output activation function, and a decay parameter of 0.0001. These parameters were again chosen as approximately optimal via multiple replicates of 10-fold cross-validation. The following R code can be used to generate ALE and PD main effects plots for x_1 and x_2 :

```
## R code for Example 4
## Load relevant packages
library(ALEPlot)
library(nnet)

## Generate some data and fit a neural network supervised learning model
n = 200
x <- runif(n, min = 0, max = 1)
x1 <- x + rnorm(n, 0, 0.05)
x2 <- x + rnorm(n, 0, 0.05)
y = x1 + x2^2 + rnorm(n, 0, 0.1)
DAT = data.frame(y, x1, x2)
nnet.DAT <- nnet(y  ~ ., data = DAT, linout = T, skip = F, size = 10,
decay = 0.0001,  maxit = 1000, trace = F)

## Define the predictive function`
yhat <- function(X.model, newdata) as.numeric(predict(X.model, newdata,
type= "raw"))

## Calculate and plot the ALE and PD main effects of x1 and x2
par(mfrow = c(2,2), mar = c(4,4,2,2) + 0.1)
ALE.1 = ALEPlot(DAT[,2:3], nnet.DAT, pred.fun = yhat, J = 1, K = 50,
NA.plot = TRUE)
PD.1 = PDPlot(DAT[,2:3], nnet.DAT, pred.fun = yhat, J = 1, K = 50)
ALE.2 = ALEPlot(DAT[,2:3], nnet.DAT, pred.fun = yhat, J = 2, K = 50,
NA.plot = TRUE)
PD.2 = PDPlot(DAT[,2:3], nnet.DAT, pred.fun = yhat, J = 2, K = 50)

## Manually plot the ALE main effects on the same scale for easier
## comparison of the relative importance of the four predictor variables
## We also plot the true linear and quadratic effects in black for reference
plot(ALE.1$x.values, ALE.1$f.values, type="l", xlab="x1",
     ylab="ALE_main_x1", xlim = c(0,1), ylim = c(-1,1), col = "blue", main = "(a)")
curve(x - 0.5, from = 0, to = 1, add = TRUE)
plot(PD.1$x.values, PD.1$f.values, type="l", xlab="x2",
     ylab="PD_x1", xlim = c(0,1), ylim = c(-1,1), col = "blue", main = "(b)")
curve(x - 0.5, from = 0, to = 1, add = TRUE)
plot(ALE.2$x.values, ALE.2$f.values, type="l", xlab="x3",
     ylab="ALE_main_x2", xlim = c(0,1), ylim = c(-1,1), col = "blue", main = "(c)")
curve(x^2 - (1/3+0.05^2), from = 0, to = 1, add = TRUE)
plot(PD.2$x.values, PD.2$f.values, type="l", xlab="x4",
     ylab="PD_x2", xlim = c(0,1), ylim = c(-1,1), col = "blue", main = "(d)")
curve(x^2 - (1/3+0.05^2), from = 0, to = 1, add = TRUE)
```

The results for a typical replicate of the Example 4 experiment and above R code are shown in Figure 7, from which we see that the ALE main effects (plotted in blue) of x_1 and x_2 (left panels) are much closer to the true linear and quadratic effects (plotted in black) than are the PD main effects (right panels). The reason for the poor accuracy of the PD plots was illustrated in Figure 1(a). Namely, the PD plot requires extrapolation of f far outside the envelope of the $\{x_1, x_2\}$ training data, in which regions the extrapolated f is very inaccurate.

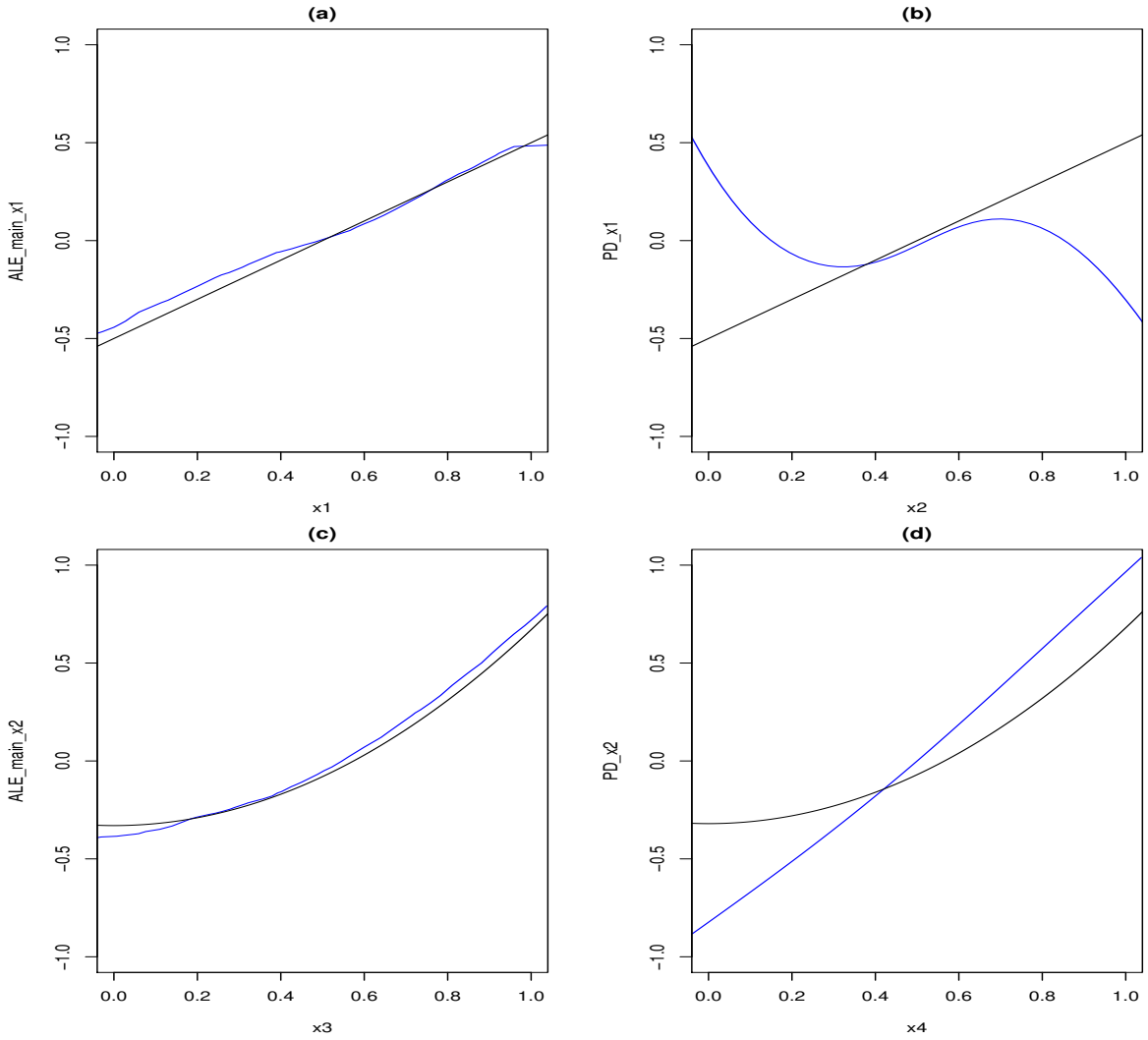

Figure 7: Comparison of (a) $\hat{f}_{1,ALE}(x_1)$, (b) $\hat{f}_{1,PD}(x_1)$, (c) $\hat{f}_{2,ALE}(x_2)$, and (d) $\hat{f}_{2,PD}(x_2)$ for neural network models fitted over one realization of the Example 4 data. In each panel, the black curve is the true effect function (linear for X_1 and quadratic for X_2), and the blue curves are the estimated effect functions for ALE plots (left panels) and PD plots (right panels).

The Example 4 results shown in Figure 7 vary each time the experiment is repeated. We conducted 50 Monte-Carlo replicates of the Example 4 experiment, where on each replicate a different set of $n = 200$ observations are generated from the same distribution and a new neural network model with the same tuning parameters is fit. The results for all 50 replicates are overlaid in Figure 8. Again, the ALE main effects plots (left panels) are far closer to the true linear and quadratic effects for X_1 and X_2, respectively, than are the PD plots (right panels).

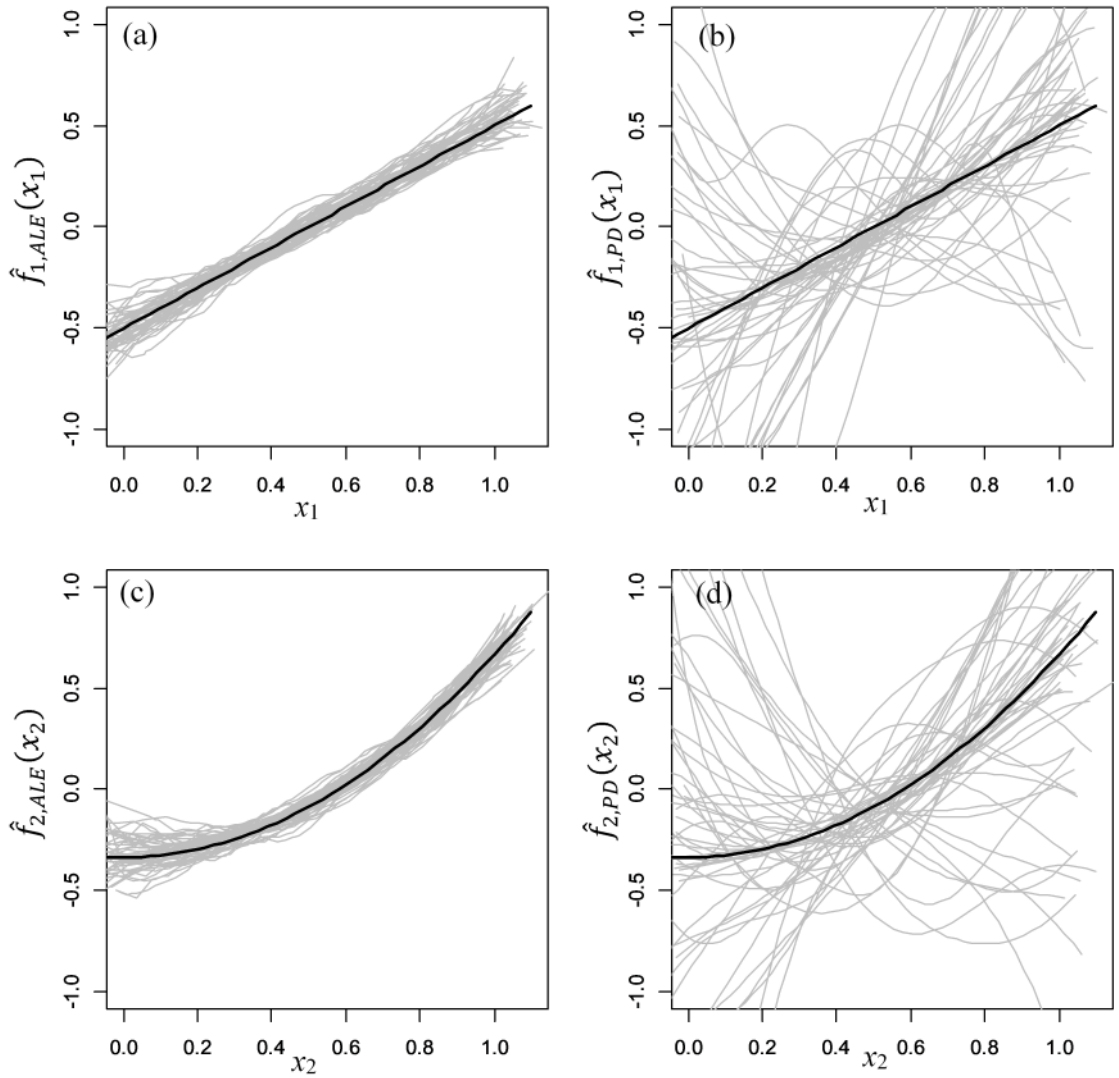

Figure 8: Comparison of (a) $\hat{f}_{1,ALE}(x_1)$, (b) $\hat{f}_{1,PD}(x_1)$, (c) $\hat{f}_{2,ALE}(x_2)$, and (d) $\hat{f}_{2,PD}(x_2)$ for neural network models fitted over 50 Monte Carlo replicates of the Example 1 data. In each panel, the black curve is the true effect function (linear for X_1 and quadratic for X_2), and the gray curves are the estimated effect functions over the 50 Monte Carlo replicates.

References

D. W. Apley. Visualizing the effects of predictor variables in black box supervised learning models. *submitted for publication*, 2016.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

G. Ridgeway and with contributions from others. gbm: Generalized boosted regression models. r package version 2.1.1, 2015. URL `http://CRAN.R-project.org/package=gbm`.

W. N. Venables and B. D Ripley. *Modern Applied Statistics with S. Fourth Edition*. Springer, New York, 2002. ISBN 0-387-95457-0.