Approximate partial dependence plots

Brandon M. Greenwell

January 14, 2022

Introduction

The partial dependence (PD) of the response on a set of predictors is essentially computed by averaging predictions over the observed values of all the other features; see Greenwell [2017] for details. Constructing PD plots can be time consuming and computationally infeasible in practice, especially when dealing with complex models or large data sets. A less accurate, but faster alternative is to fix the other predictors at a "typical" value (e.g., the mean/median for continuous features and the most frequent value for categorical features). This is essentially what's accomplished by default with the `plotmo()` function from the **plotmo** package [Milborrow, 2015][1]; Steven Millborrow, the author of **plotmo**, even refers to them as "a poor man's partial dependence plot" in the package documentation (see `?plotmo::plotmo`).

To this end, I've added a new `exemplar()` function to the package. This function essentially flattens a data frame by summarizing each column by a "typical" value (i.e., the median for numeric columns and the most frequent value for categoricals and factors) into an "exemplar" record. To illustrate, the code chunk below constructs an exemplar record from the Boston housing data frame that's built into **pdp** (see `?pdp::boston` for details):

```
library(pdp)

(boston.ex <- exemplar(boston))  # construct exemplar record
```

```
#>        lon     lat cmedv    crim zn indus chas   nox     rm  age     dis rad
#> 1 -71.0529 42.2181  21.2 0.25651  0  9.69    0 0.538 6.2085 77.5 3.20745   5
#>   tax ptratio      b lstat
#> 1 330   19.05 391.44 11.36
```

Notice that the numeric feature `nox` has been replaced by its median 0.538. Same goes for the rest of the columns. SO, how can we use this to compute faster, but approximate PD plots? Well, a simple trick is to pass the "exemplar" record into the `train` argument in the call to `partial()`, but you'd also have to provide a grid of plotting values via the `pred.grid` argument; see `?pdp::partial` for details on `pred.grid`. This is demonstrated below for a simple random forest fit to the Boston housing data via the awesome **ranger** package [Wright, 2016]; the results are displayed in Figure 1.

```
library(ranger)

# Fit a default random forest to the Boston housing data
set.seed(1228)  # for reproducibility
(rfo <- ranger(cmedv ~ ., data = boston))
```

```
#> Ranger result
#>
#> Call:
#>  ranger(cmedv ~ ., data = boston)
```

[1] As of version 3.3.0, **plotmo** includes support for ordinary PD plots as well.

```
#>
#> Type:                             Regression
#> Number of trees:                  500
#> Sample size:                      506
#> Number of independent variables:  15
#> Mtry:                             3
#> Target node size:                 5
#> Variable importance mode:         none
#> Splitrule:                        variance
#> OOB prediction error (MSE):       10.19975
#> R squared (OOB):                  0.8790243
```

```
# Construct plotting grid; evenly spaced grid of 51 values
lstat.grid <- data.frame("lstat" = seq(
  from = min(boston$lstat), to = max(boston$lstat), length = 51
))

# Approximate PD plot (Figure 1)
partial(rfo, pred.var = "lstat", pred.grid = lstat.grid,
        train = boston.ex, plot = TRUE)
```

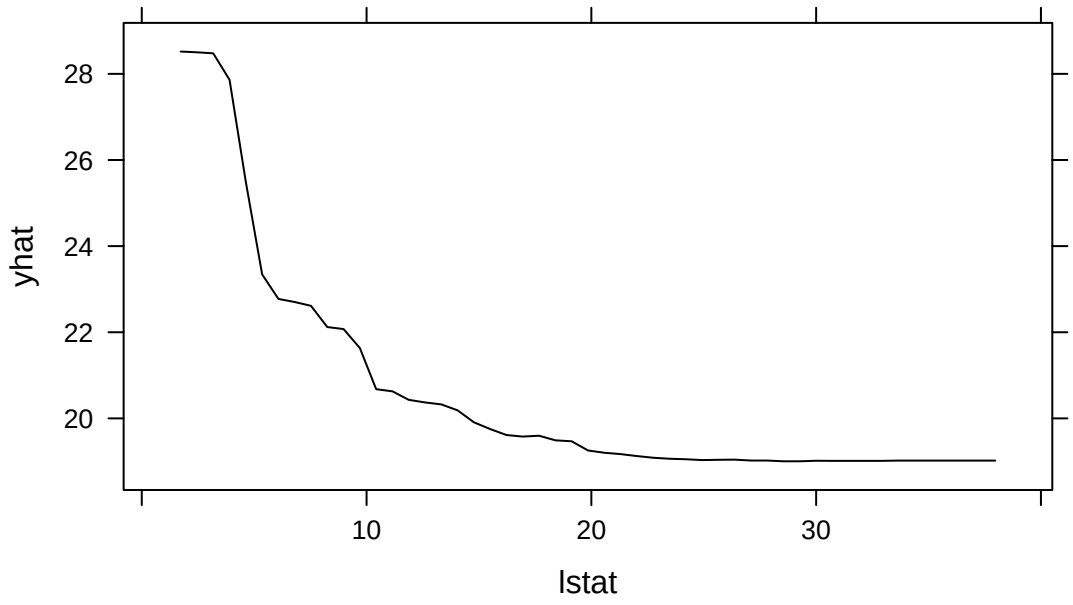

Figure 1: Marginal effect of `lstat` on median home value.

To simplify the construction, you can just set `approx = TRUE` in the call to `partial()`, as demonstrated below (see Figure 2):

```
partial(rfo, pred.var = "lstat", approx = TRUE, plot = TRUE,
        plot.engine = "ggplot2")  # Figure 2

system.time({  # standard PD
  pd1 <- partial(rfo, pred.var = "lstat", grid.resolution = 100)
})
```

```
#>    user  system elapsed
#>   5.397   0.224   2.596
```

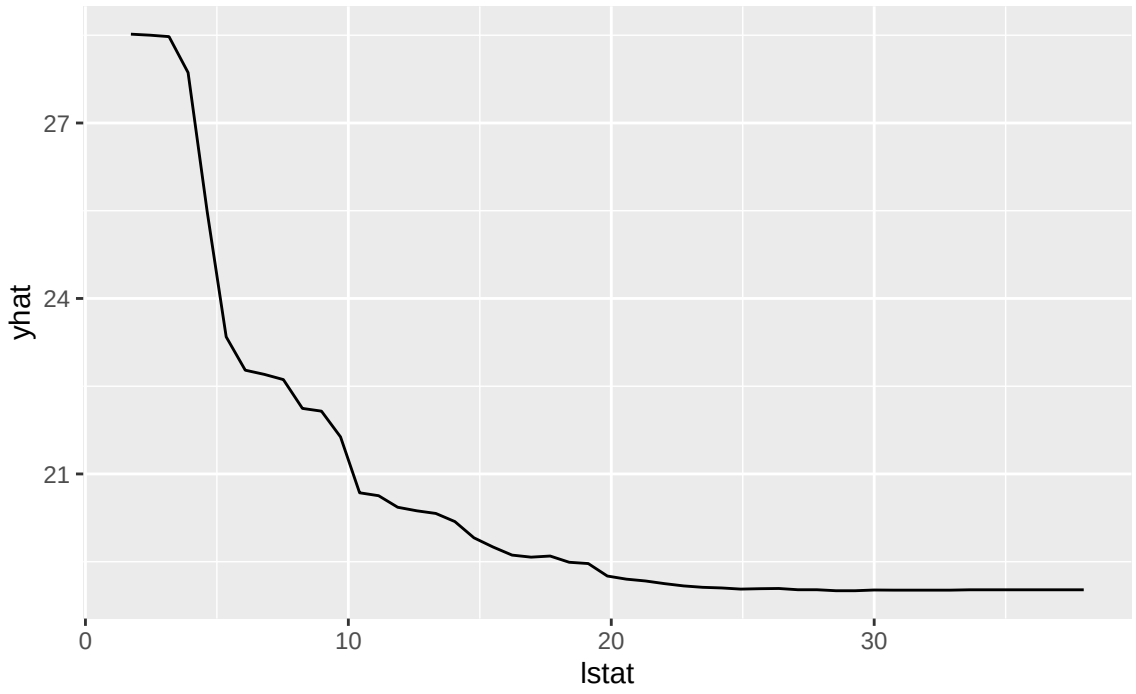

Figure 2: Marginal effect of `lstat` on median home value using the new `approx = TRUE` argument.

```
system.time({  # approximate PD
  pd2 <- partial(rfo, pred.var = "lstat", approx = TRUE, grid.resolution = 100)
})
```

```
#>    user  system elapsed
#>   0.630   0.166   0.690
```

The code chunk below displays the resulting plots in a single display; see Figure **??**. Notice how the two curves are nearly parallel, but the approximate method is much faster to compute.

```
ylim <- range(c(pd1$yhat, pd2$yhat))
palette("Okabe-Ito")
plot(pd1, type = "l", ylim = ylim)
lines(pd2, col = 2)
legend("topright", legend = c("Original PD plot", "Approximate PD plot"),
       inset = 0.01, lty = 1, col = 1:2, bty = "n")
palette("default")
```

As mentioned in the **plotmo** vignette, an approximate PD plot will differ from the original PD plot in the presence of interaction effects. If there are no substantial interaction effects, the two plots will have a similar shape, but may differ slightly in scale.

References

Brandon M. Greenwell. pdp: An R Package for Constructing Partial Dependence Plots. *The R Journal*, 9(1): 421–436, 2017. doi: 10.32614/RJ-2017-016. URL https://doi.org/10.32614/RJ-2017-016.

Stephen Milborrow. *Plotmo: Plot a Model's Response and Residuals*, 2015. URL https://CRAN.R-project.org/package=plotmo. R package version 3.1.4.

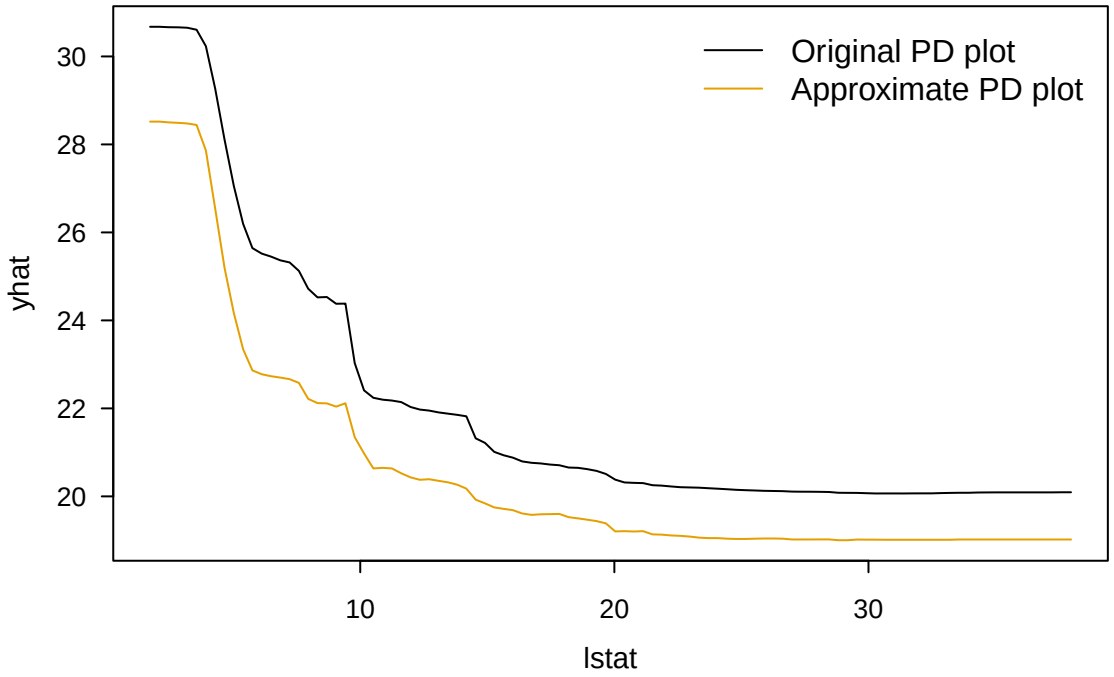

Figure 3: Partial dependence of median home value on `lstat` using the original method (black curve) and approximate method (yellow curve)

Marvin N. Wright. *Ranger: A Fast Implementation of Random Forests*, 2016. URL https://CRAN.R-project.org/package=ranger. R package version 0.6.0.