Automatic Piecewise Linear Regression version 10.0.0

Von Ottenbreit Data Science

Automatic Piecewise Linear Regression (APLR)

- Automatically handles variable selection, non-linear relationships and interactions
- Empirical tests show that APLR is often able to compete with tree-based methods on predictiveness
- APLR produces interpretable models
- APLR can be used for regression and classification tasks, including multiclass classification

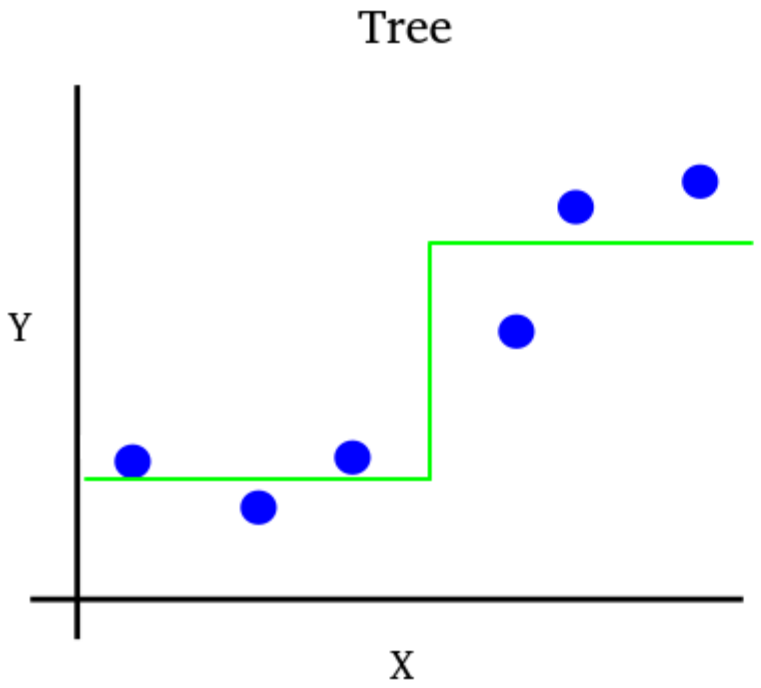
Some useful functionality

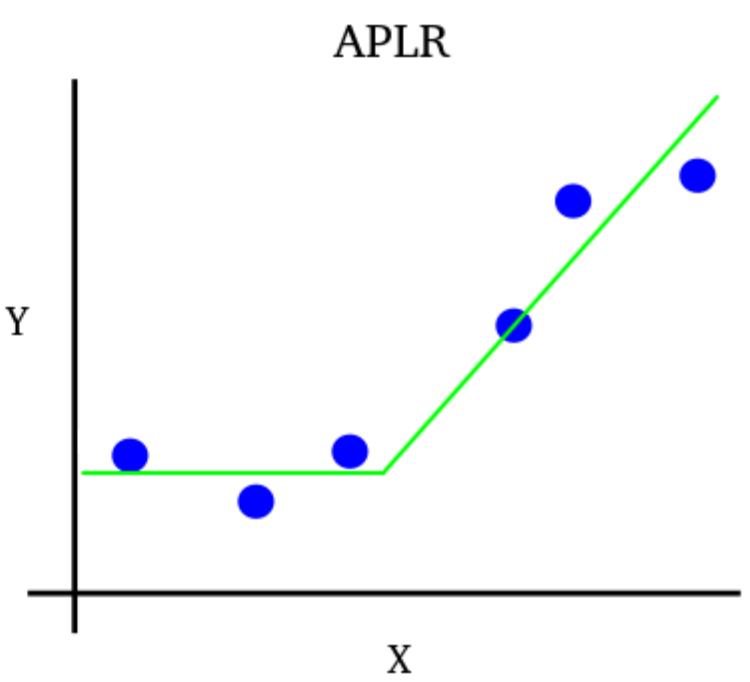
- Allows to specify the loss function among several built-in options such as *mse* (default), *poisson*, *gamma*, etc
- Allows to specify the link function among built in variants: *identity* (default), *logit* and *log*
- Allows to specify the function for calculating the validation set tuning metric among built-in variants such as *default* (same as loss function), *mse*, *negative_gini* etc
- Alternatively allows to pass custom Python functions for each of the above
- The user can specify a list of predictors to be prioritized. Terms based on these will be added or updated in each boosting step as long as training error is reduced. This can be particularly useful to include the full effect of a predictor that is weak but important
- The user can specify a list of predictors with monotonic constraints. This can improve model realism, usually with only a minimal loss increase
- The user can provide constraints on which predictors are allowed in interaction terms
- It is possible to get the shape of each main effect. This makes it easier to interpret main effects
- It is possible to calculate local (observation specific) contributions from an user specified combination of interacting predictors. This makes it easier to interpret interactions (or main effects if just one predictor is specified)
- Regarding classification, the object APLRClassifier fits a logit APLR model for each response category. When predicting, each observation is predicted to the class with the highest predicted class probability

APLR fitting procedure: Important hyperparameters

- *m* (default is 3000) is the maximum boosting steps to try. Should be large enough to minimize the validation error
- *v* (default is 0.1) is the learning rate. Should ideally be <= 0.1. However, for some datasets, especially large ones with a strong signal, a significantly higher learning rate may be useful in order to speed up the training. It is optionally possible to provide predictor specific learning rates (for example in order to put more or less emphasis on certain predictors). The latter can be done in the *fit* method by passing the *predictor_learning_rates* parameter
- *max_interaction_level* (default is 1) specifies the maximum allowed interaction depth. Should be tuned by for example doing a grid search
- *min_observations_in_split* (default is 20) specifies the minimum number of training observations where a term must not have a value of zero due to the max, min or indicator functions (number of effective observations). The higher value the lower variance (the term relies on more observations) but higher bias (less fine grained splits). Should be tuned by for example doing a grid search
- *max_terms* (default is 0 which means no limit) specifies the maximum number of terms in each underlying model. Setting a limit may increase model interpretability but can also degrade predictiveness. An optional tuning objective could be to find the lowest positive value of *max_terms* that does not increase the prediction error significantly. Setting a limit with *max_terms* may require a higher learning rate for best results

Some differences compared to tree-based methods





- Trees fit piecewise constants
- APLR fits piecewise linear basis functions or linear effects
- Trees often only fit interactions (unless for example max tree depth is 1)
- APLR fits main effects and potentially interactions (often simpler to interpret)

References

- Link to published article: https://link.springer.com/article/10.1007/s00180-024-01475-4