# Automatic Piecewise Linear Regression version 10.6.0

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 produces smoother predictions than tree-based methods
- APLR can be used for regression and classification tasks, including multiclass classification

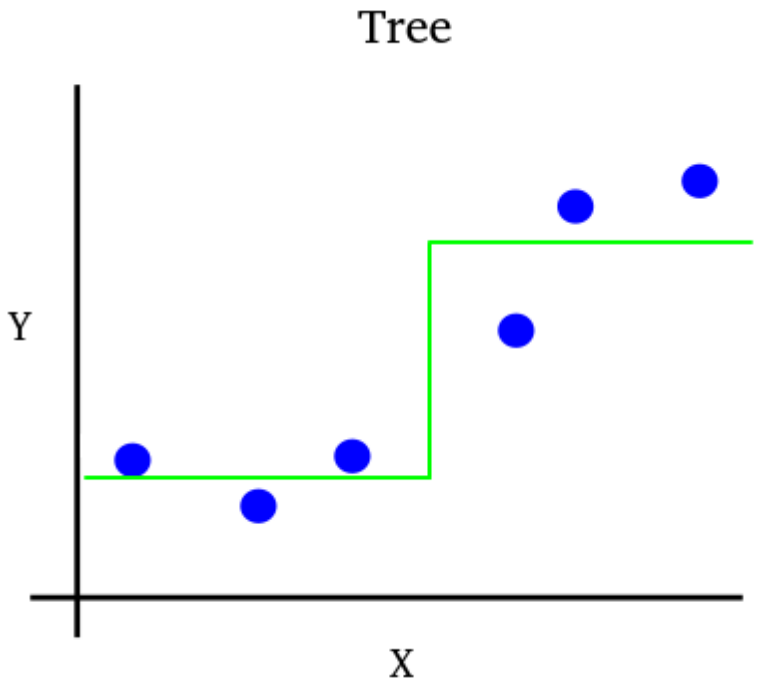
# Some useful functionality

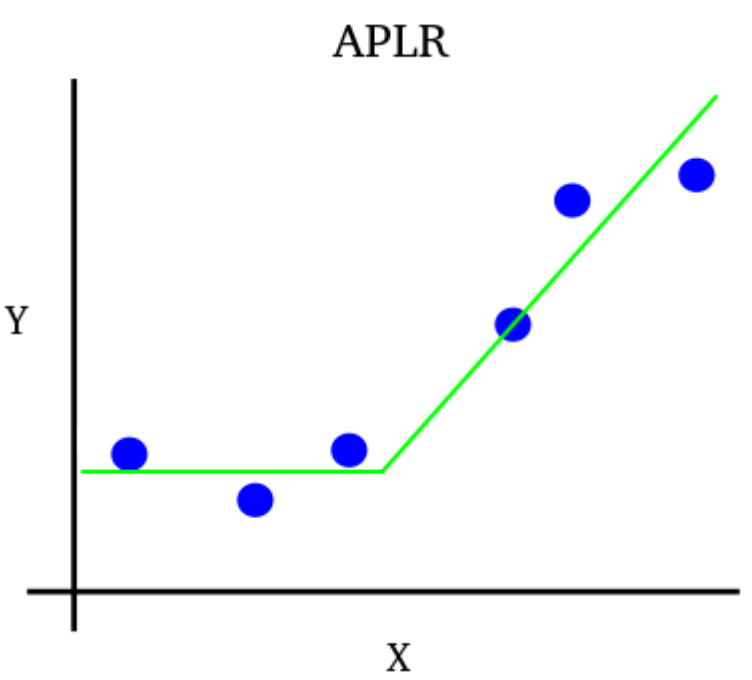
- Allows to specify the loss function among several built-in options such as *mse* (default), *poisson*, *gamma*, etc
- Allows to specify the link function among built in variants: *identity* (default), *logit* and *log*
- Allows to specify the function for calculating the validation set tuning metric 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 and interaction. This makes it easier to interpret the model, for example by plotting the shapes of main effects or two-way interactions
- It is possible to calculate local (observation specific) feature importance as well as local contributions to the linear predictor from each feature in the model
- 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

# Tuning APLR

- APLR often works well with the default settings. However, it is usually possible to get better results by tuning. Below are the most important tuning parameters
- *m* (default is 20000) is the maximum boosting steps to try. Should be large enough to minimize the validation error. The default value should be more than sufficient in most cases. Early stopping will prevent unnecessary computational costs
- *v* (default is 0.5) is the learning rate. Must be greater than zero and not more than one. The higher the faster the algorithm learns and the lower *m* is required, reducing computational costs potentially at the expense of predictiveness. Empirical evidence suggests that *v* <= *0.5* gives good results for APLR (potentially lower for smaller datasets). 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. Tune, for example by doing a grid search, for best predictiveness. For best interpretability use 0 (or 1 if interactions are needed)
- *min_observations_in_split* (default is 2) 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). Tune, for example by doing a grid search, for best predictiveness
- *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
- *boosting_steps_before_interactions_are_allowed* (default is 500) specifies how many boosting steps to wait before searching for interactions. The default value often works well with the default value of *max_interaction_level*. A tuning strategy for best predictiveness and without too many tuning parameters may be to set this parameter to zero and tune *max_interaction_level*

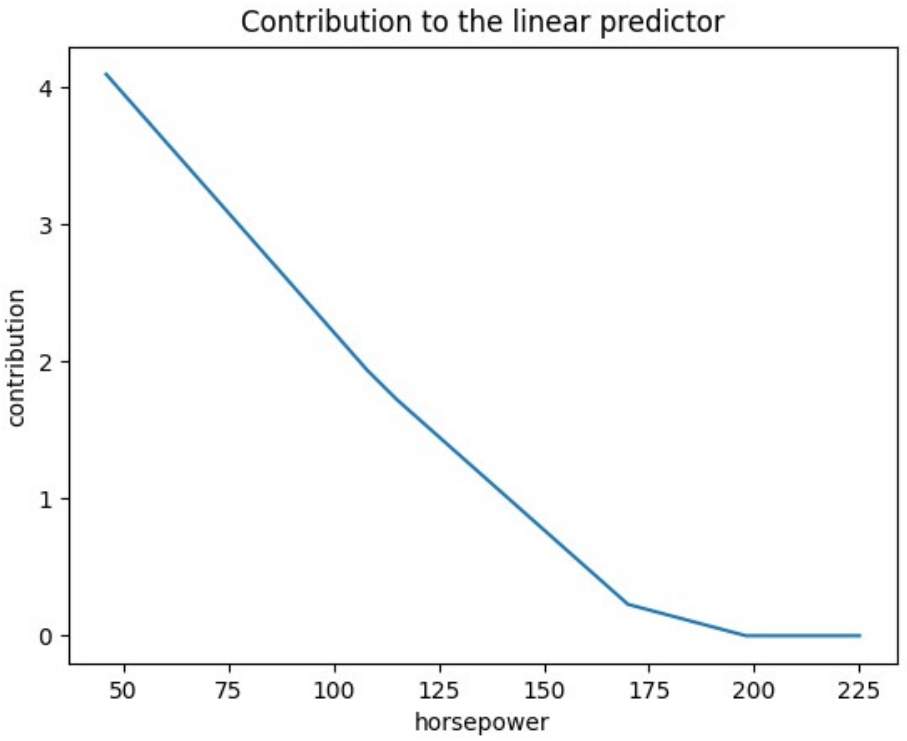
# Some differences compared to tree-based methods

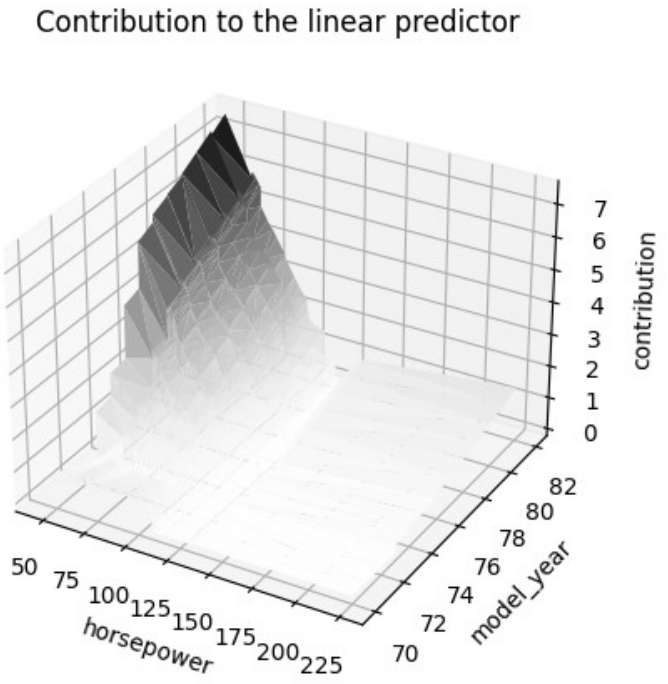



- Trees fit piecewise constants
- APLR fits piecewise linear basis functions or linear effects
- Trees often only fit interactions (unless for example max tree depth is 1)
- APLR fits main effects and potentially interactions (often simpler to interpret)

# Interpretation example on the Auto MPG dataset




- **The APLR model in this example predicts miles per gallon (*mpg*) for cars and was trained on the Auto MPG dataset. The model was allowed to use two-way interactions (*max_interaction_level* = 1).**
- **The above charts were generated from the output of two calls to the *get_unique_term_affiliation_shape* method in APLR.**
- **The left chart shows the combined effect of all non-interaction terms using the *horsepower* predictor. When *horsepower* increases then predicted *mpg* decreases.**
- **The right chart shows the combined effect of all interaction terms between *horsepower* and *model_year*. An increase in *model_year* increases predicted *mpg* and the effect is greatest when *horsepower* is low.**

# References


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