

Today

- Updates to Tue slides & code
 - better intuition and visualization?
- Innovations so far that address bias-variance tradeoff
- Putting it all together
 - stochastic gradient descent
 - extreme gradient boosting, xgboost

Boosting – model or training?

- Boosting = training algorithm
- Model ensemble = model algorithm
 - e.g. for a tree-based model
 - walk each tree to get predictions
 - average the trees, weighted by the learning rate
- In this sense, the model algorithm is the same as bagging or random forests

Innovations so far

- What innovations do we have so far that address the bias-variance tradeoff to give accurate out-of-sample predictions?

Stochastic gradient descent

Gradient descent training algorithm for a linear model

set lambda (learning rate)

make initial guess for β_0, β_1

for many iterations

 randomly sample rows (y, x) ← new step

 find gradient at β_0, β_1

 step down: $\beta = \beta - \text{lambda} * \text{gradient}(\beta)$

print β_0, β_1

random sample options: bootstrap, subset, single points, mini-batch

Stochastic gradient boosting

Training algorithm

load y, x, x_{new}

set $\hat{f}(x_{\text{new}}) = 0$

set $r \leftarrow y$ (residuals equal to the data)

for m in 1 to $n_{\text{iterations}}$

 randomly sample rows ← new step

 train model on r and x

 predict residuals, $\hat{r}_m(x)$, from trained model

 update residuals: $r \leftarrow r - \lambda \hat{r}_m(x)$

 predict y increment, $\hat{f}_m(x_{\text{new}})$, from trained model

 update prediction: $\hat{f}(x_{\text{new}}) \leftarrow \hat{f}(x_{\text{new}}) + \lambda \hat{f}_m(x_{\text{new}})$

return $\hat{f}(x_{\text{new}})$

Extreme gradient boosting

Training algorithm

load y, x, x_{new}

set $\hat{f}(x_{\text{new}}) = 0$

set $r \leftarrow y$ (residuals equal to the data)

for m in 1 to $n_{\text{iterations}}$

 randomly sample rows and columns

 train model on r and x

 predict residuals, $\hat{r}_m(x)$, from trained model

 update residuals: $r \leftarrow r - \lambda \hat{r}_m(x)$

 predict y increment, $\hat{f}_m(x_{\text{new}})$, from trained model

 update prediction: $\hat{f}(x_{\text{new}}) \leftarrow \hat{f}(x_{\text{new}}) + \lambda \hat{f}_m(x_{\text{new}})$

return $\hat{f}(x_{\text{new}})$

Boosting packages in R

- **gbm**: gradient boosting machines
 - boosted decision trees
 - C++
 - regression, classification, + extensions (e.g. survival, count data, quantile regression)
 - fast, good, stable, maintained
 - retired (no new features)

Boosting packages in R

- **gbm3**: successor to gbm
 - apparently mostly abandoned ca 2017 (perhaps due to xgboost?)
 - active development restarted Jan 2024
 - github only for now
 - watch this space?

Boosting packages in R


- **xgboost**: extreme gradient boosting
 - R interface to very fast C++ library
 - many additional algorithm innovations to speed training
 - large data innovations
 - inbuilt parallel and GPU support
 - current industry standard (interfaces to all major data science languages)

xgboost

- slow learning
 - gradient descent, boosting
- random sampling data
 - like bagging, but no replacement
- random sampling variables
 - like random forest, but also per decision node
- regularized cost function
 - like smoothing spline, tree complexity penalty
- best split algorithm optimizes the whole tree
 - not per split

Categorical variables

- “One hot” encoding

habitat		bog	forest	grassland
“forest”		0	1	0
“forest”		0	1	0
“bog”		1	0	0
“grassland”		0	0	1
“bog”		1	0	0
“grassland”		0	0	1
“forest”		0	1	0

- binary categories: only needs one column

xgboost hyperparameters

Tuning strategies

- Tuning is optimization
- Grid search
- Random search
- Evolutionary algorithms
- Bayesian optimization

Bischl et al (2023). Hyperparameter optimization: foundations, algorithms, best practices, and open challenges. <https://doi.org/10.1002/widm.1484>.

ML “pipeline” packages

- R: tidy models
- R: caret (predecessor to tidy models)
- Python: scikit-learn
- Handle
 - data pre-processing (“feature engineering”)
 - test-train splits
 - hyperparameter tuning