Penalized GLM and GBM Guide

Arno Candel, PhD
Tomas Nykodym
Patrick Hall

Penalized Generalized Linear Models

Part 1

Linear Modeling Methods

Ordinary Least Squares

Carl Friedrich Gauss
(1777–1855)

Elastic Net

Hui Zou and Trevor Hastie
Regularization and variable selection via the elastic net,
Journal of the Royal Statistical Society, 2005

Ordinary Least Squares Requirements

Requirements	If broken …
Linear relationship between inputs and targets; normal y, normal errors	Inappropriate application/unreliable results; use a machine learning technique; use GLM
N > **p**	Underspecified/unreliable results; use LASSO or elastic net penalized regression
No strong multicollinearity	Ill-conditioned/unstable/unreliable results; Use ridge(L2/Tikhonov)/elastic net penalized regression
No influential outliers	Biased predictions, parameters, and statistical tests; use robust methods, i.e. IRLS, Huber loss, investigate/remove outliers
Constant variance/no heteroskedasticity	Lessened predictive accuracy, invalidates statistical tests; use GLM in some cases
Limited correlation between input rows (no autocorrelation)	Invalidates statistical tests; use time-series methods or machine learning technique

Anatomy of GLM

Family/distribution defines mean and variance of Y

Nonlinear link function between linear component and E(Y)

$$\boxed{E(\boldsymbol{Y})} = \mu = \boxed{g^{-1}}(\boxed{\boldsymbol{X}\beta})$$

Linear component

$$\boxed{Var(\boldsymbol{Y}) = V(\mu) = V(g^{-1}(\boldsymbol{X}\beta))}$$

Family/distribution allows for non-constant variance

H2O.ai

Distributions / Loss Functions

For **regression** problems, there's a large choice of different distributions and related loss functions:

- **Gaussian** distribution, squared error loss, sensitive to outliers
- **Laplace** distribution, absolute error loss, more robust to outliers
- **Huber** loss, hybrid of squared error & absolute error, robust to outliers
- **Poisson** distribution (e.g., number of claims in a time period)
- **Gamma** distribution (e.g, size of insurance claims)
- **Tweedie** distribution (compound Poisson-Gamma)

- **Binomial** distribution, log-loss for binary classification

Also, H2O supports:
- **Offsets**
- **Observation weights**

Iteratively Reweighed Least Squares

Iteratively reweighted least squares (IRLS) complements model fitting methods in the presence of outliers by:

- Initially setting all observations to an equal weight
    - Fitting GLM parameters (β's)

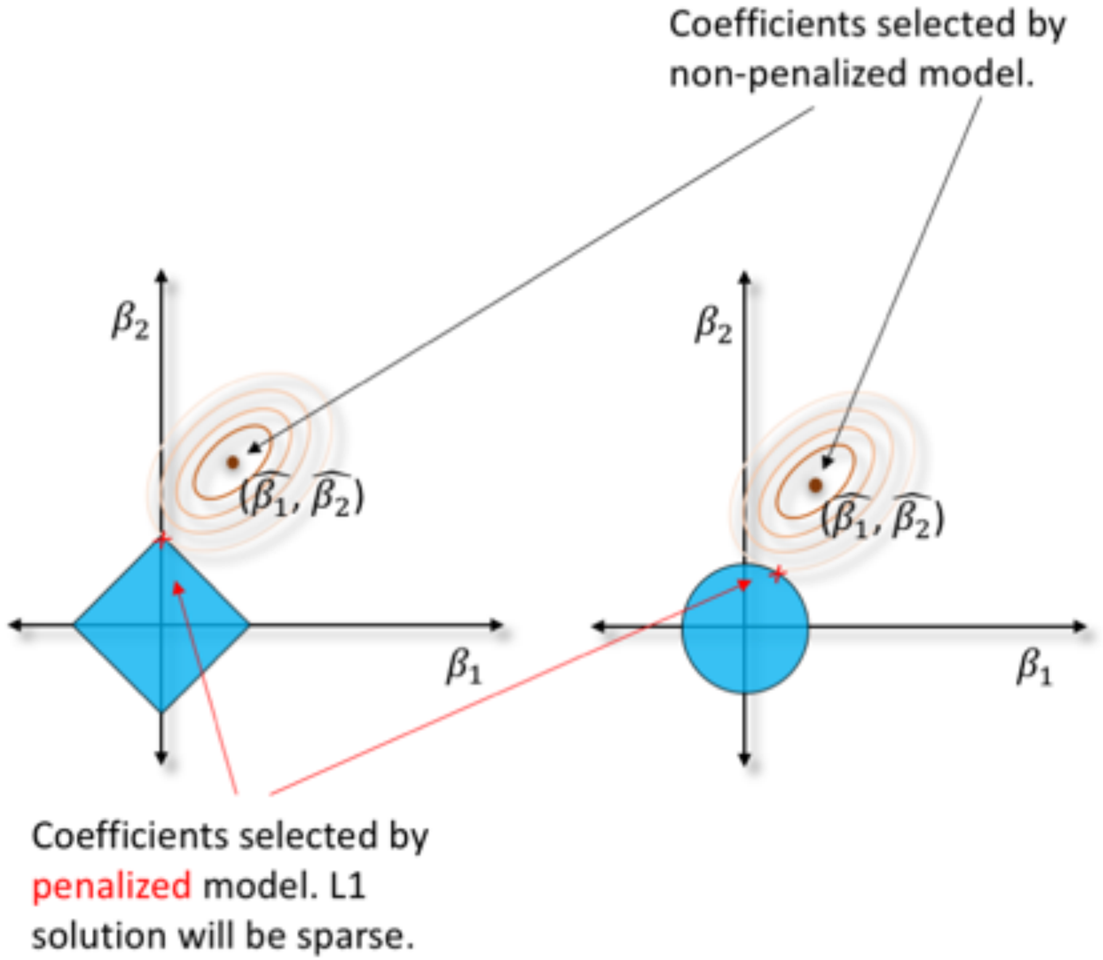
"Outer loop"

$$\tilde{\beta} = \min_{\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} * \beta_j \right)^2 \right\}$$

"Inner loop"
(ADMM optimization in H2O)

- Calculating the residuals of the fitted GLM
- Assigning observations with high residuals a lower weight
- Repeating until GLM parameters (β's) converge

Regularization (e.g. Penalties)



Coefficients selected by non-penalized model.
β_2
$(\widehat{\beta_1}, \widehat{\beta_2})$
β_1
β_2
$(\widehat{\beta_1}, \widehat{\beta_2})$
β_1
Coefficients selected by penalized model. L1 solution will be sparse.

Combining GLM, IRLS and Regularization

λ controls magnitude of penalties. Variable selection conducted by refitting model many times while varying λ. Decreasing λ allows more variables in the model.

"Outermost loop"

L1/LASSO helps with variable selection.

L2/Ridge/Tinkhonov penalty helps address multicollinearity.

$$\tilde{\beta} = \min_{\beta}\left\{\boxed{\sum_{i=1}^{N}\left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} * \beta_j\right)^2} + \boxed{\lambda}\sum_{j=1}^{p}\left(\boxed{\alpha} * \boxed{|\beta_j|} + \boxed{(1-\alpha)} * \boxed{\beta_j^2}\right)\right\}$$

Error function for a GLM.

α tunes balance between L1 and L2 penalties, i.e. elastic net.

- Inner loop: Fitting GLM parameters for a given λ and α
- Outer loop: IRLS until β's converge
- Outermost loop: λ varies from λ_{max} to 0

Elastic net advantages over L1 or L2:

- Does not saturate at min(p, N)
- Allows groups of correlated variables

Hyperparameter Selection

Outer most loop(s):

- λ search from λ_{max} (where all coefficients = 0) to $\lambda = 0$
- Grid search on alpha usually not necessary
    - Just try a few: 0, 0.5, 0.95
    - Always keep some L2
    - Set max_predictors, large models take longer
- Models can also be validated:
    - Validation and test partitioning available
    - Cross-validated (k-fold, CV predictions available)

Practical Pointers

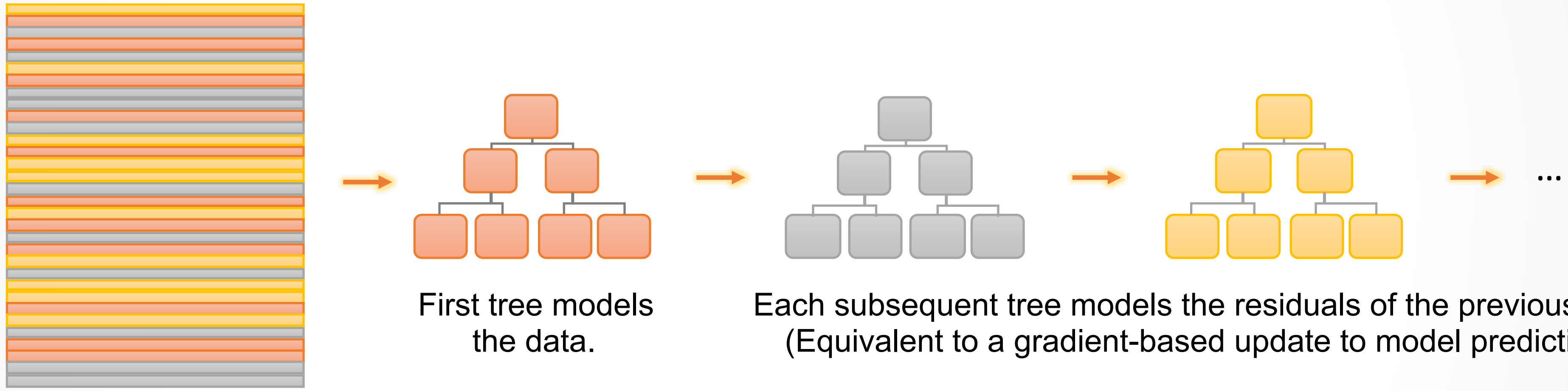
- P-values available for non-penalized models
- β constraints available, i.e. for all positive β's
- Use IRLS optimization for tall, skinny data sets
- L1 **OR** LBFGS for wide data (> 500 predictors)
  - (L1 **AND** LBFGS possible, but can be slower)

Gradient Boosting Machines

Part 2

Gradient Boosting Machines

GBM builds a sequential ensemble of decision trees: each tree attempts to correct the mis-predictions of all previous trees



Data is sampled with replacement for each tree (iteration). Both rows and columns can be sampled.

First tree models the data.

Each subsequent tree models the residuals of the previous trees. (Equivalent to a gradient-based update to model predictions)

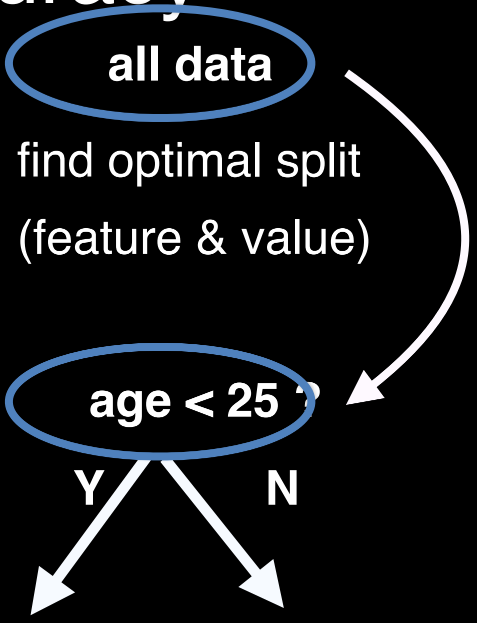
Jerome H. Friedman
Greedy function approximation: a gradient boosting machine
Annals of statistics, 2001

Ensemble models: *Intuition*

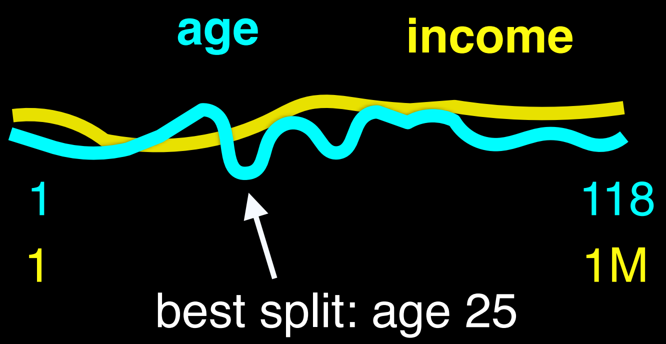
- *Stability: the predictions of ensemble models are stable w.r.t. minor perturbations of training data*
- *Variable hiding: important variables are often correlated and can hide one-another (only the single most important variable from a group of important correlated variables will be used in many models); in different samples, many different important variables can shine through*
- *Representative samples: some samples can be highly representative of new data*

Distributed Gradient Boosting Machine

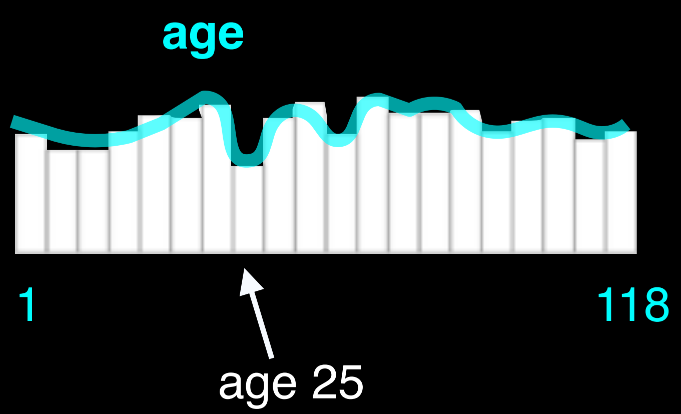
- **H2O: First open-source implementation of scalable, distributed Gradient Boosting Machine - fully featured**
- Each tree is built in parallel on the entire compute cluster
- Discretization (binning) for speedup without loss of accuracy



Analytical error

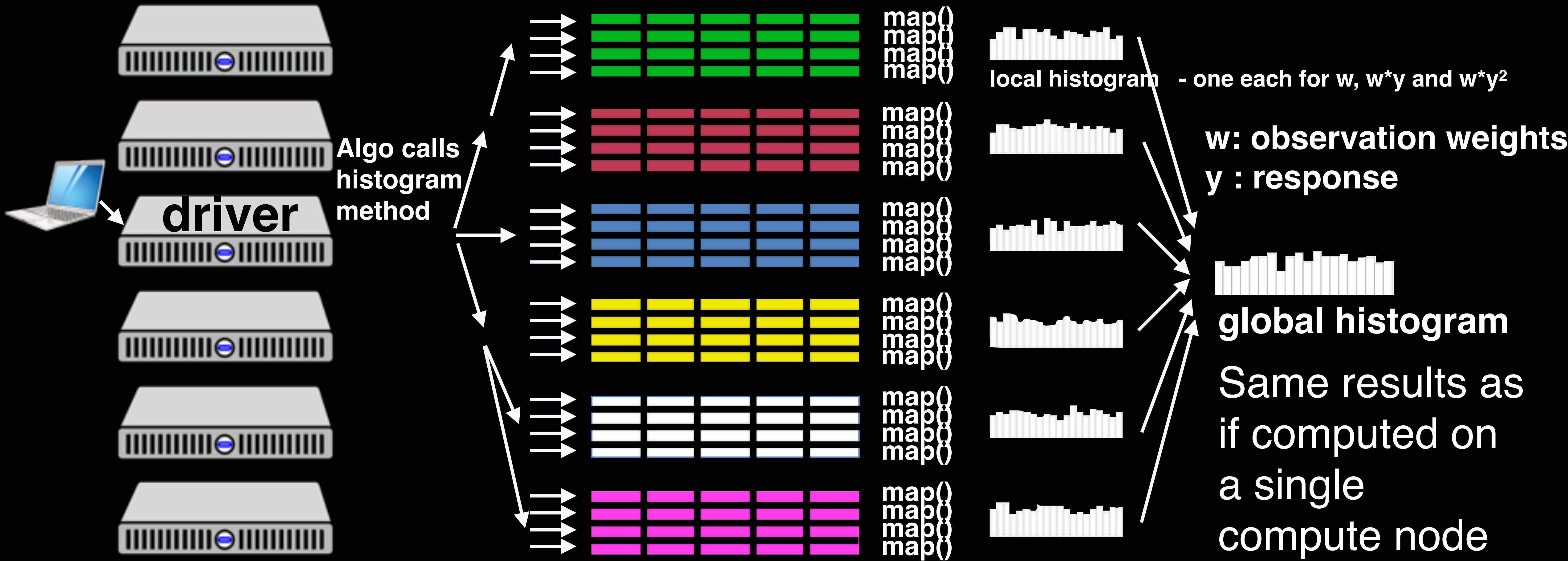


H2O: discretized into



Scalable Distributed Histogram Calculation

data parallelism - global histogram computation

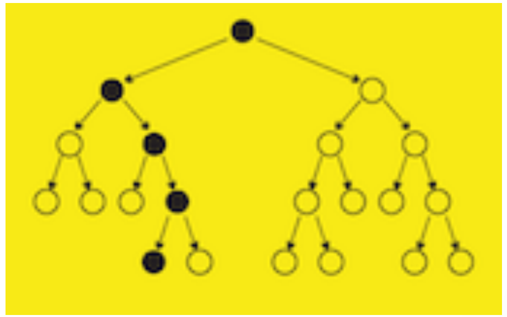


Same results as if computed on a single compute node

Model Complexity

- GBM builds a sequential ensemble of decision trees: each tree attempts to correct the mis-predictions of all previous trees
- Each decision tree partitions the training data iteratively (left vs right)
- Splits are found by an approximated exhaustive search

  (all features, all values)
- Stop splitting if
  - max depth is reached
  - no error decrease from further splitting
- Number of leaves grows exponentially, model complexity (training/ scoring time and storage) grows as O(ntrees * 2^max_depth)
- Training is iterative: can stop at any point and continue training



Validation Schemes

Training data	**Model training**
Validation data	**Parameter tuning**
Testing data	**Final estimate of generalization error**

- **Validation data can be created upfront (e.g., a split)**
  - for big data, it’s usually fine to not use all the data for training
  - 60%/20%/20% train/valid/test splits are usually fine for big data
- **Validation data can be held out in N-fold cross-validation**
  - best for smaller datasets or when validation is critical
  - e.g.: 80%/20% train/test, 5-fold CV on the 80%, final testing on 20%
- **Random shuffling only works for i.i.d. distribution**
- If the data is non-i.i.d., has structure (time, events, etc.), you must stratify!

Validate properly

Establishing Baselines

- **Train default GBM, GLM, DRF, DL models** and inspect convergence plots, train/validation metrics and variable importances

- **Train a few extra default GBM models**, with:

- **max_depth=3**	“Do Interactions matter at all?”
- **max_depth=7**	“I feel lucky”
- **max_depth=10**	“Master memorizer”

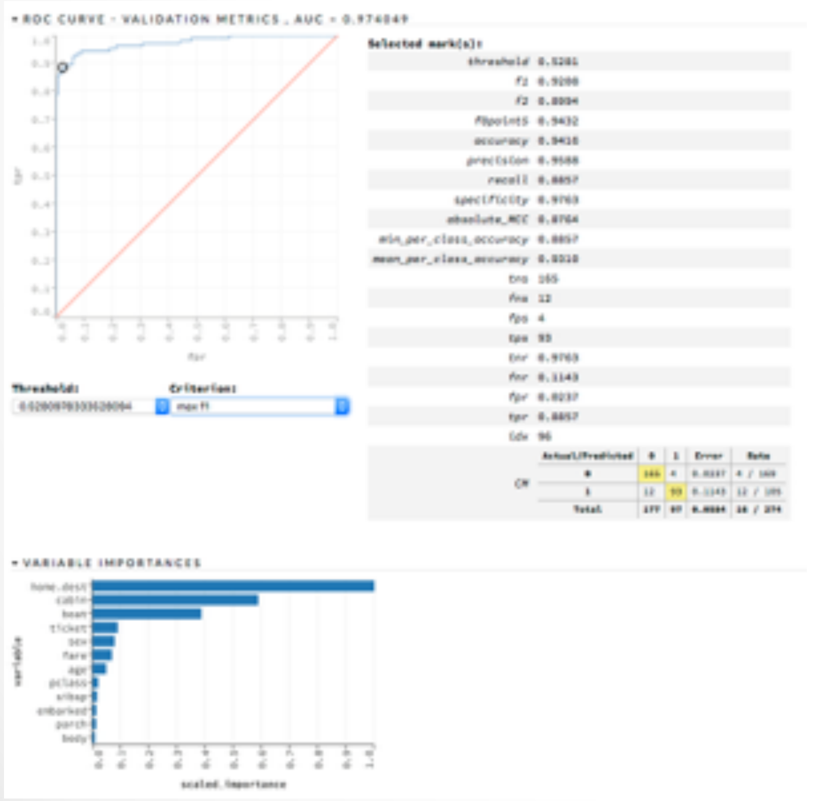
- Develop a feel for the problem and the holdout performance of the different models

Develop a Feel

H2O.ai

Viewing Models in Flow

- Inspect the model in **Flow** during training: **getModel ‘model_id’**
- If the model is **wrong** (wrong architecture, response, parameters, etc.), cancel it
- If the model is **taking too long**, cancel and decrease model complexity
- If the model is **performing badly**, cancel and increase model complexity



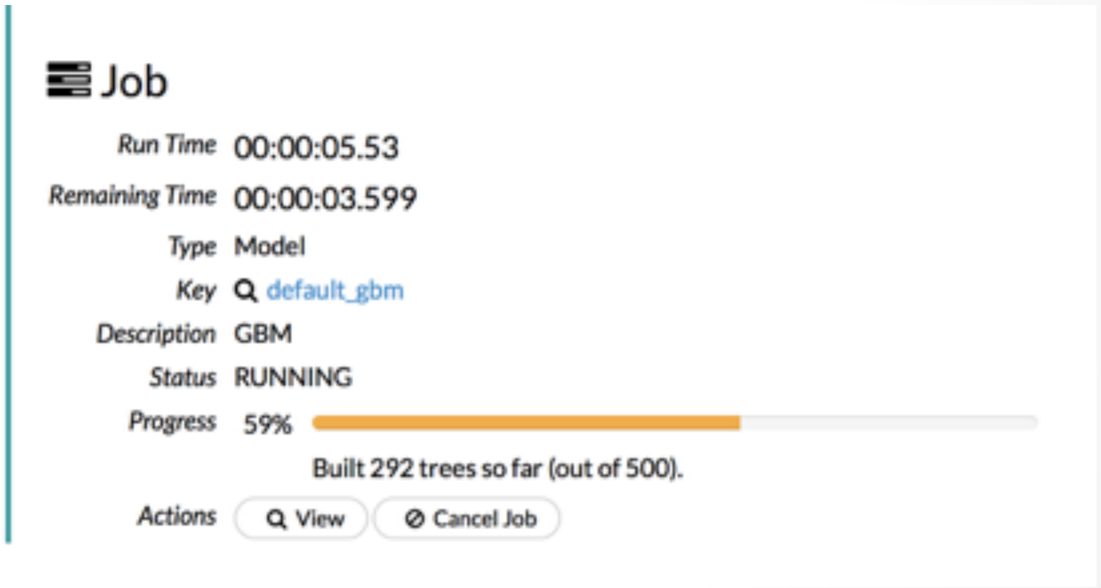
Threshold: 0.5280978...
Criterion: max f1

Selected mark(s):

threshold	0.5281
f1	0.9208
f2	0.8894
f0point5	0.9432
accuracy	0.9416
precision	0.9588
recall	0.8857
specificity	0.9763
absolute_MCC	0.8764
min_per_class_accuracy	0.8857
mean_per_class_accuracy	0.9310
tns	165
fns	12
fps	4
tps	93
tnr	0.9763
fnr	0.1143
fpr	0.0237
tpr	0.8857
idx	96

CM

Actual/Predicted	0	1	Error	Rate
0	165	4	0.0237	4 / 169
1	12	93	0.1143	12 / 105
Total	177	97	0.0584	16 / 274



Job

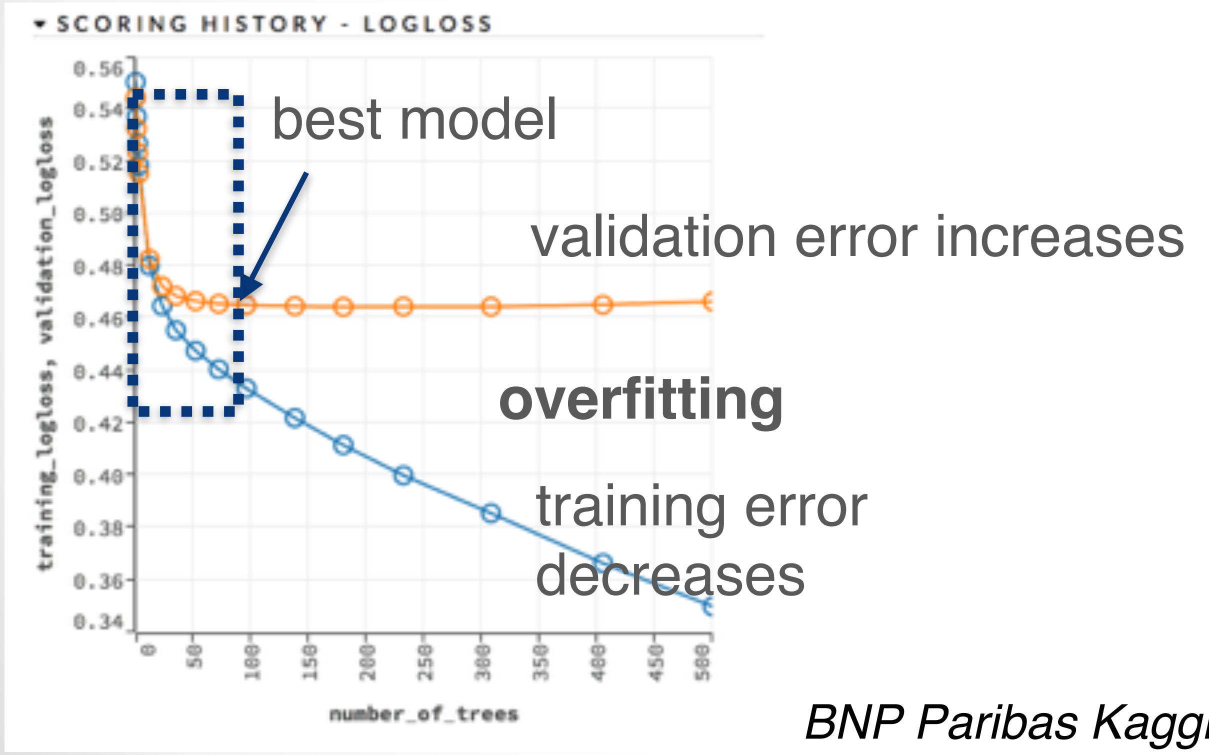
Run Time	00:00:05.53
Remaining Time	00:00:03.599
Type	Model
Key	default_gbm
Description	GBM
Status	RUNNING
Progress	59%
	Built 292 trees so far (out of 500).

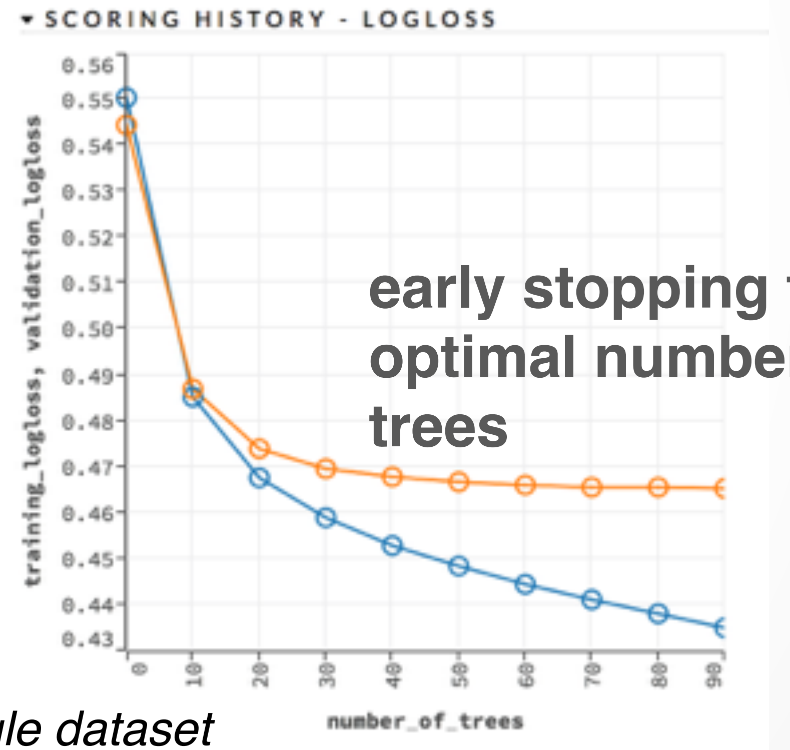
Cancel early, cancel often

Use Early Stopping!
(Off by default)

Before: build as many trees as specified - can overfit on training data

Now: stop training once user-specified metric converges, e.g., **stopping_rounds=2, stopping_metric="logloss", stopping_tolerance=1e-3, score_tree_interval=10**





early stopping finds optimal number of trees

BNP Paribas Kaggle dataset

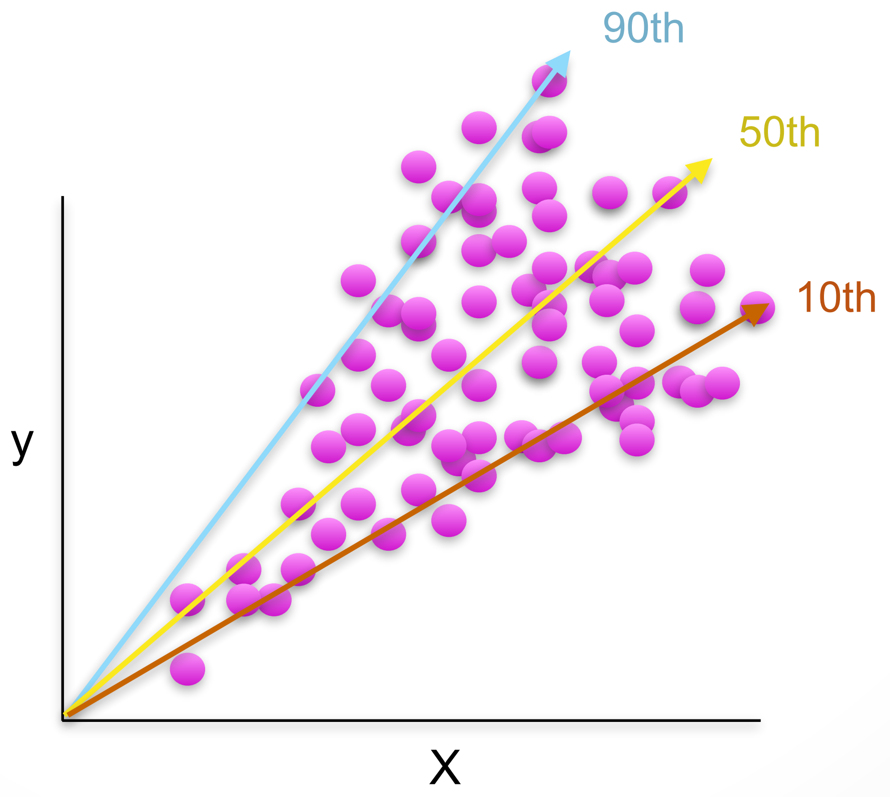
Early stopping saves time and leads to higher accuracy

Distributions / Loss Functions

- For **regression** problems, there's a large choice of different distributions and related loss functions, just like in GLM.

- GBM also supports **Quantile** regression (e.g., predict the 80-th percentile)

Do your math!



HyperParameter Search

There are a lot of hyper parameters for H2O GBM. Many are for expert-level fine control and do not significantly affect the model performance.

The most important parameters to tune for GBM are
(e.g., with a Random search via **strategy = "RandomDiscrete"**):

- **max_depth = [1,2,…,25]**
- **learn_rate = [0.01,0.02,0.03,0.05,0.1]**
- **sample_rate = [0.2,…,1]**
- **col_sample_rate = [0.2,…,1]**
- **min_rows = [1,2,…,20]**
- **nbins_cats = [10, 100, 1000, 10000]**
- **histogram_type = ["UniformAdaptive", "QuantilesGlobal"]**
- **stopping_rounds = 3** (together with **ntrees=5000** or so)

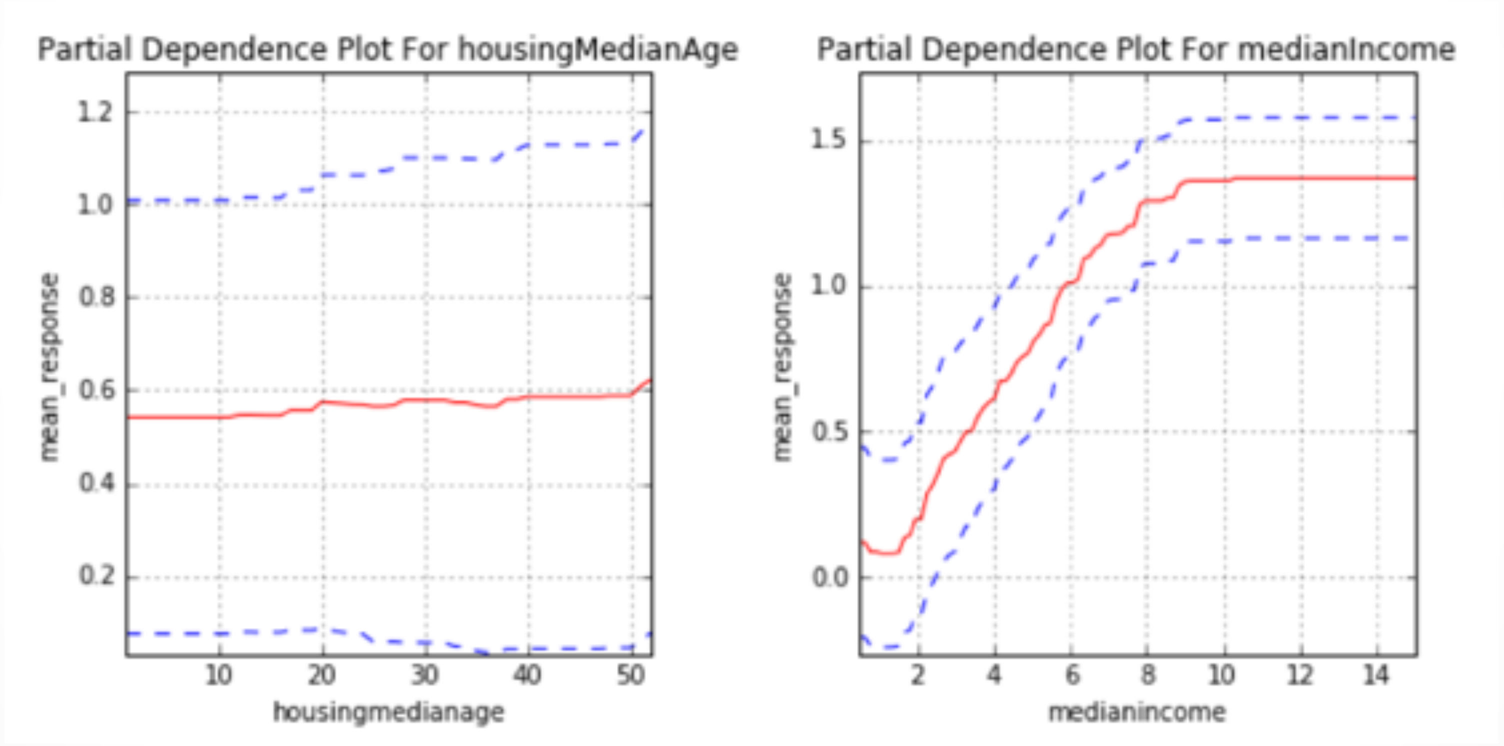
Just need to find one of the many good models

H2O.ai

Partial Dependence Plots

Partial dependence plots display the mean prediction for a given model and a given value of a dependent variable, over the range of the dependent variable.

Partial dependence plots now available in R, Python, and Flow.



Partial dependence plots for the well-known California housing data set.

Trust and Understanding

Questions?