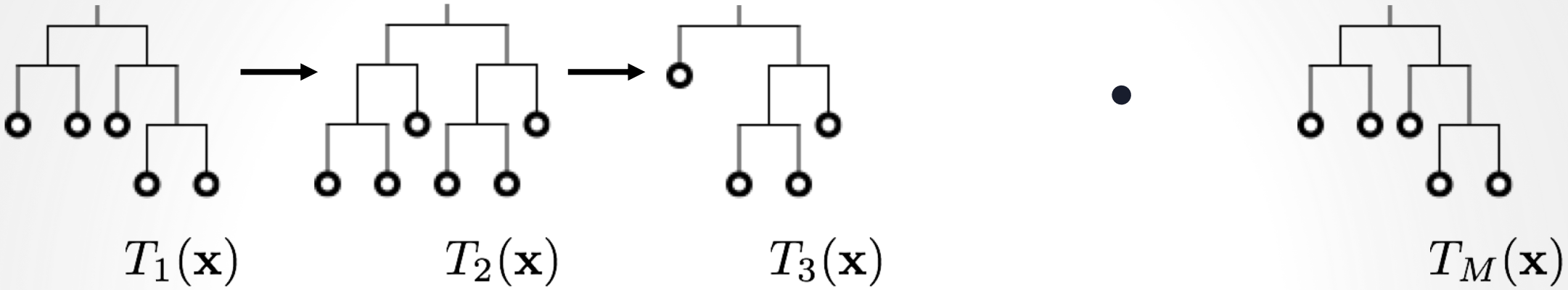
Supervised Learning:

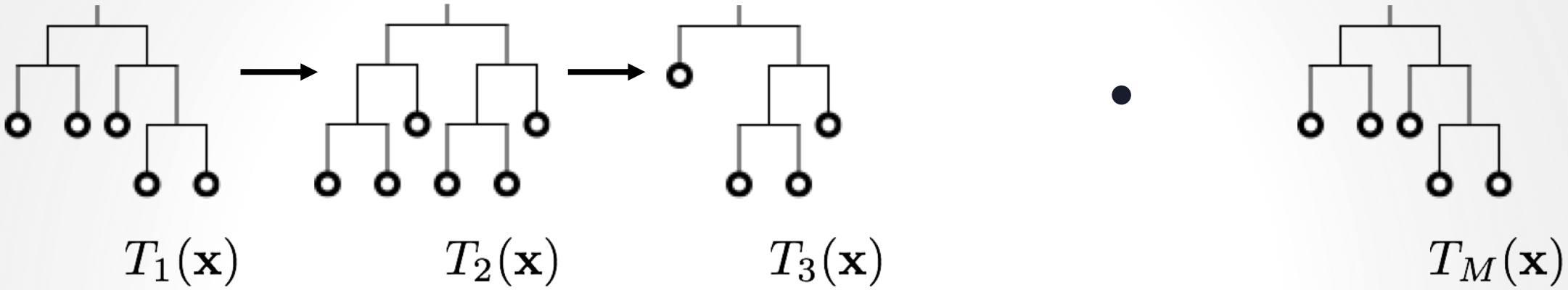
GRADIENT BOOSTING MACHINES

Gradient Boosting Machine (GBM)



$$f_i(\mathbf{x}) = f_{i-1}(\mathbf{x}) + T_i(\mathbf{x}; \hat{\Theta}_i)$$

Gradient Boosting Machine (GBM)



Algorithm 1 Forward Stagewise Additive Training

1: Initialize $f_0(\mathbf{x}) = 0$.
2: **for** $i = 1$ to M **do**
3: $\quad \hat{\Theta}_i = \arg\min_{\Theta_i} \sum_{i=1}^{n} L(y_i, f_{i-1}(x_i) + T(x_i; \Theta_i))$
4: $\quad f_i(\mathbf{x}) = f_{i-1} + T(\mathbf{x}; \hat{\Theta}_i)$
5: **end for**
6: $f(\mathbf{x}) = f_M(\mathbf{x})$

GBM Algorithm

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

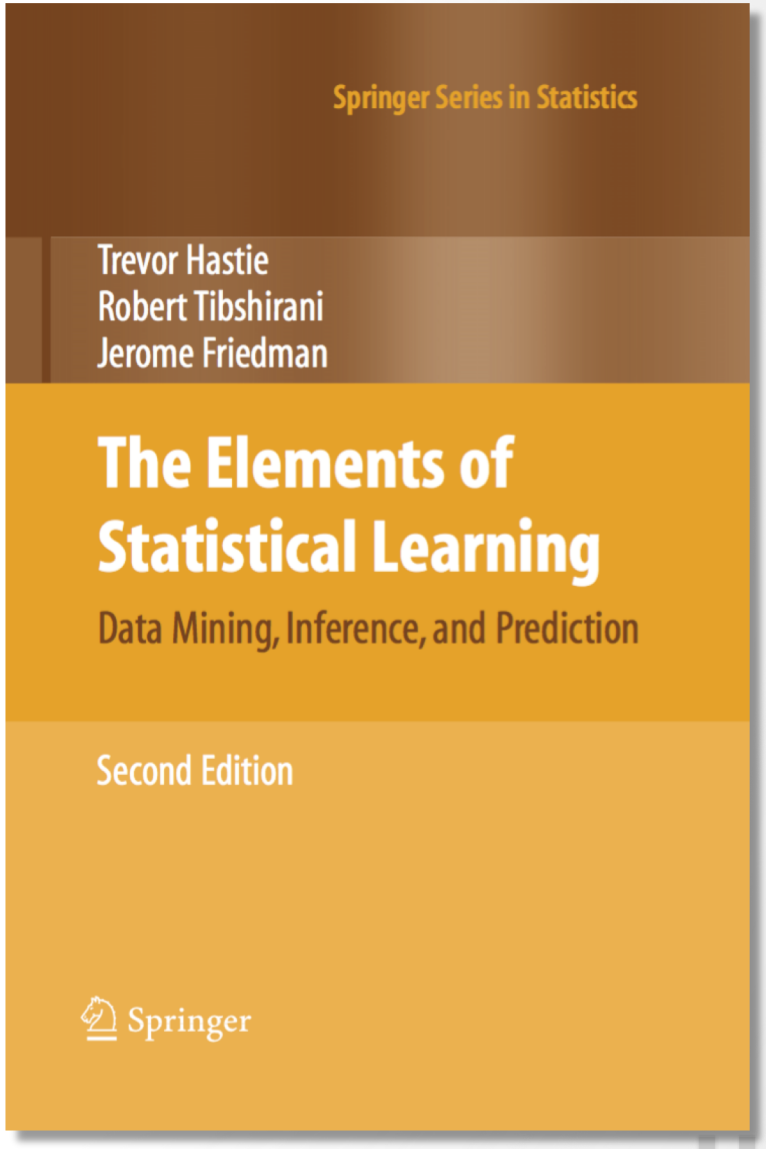
1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute

$$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.



Gradient Boosting Machine (GBM) Implementations

- H2O GBM
- XGBoost (in H2O)
  - LightGBM
  - DART

GBM Parameters: Setting Up a Model

Python API

```
h2o.estimators.gbm.H2OGradientBoostingEstimator
- h2o.estimators.estimator_base.H2OEstimator
-- h2o.model.model_base.ModelBase
```

`model_id:`	specify a custom name for the model to use as a reference. (default: randomly generated)
`checkpoint:`	load a previously generated model
`seed:`	specify a pseudorandom seed, for reproducibility
`verbose:`	print verbose scoring history to console

GBM Parameters: Defining Data

(same as RF)

- `training_frame:` H2OFrame used to train model
- `validation_frame:` H2OFrame used to validate model (optional)
- `y:` column name or index of target variable
- `x:` column names (of indices) of predictor variables (default: all but y and ignored_columns)

- `ignored_columns:` columns to ignore (e.g. ID column)
- `ignore_const_cols:` ignore constant columns (default: True)
- `weights_column:` column by which to weight the data points

- `categorical_encoding:` encoding scheme for categorical variables: {"enum", "one_hot_explicit", "binary", "eigen", "label_encoder", "sort_by_response"} (default: "enum")

GBM Parameters: Loss Function

(same as RF)

- **`distribution:`** specifies the loss function, for use in determining gradients
  - `'bernoulli'`
  - `'multinomial'`
  - `'gaussian'`
  - `'poisson'`
  - `'gamma'`
  - `'laplace'`
  - `'quantile'`
  - `'huber'`
  - `'tweedie'`

- **`huber_alpha, quantile_alpha`**

GBM Parameters: Ensemble

(same as RF)

- `ntrees:` number of trees

- `sample_rate:` row sampling rate per tree (default:1)
- `col_sample_rate_per_tree:` column sampling rate per tree (default: 1)

GBM Parameters: Individual Trees

- **Column sampling for split:**
- **col_sample_rate:** number of columns to sample on each split (default: sqrt(n) for classification, n/3 for regression)
- **col_sample_rate_change_per_level:** factor by which to increase or decrease mtries per level of tree

- **When to stop splitting?**
- **max_depth:** maximum depth of each tree
- **min_rows:** minimum rows in a leaf (i.e. stop splitting when data size is this small)
- **min_split_improvement:** **minimum relative improvement** in split criterion for a split to occur

- **Histogramming**
- **nbins**: number of bins for numeric variables (default: 20)
- **nbins_top_level**: can be used instead of nbins; nbins will then decrease by 2 each level
- **nbins_cats**: **histogram_type**: number of bins for categorical variables (default: 1024)

- **max_abs_leafnode_pred**

GBM Parameters: Boosting

- `learn_rate:` boosting factor, eta (default=0.1)
- `learn_rate_annealing:` factor by which to reduce the learn_rate every tree (default=1)

GBM Parameters: Class Imbalances

(same as RF)

- **`balance_classes`**: balance training data class counts via over/under-sampling (default: False)
- **`class_sampling_factors`**: desired over/under-sampling ratios per class (in lexicographic order). If not specified, sampling fac- tors will be automatically
- **`max_after_balance_size`**: maximum relative size of the training data after balancing class counts (can be less than 1.0). Requires balance classes. Defaults to 5.0.
- **`sample_rate_per_class:`** variable row sampling rate per class

GBM Parameters: Scoring & Stopping

(same as RF)

- **score_each_iteration**: score model after each tree (default: false)
- **score_tree_interval**: score model after n trees

- **stopping_rounds**: early stop if stopping metric's moving average does not improve for this many rounds
- **stopping_metric**: metric for early stopping (AUTO: logloss for classification, deviance for regression) Must be one of: "AUTO", "deviance", "logloss", "MSE", "RMSE", "MAE", "RMSLE", "AUC", "lift_top_group", "misclassification", "mean_per_class_error". Defaults to AUTO.
- **stopping_tolerance**: Relative tolerance for metric-based stopping criterion (stop if relative improvement is not at least this much) Defaults to 0.001.
- **max_runtime_secs**: maximum runtime to allow for model building (default: 0, disabled)

GBM Parameters: Cross Validation

(same as RF)

- **nfolds**: number of folds for N-fold cross-validation (default: 0, disabled)
- **keep_cross_validation_predictions**: keep the predictions of the cross-validation models (default: False)
- **keep_cross_validation_fold_assignment**: keep the cross-validation fold assignment (default: False)
- **fold_assignment**: cross-validation fold assignment scheme, if fold column is not specified. The "Stratified" option will stratify the folds based on the response variable, for classification problems. Must be one of: "AUTO", "Random", "Modulo", "Stratified". Defaults to AUTO.
- **fold_column**: column with cross-validation fold index assignment per observation.

GBM Parameters: Misc.

- **offset_column**
- **build_tree_one_node**
- **calibrate_model**
- **calibrate_frame**
- **pred_noise_bandwith**
- **max_hit_ratio_k**: Max. number (top K) of predictions to use for hit ratio computation (for multiclass only, 0 to disable) Defaults to 0.

H2O GBM Tuning Tutorial for Python

https://github.com/h2oai/h2o-3/blob/master/h2o-docs/src/product/tutorials/gbm/gbmTuning.ipynb

H2O GBM Tuning Tutorial for Python

Arno Candel, PhD, Chief Architect, H2O.ai

Ported to Python by Navdeep Gill, M.S., Hacker/Data Scientist, H2O.ai

In this tutorial, we show how to build a well-tuned H2O GBM model for a supervised classification task. We specifically don't focus on feature engineering and use a small dataset to allow you to reproduce these results in a few minutes on a laptop. This script can be directly transferred to datasets that are hundreds of GBs large and H2O clusters with dozens of compute nodes.

You can download the source from H2O's github repository.

Ports to R Markdown and Flow UI (now part of Example Flows) are available as well.

XGBoost

- Popular open-source GBM library
- Used by many Kagglers and a successful algorithm implementation to win competitions

XGBoost Framework

- DMatrix: Data set abstraction and data structure
- Booster: GBM algorithm abstraction

XGBoost in H2O

- No algorithmic difference between H2O XGBoost and “regular” XGBoost
  - H2O just calls the regular XGBoost backend
  - H2O uses JNI to communicate to native C++ XGBoost libraries

- Two modules
  - **`h2o-ext-xgboost`**: contains actual XGBoost model and model builder code
  - **`h2o-genmodel-ext-xgboost`**: extends **`h2o-genmodel`** module and registers XGBoost-specific MOJO

Current XGBoost Support

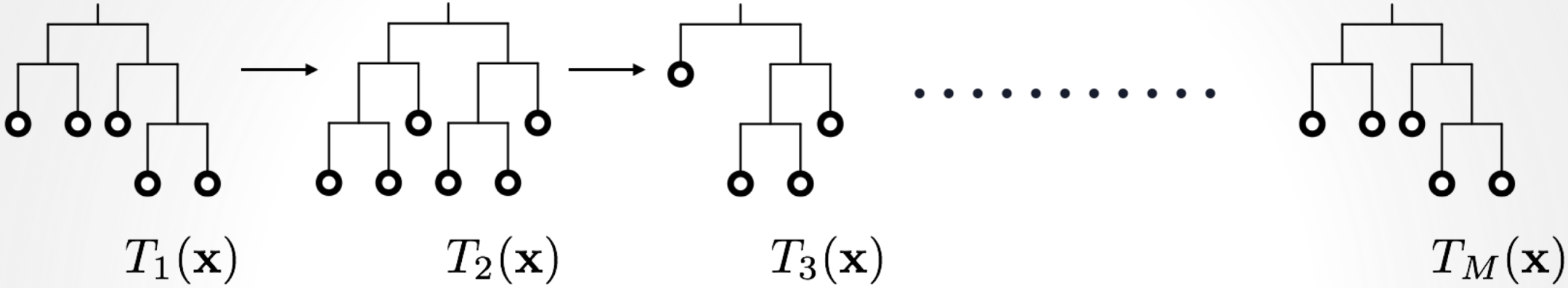
1. XGBoost is not supported on Windows.
2. XGBoost is initialized for single-node H2O clusters; however multi-node XGBoost support is available as a Beta feature.
3. The list of supported platforms includes:

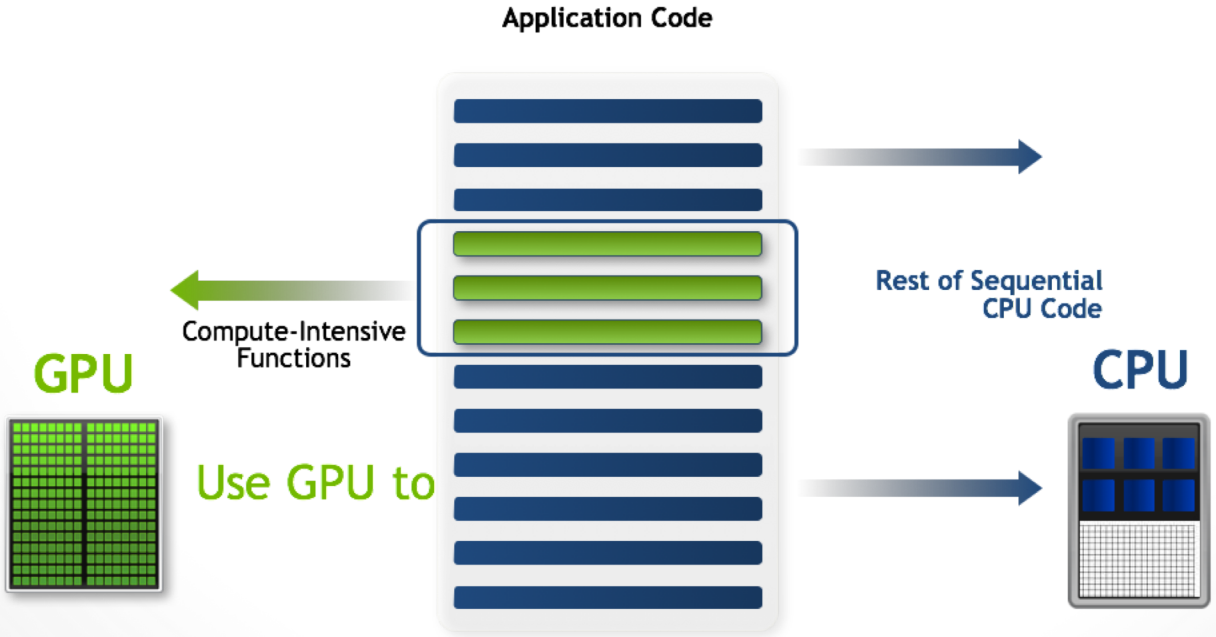
Platform	Minimal XGBoost	OMP	GPU	Compilation OS
Linux	yes	yes	yes	Ubuntu 14.04, g++ 4.7
OS X	yes	no	no	OS X 10.11
Windows	no	no	no	NA

Note: Minimal XGBoost configuration includes support for a single CPU.

4. Because we are using native XGBoost libraries that depend on OS/platform libraries, it is possible that on older operating systems, XGBoost will not be able to find all necessary binary dependencies, and will not be initialized and available.
5. XGBoost GPU libraries are compiled against CUDA 8, which is a necessary runtime requirement in order to utilize XGBoost GPU support.

GBM on GPU





Pros and Cons of Gradient Boosting Machines

Pros

- Robust to correlated features
- Robust to missing values
- Handles non-linearities in data
- Handles interaction effects

Cons

- Requires care in tuning params
  - Tends to overfit the training set
- Sensitive to noise and outliers