Mathematical techniques in data science

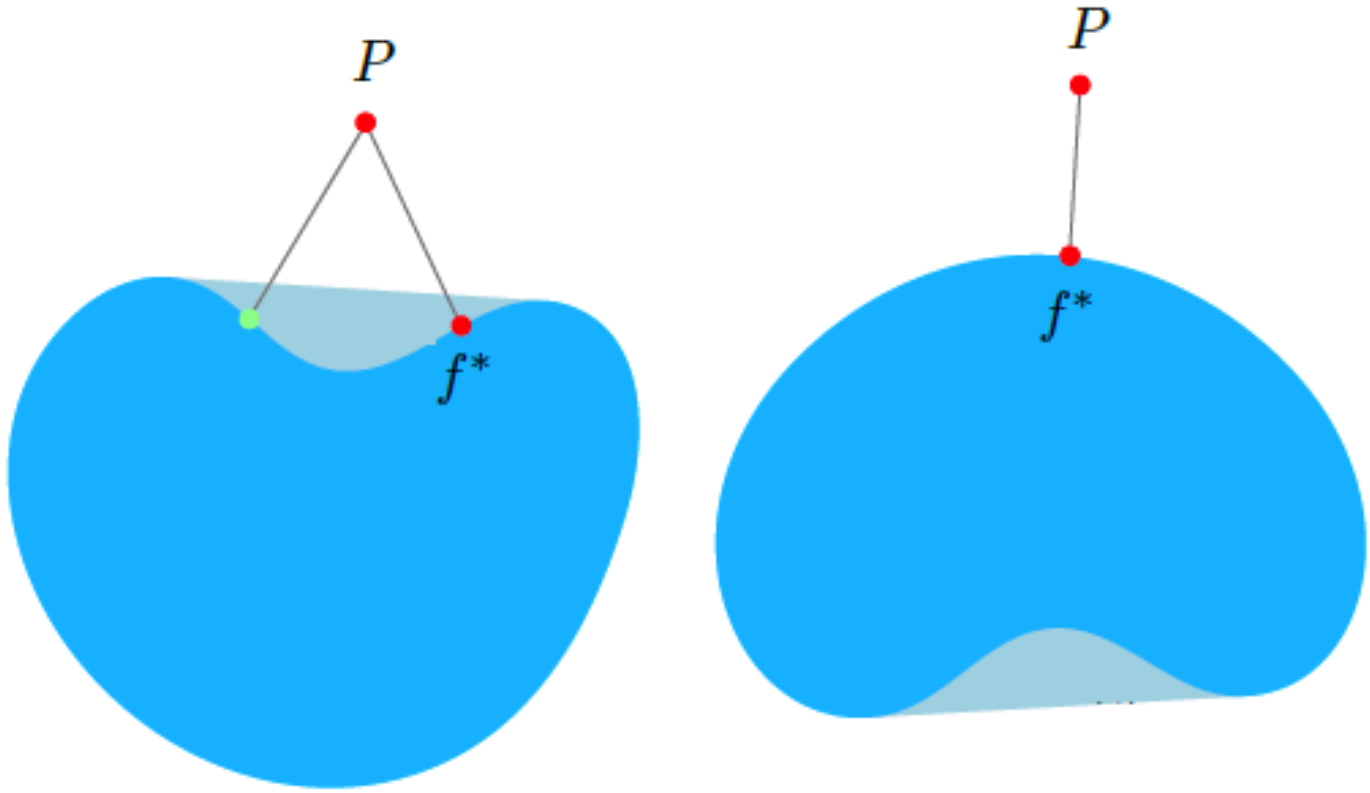
Lecture 32: Boosting

Boosting

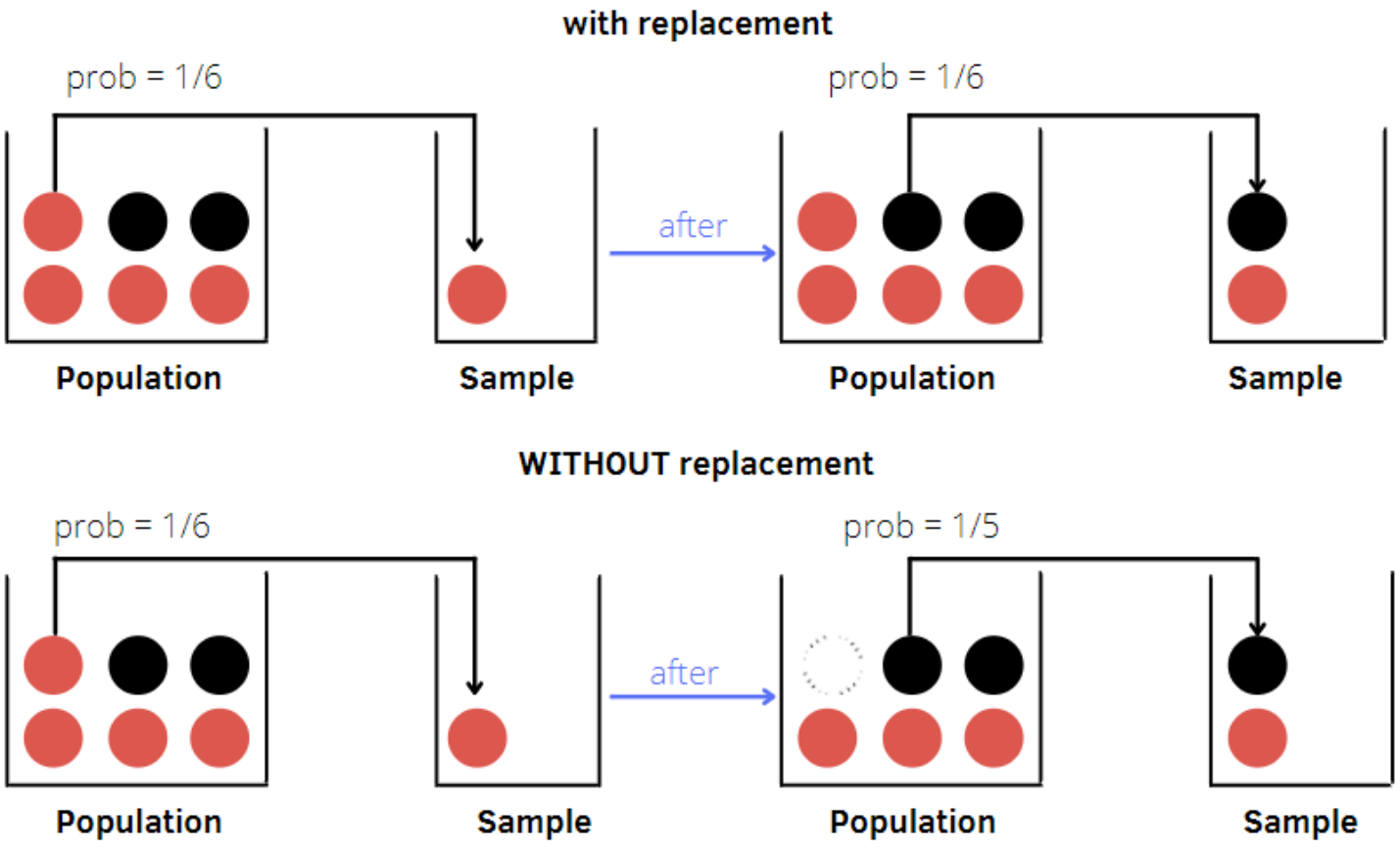
- Decision trees
- Bagging
- Random forests
- Boosting

Main idea: we can combine weak learners into a single strong learner

Convexification of the hypothesis space



Sampling with replacement



Bootstrap

Bootstrapping: General statistical method that relies on resampling data with replacement.

Idea: Given data (y_i, x_i), $i = 1, \ldots, n$, construct *bootstrap samples* by sampling n of the observations **with replacement** (i.e., allow repetitions):

Sample 1	Sample 2	Sample 3
(y_{i_1}, x_{i_1})	(y_{j_1}, x_{j_1})	(y_{k_1}, x_{k_1})
(y_{i_2}, x_{i_2})	(y_{j_2}, x_{j_2})	(y_{k_2}, x_{k_2})
$\vdots$	$\vdots$	$\vdots$
(y_{i_n}, x_{i_n})	(y_{j_n}, x_{j_n})	(y_{k_n}, x_{k_n})

Bagging

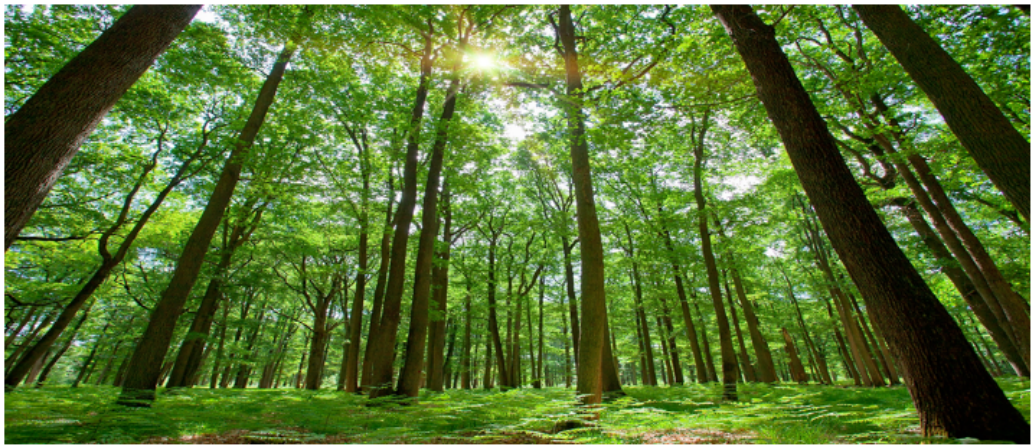
Bagging:(bootstrap aggregation) Suppose we have a model $y \approx \hat{f}(x)$ for data $(y_i, x_i) \in \mathbb{R}^{p+1}$.

1. Construct $B \in \mathbb{N}$ bootstrap samples.
2. Train the method on the b-th bootstrap sample to get $\hat{f}^{*b}(x)$.
3. Compute the average of the estimators:

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{f}^{*b}(x).$$

- Bagging is often used with regression trees.
- Can improve estimators significantly.

Random forests



Random forests: Each time a split in a tree is considered, a random selection of m predictors is chosen as split candidates from the full set of p predictors.

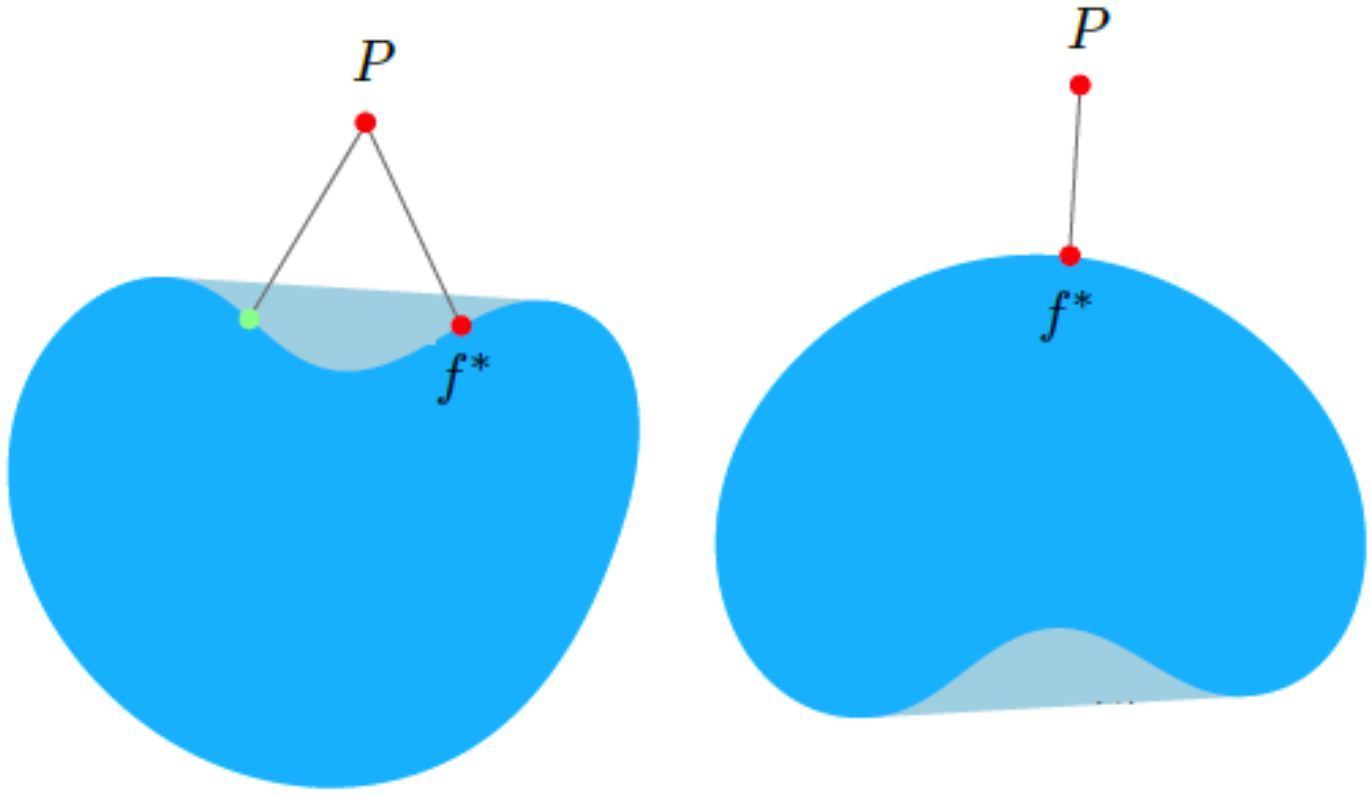
- Typical value for m is $\sqrt{p}$.
- We construct $T_1, \ldots, T_B$ trees using that method on bootstrap samples. The **random forest (regression) predictor** is

$$\hat{f}_{\text{rf}}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x).$$

For classification: use majority vote.

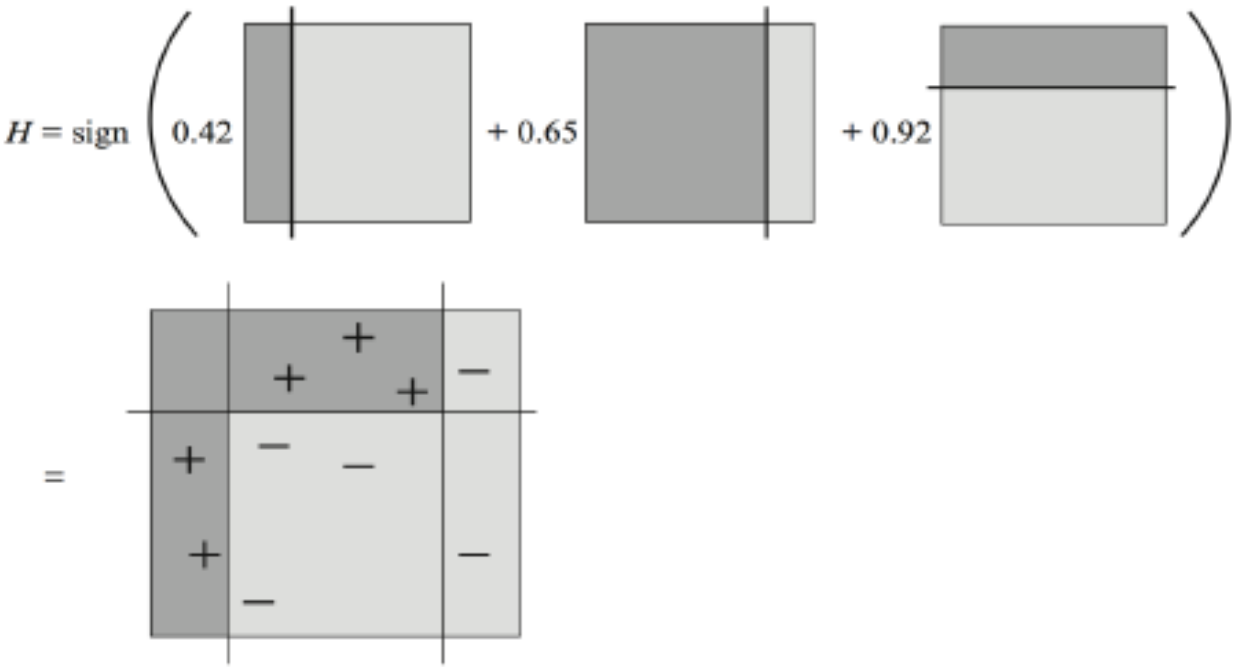
Adaboost

Non-convexity of the hypothesis space



Moving out of the hypothesis space

$$H(x) = \sum_t \rho_t h_t(x)$$

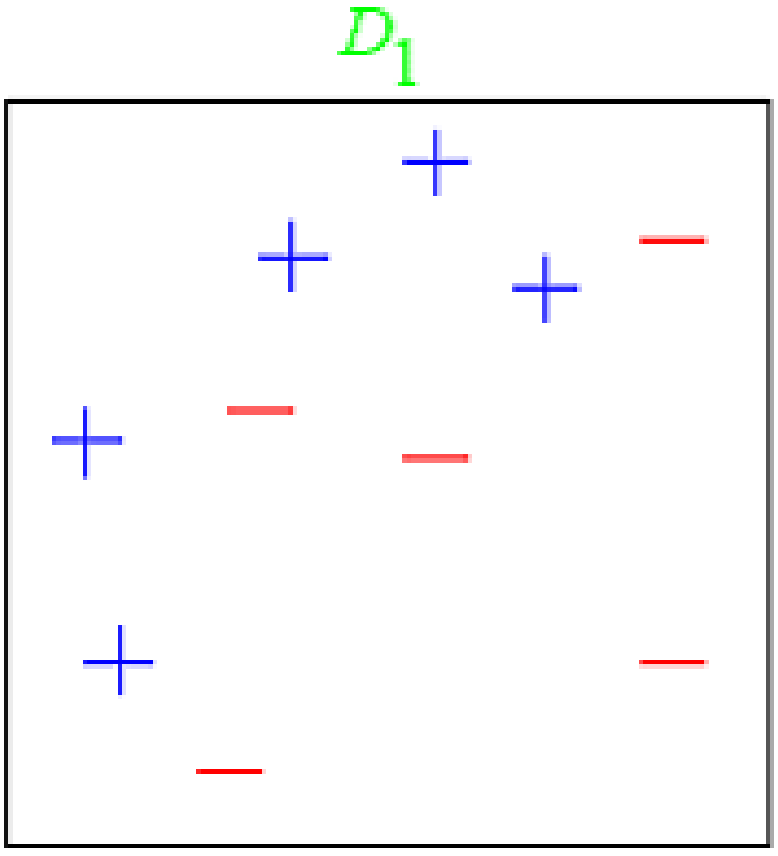


Adaboost

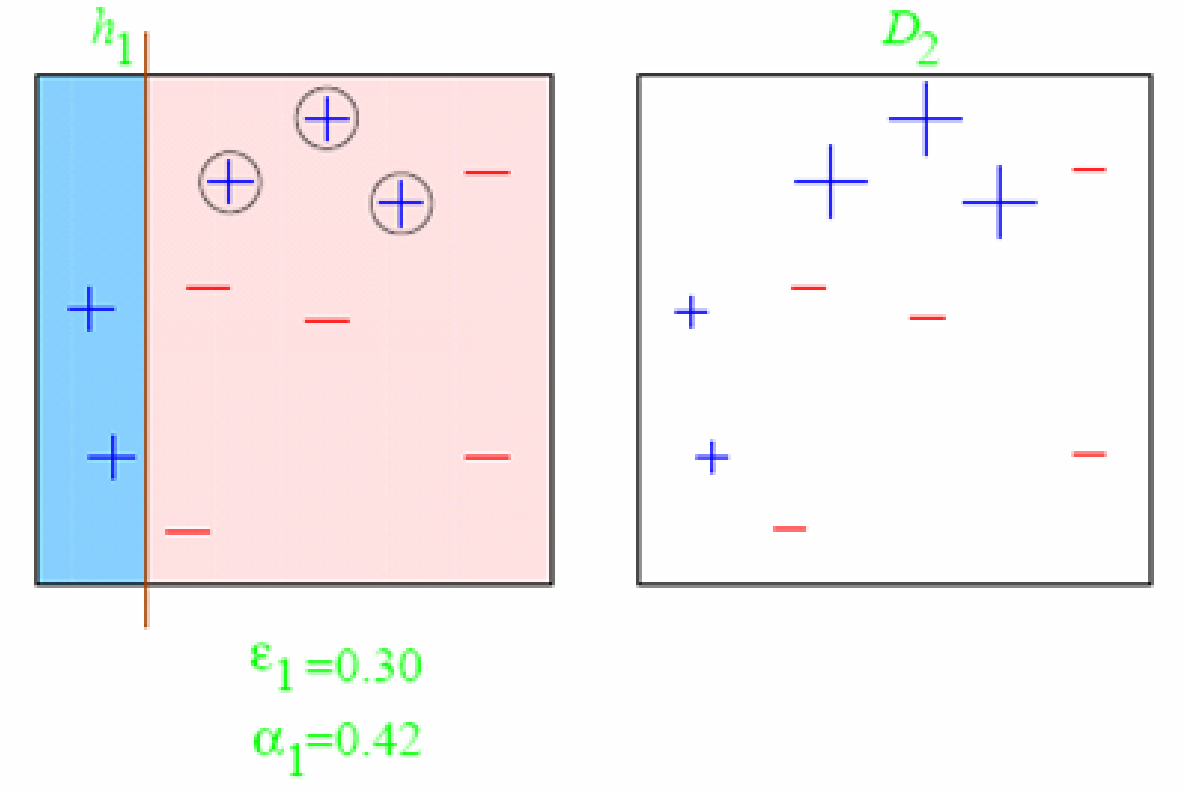
Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

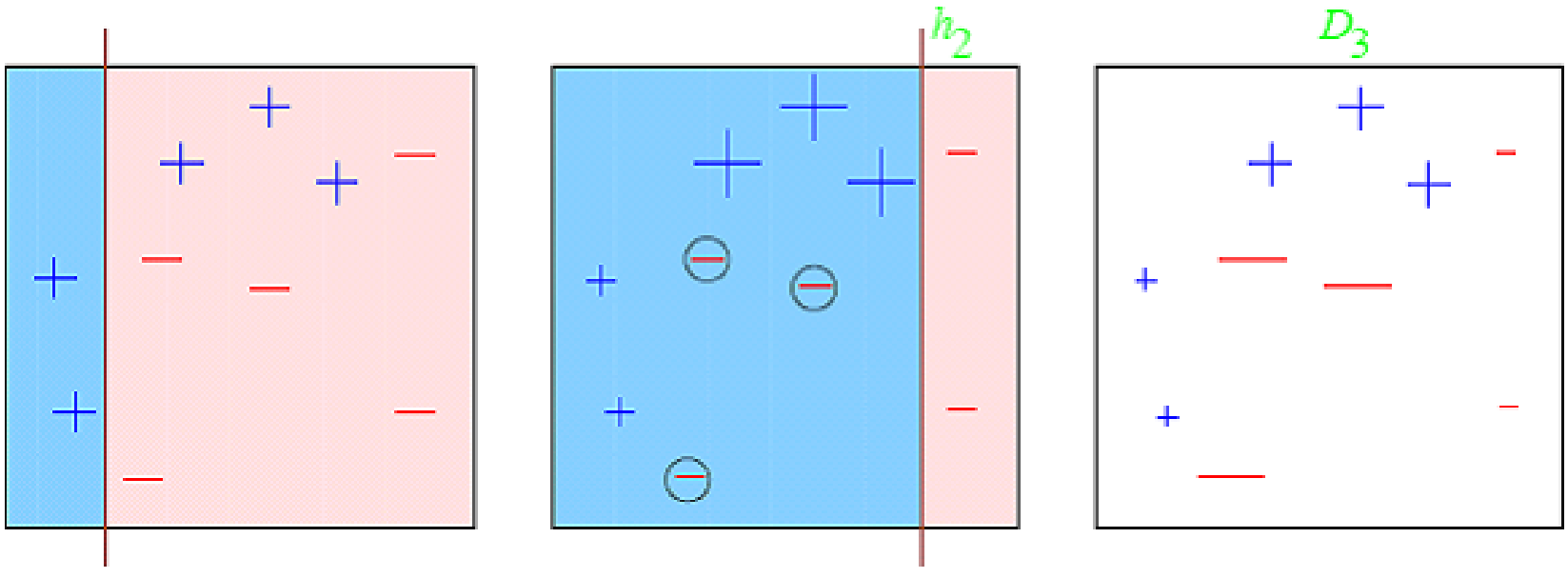
Adaboost



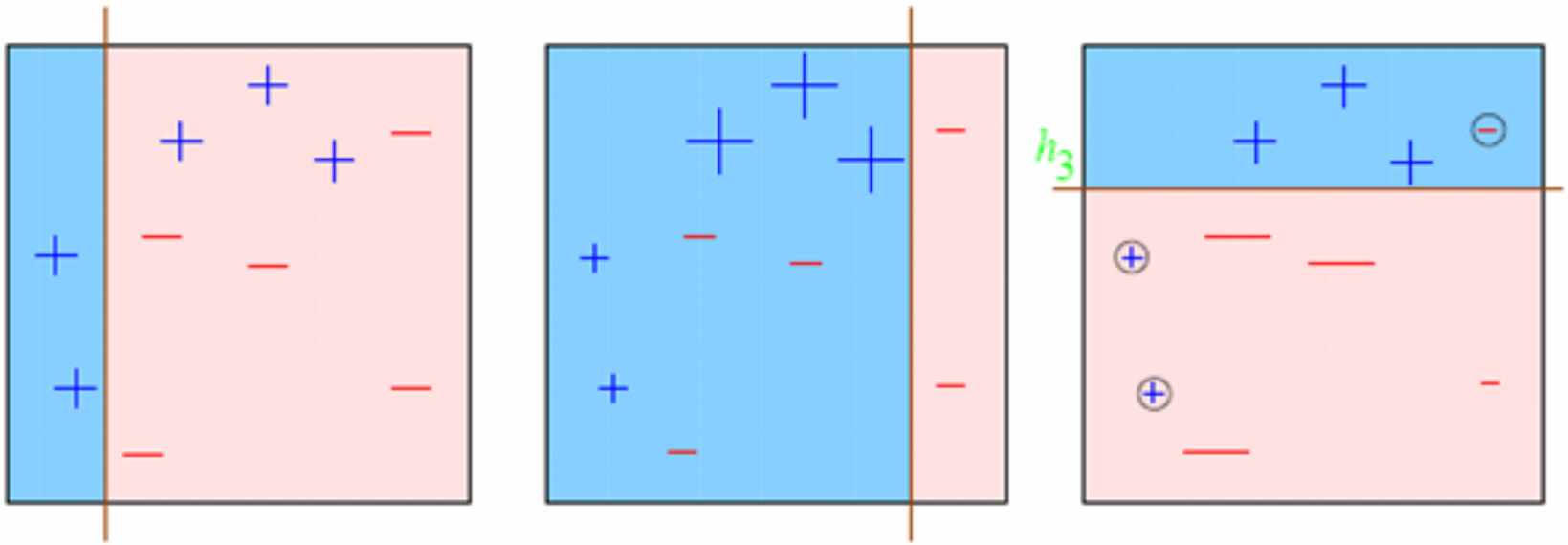
Adaboost



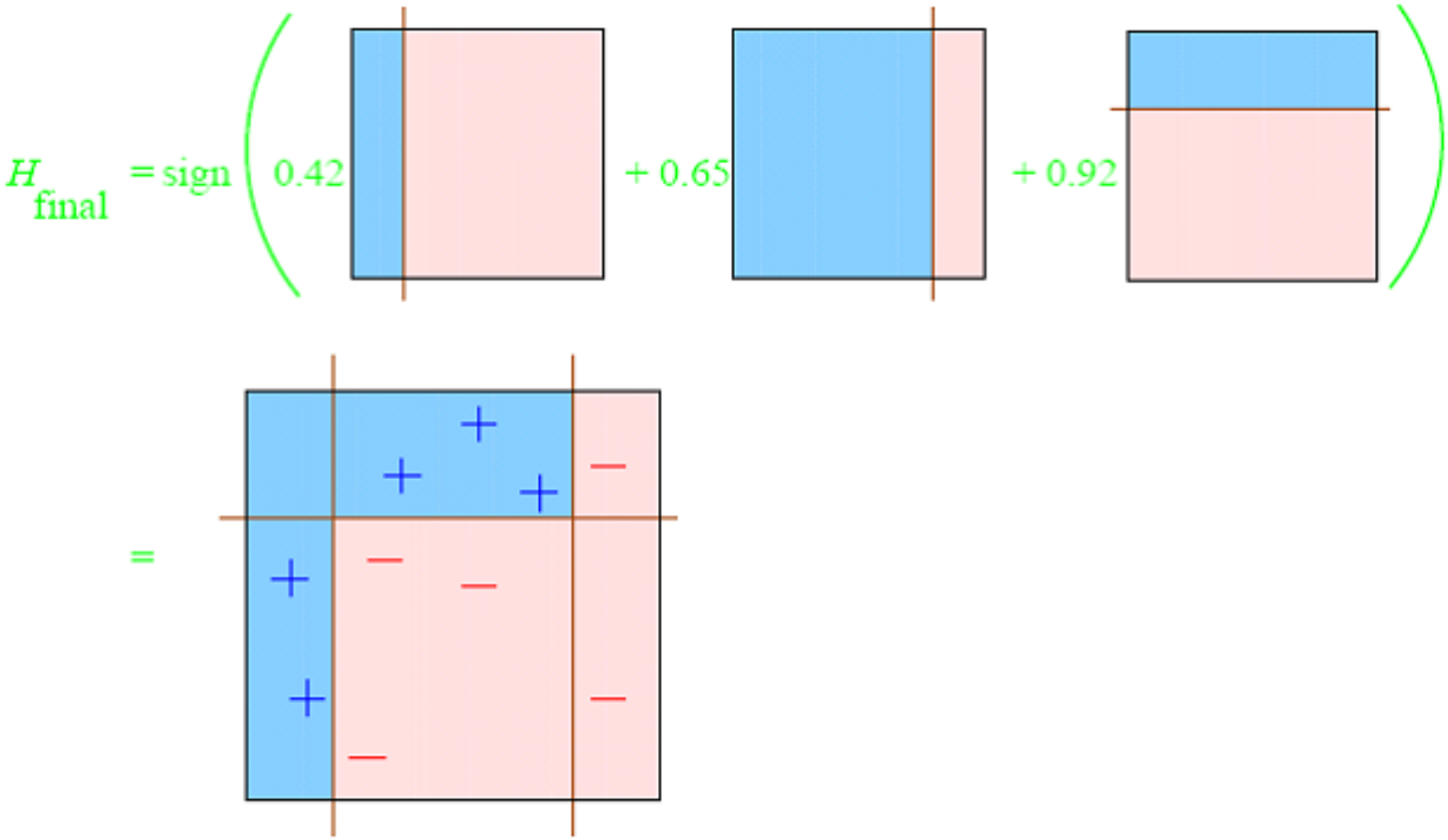
Adaboost



Adaboost



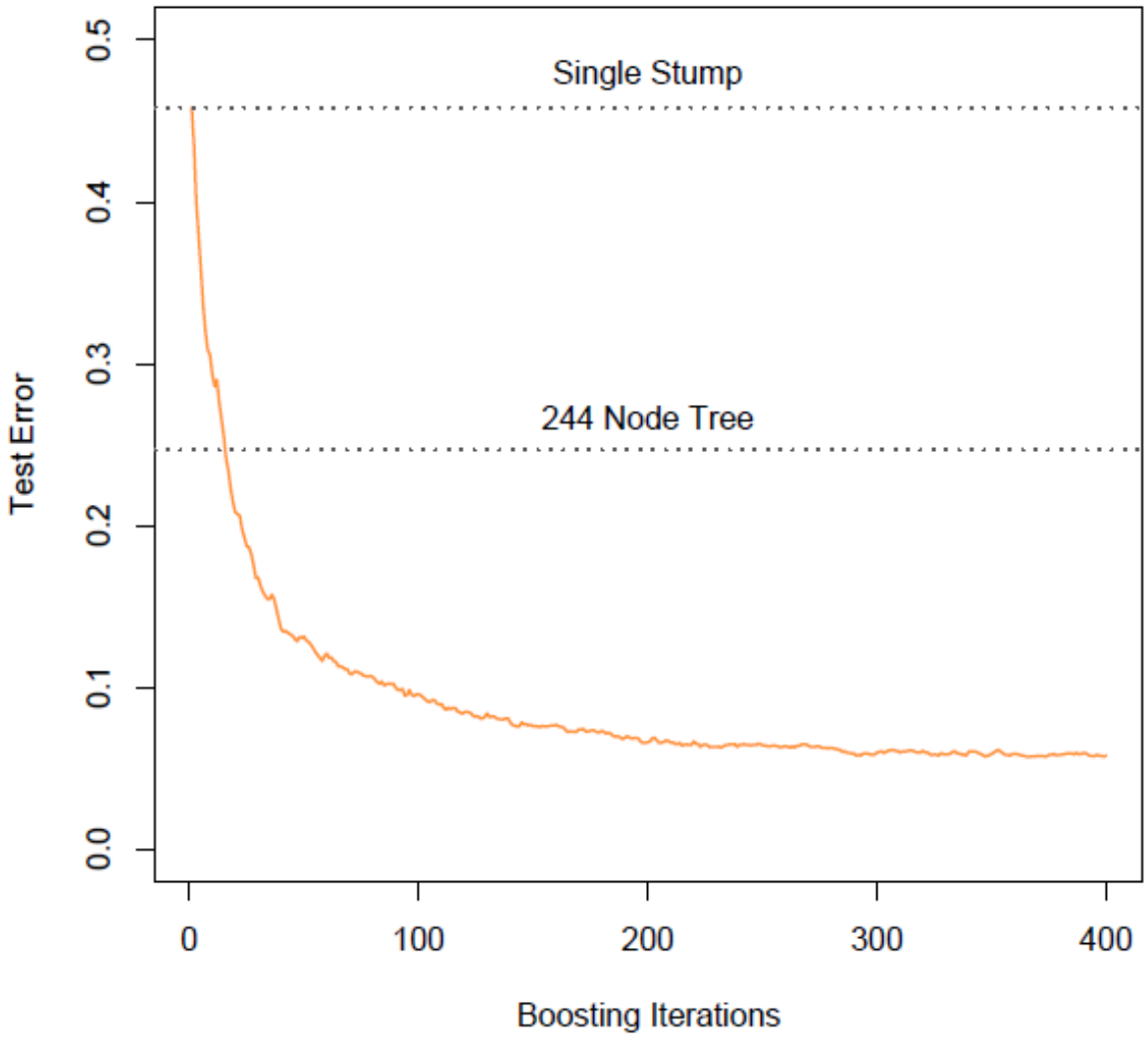
Adaboost



Gradient boosting: history

- Invent Adaboost, the first successful boosting algorithm [Freund et al., 1996, Freund and Schapire, 1997]
- Formulate Adaboost as gradient descent with a special loss function[Breiman et al., 1998, Breiman, 1999]
- Generalize Adaboost to Gradient Boosting in order to handle a variety of loss functions [Friedman et al., 2000, Friedman, 2001]

Gradient boosting



Gradient descent

Gradient Descent

Minimize a function by moving in the opposite direction of the gradient.

$$\theta_i := \theta_i - \rho \frac{\partial J}{\partial \theta_i}$$

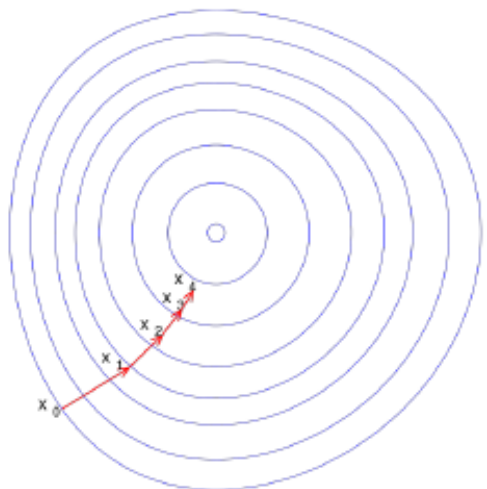
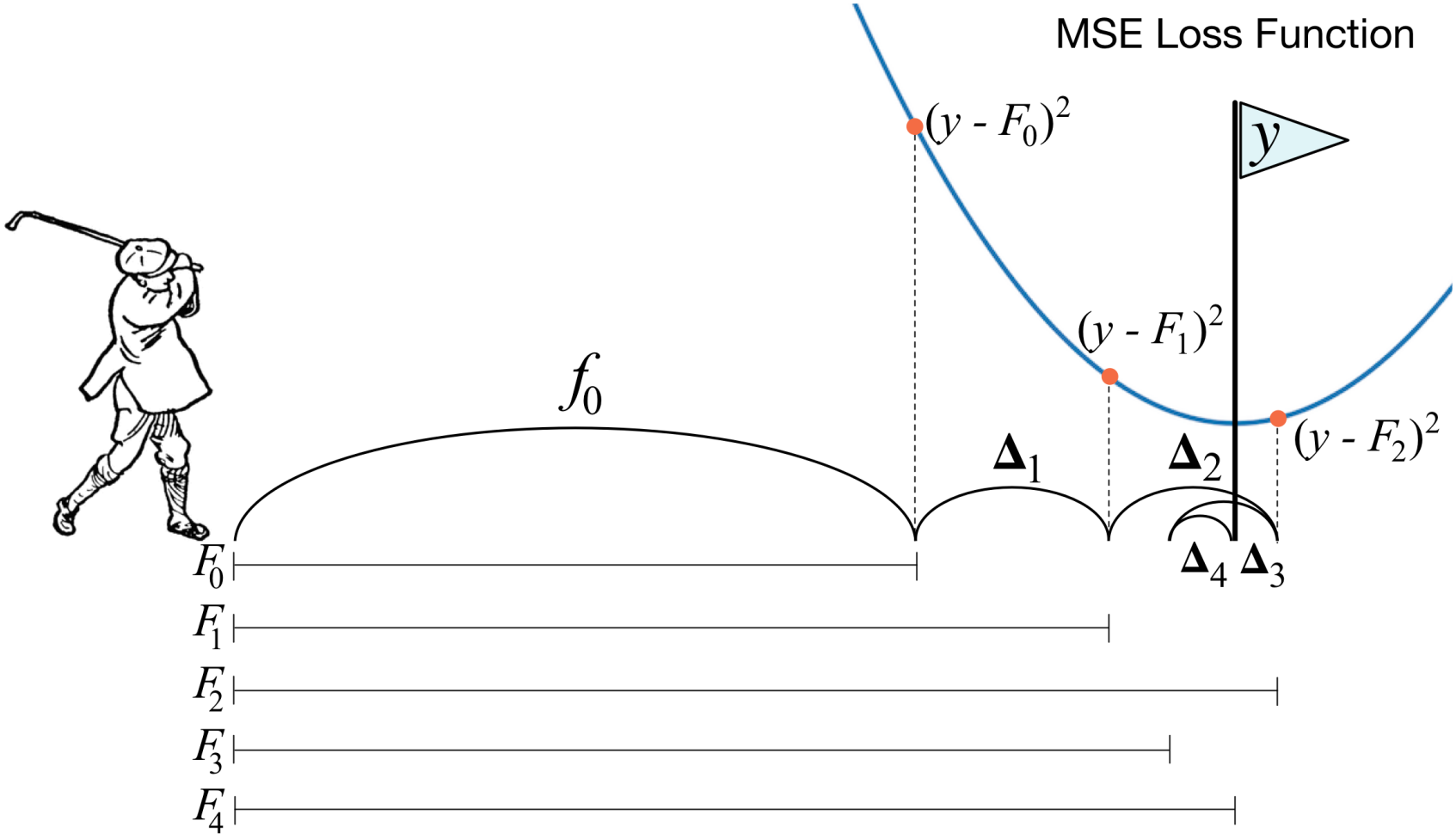


Figure: Gradient Descent. Source: http://en.wikipedia.org/wiki/Gradient_descent

Gradient boosting



Gradient boosting

Boosting: Recursively fit trees to residuals. (Compensate the shortcoming of previous model.)
Input: $(y_i, x_i) \in \mathbb{R}^{p+1}$, $i = 1, \ldots, n$. Initialize $\hat{f}(x) = 0$, $r_i = y_i$.
For $b = 1, \ldots, B$:

1. Fit a tree estimator $\hat{f}^b$ with d splits to the training data.
2. Update the estimator using:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \cdot \hat{f}^b(x).$$

3. Update the residuals:

$$r_i \leftarrow r_i - \lambda \cdot \hat{f}^b(x_i).$$

Output: Boosted tree:

$$\hat{f}(x) = \sum_{i=1}^{B} \lambda \hat{f}^b(x).$$

Gradient boosting

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

`sklearn.ensemble` : Ensemble Methods

The `sklearn.ensemble` module includes ensemble-based methods for classification, regression and anomaly detection.

User guide: See the Ensemble methods section for further details.

`ensemble.AdaBoostClassifier` ([…])	An AdaBoost classifier.
`ensemble.AdaBoostRegressor` ([base_estimator, …])	An AdaBoost regressor.
`ensemble.BaggingClassifier` ([base_estimator, …])	A Bagging classifier.
`ensemble.BaggingRegressor` ([base_estimator, …])	A Bagging regressor.
`ensemble.ExtraTreesClassifier` ([…])	An extra-trees classifier.
`ensemble.ExtraTreesRegressor` ([n_estimators, …])	An extra-trees regressor.
`ensemble.GradientBoostingClassifier` ([loss, …])	Gradient Boosting for classification.
`ensemble.GradientBoostingRegressor` ([loss, …])	Gradient Boosting for regression.
`ensemble.IsolationForest` ([n_estimators, …])	Isolation Forest Algorithm
`ensemble.RandomForestClassifier` ([…])	A random forest classifier.
`ensemble.RandomForestRegressor` ([…])	A random forest regressor.
`ensemble.RandomTreesEmbedding` ([…])	An ensemble of totally random trees.
`ensemble.VotingClassifier` (estimators[, …])	Soft Voting/Majority Rule classifier for unfitted estimators.

Gradient boosting

dmlc
XGBoost eXtreme Gradient Boosting

Community | Documentation | Resources | Contributors | Release Notes

XGBoost is an optimized distributed gradient boosting library designed to be highly ***efficient***, ***flexible*** and ***portable***. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

License