

Boosting

...

Boosting Overview

- Like bagging, going to draw a sample of the observations from our data with replacement
- Unlike bagging, the observations not sampled randomly
- Boosting assigns a weight to each training observation and uses that weight as a sampling distribution
 - Higher weight observations more likely to be chosen.
- May adaptively change that weight in each round
- **The weight is higher for examples that are harder to classify**

Boosting Example

input variable

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	1	1	1	-1	-1	-1	-1	1	1	1

target

- Same dataset used to illustrate bagging
- Boosting typically requires fewer rounds of sampling and classifier training.
- Start with equal weights for each observation
- Update weights each round based on the classification errors

Boosting Example

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

(a) Training records chosen during boosting

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

(b) Weights of training records

Boosting: Weighted Ensemble

- Unlike Bagging, Boosted Ensembles usually weight the votes of each classifier by a function of their accuracy.
- If a classifier gets the higher weight observations wrong, it has a higher error rate.
- More accurate classifiers get higher weight in the prediction.

Boosting: Classifier weights

Errors made: First 3 observations

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Errors made: Middle 4 observations

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Errors made: Last 3 observations

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

(a) Training records chosen during boosting

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

(b) Weights of training records

Boosting: Classifier weights

Errors made: First 3 observations

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Errors made: Middle 4 observations

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Errors made: Last 3 observations

x	0.2	0.2	0.4	0.4	0.4	0.4				
y	1	1	-1	-1	-1	-1				

Lowest weighted error.
Highest weighted model.

(a) Training records chosen during boosting

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

(b) Weights of training records

Boosting: Weighted Ensemble

Round	Split Point	Left Class	Right Class	Weight
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

Classifier Decision Rules and Classifier Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Individual Classifier Predictions and Weighted Ensemble Predictions

Boosting: Weighted Ensemble

Round	Split Point	Left Class	Right Class	Weight
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

Classifier Decision Rules and Classifier Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

5.16 = -1.738 + 2.7784 + 4.1195

Individual Classifier Predictions and Weighted Ensemble Predictions

(Major) Boosting Algorithms

AdaBoost (This is sooo 2007)

Gradient Boosting [xgboost]
(Welcome to the New Age of learning)

(Self-Study)

AdaBoost Details: The Classifier Weights

- Let w_j be the weight of observation j entering into present round.
- Let $m_j = 1$ if observation j is misclassified, 0 otherwise
- The error of the classifier this round is

$$\epsilon_i = \frac{1}{N} \sum_{j=1}^N w_j m_j$$

- The voting weight for the classifier this round is then

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \epsilon_i}{\epsilon_i} \right)$$

(Self-Study)

AdaBoost Details: Updating observation Weights

To update the observation weights from the current round (round i) to the next round (round $i + 1$):

$$w_j^{(i+1)} = w_j^i e^{-\alpha_j} \quad \text{if observation } j \text{ was correctly classified}$$

$$w_j^{(i+1)} = w_j^i e^{\alpha_j} \quad \text{if observation } j \text{ was misclassified}$$

The new weights are then normalized to sum to 1 so they form a probability distribution.

Gradient Boosting



The latest and greatest
(Jerome H. Friedman 1999)



Gradient Boosting Overview

- Build a simple model $f_1(x)$ trying to predict a target y
- It has error, right?

$$y = f_1(x) + \epsilon_1$$

actual value modeled value error

- Now, let's try to predict that error with another simple model, $f_2(x)$. Unfortunately, it still has some error:

$$y = f_1(x) + f_2(x) + \epsilon_2$$

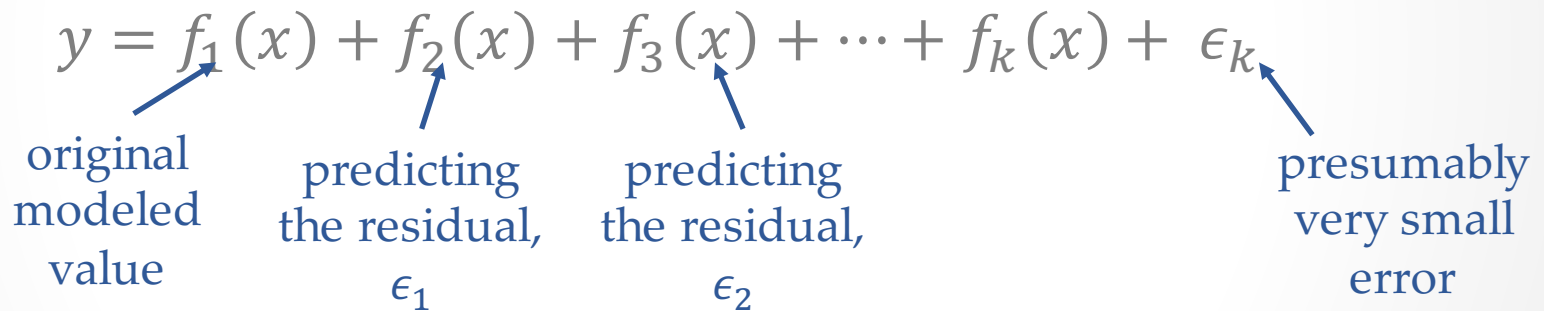
original modeled value predicting the residual, ϵ_1 error

Gradient Boosting Overview

- We could just continue to add model after model, trying to predict the residuals from the previous set of models.

$$y = f_1(x) + f_2(x) + f_3(x) + \cdots + f_k(x) + \epsilon_k$$

original modeled value predicting the residual, ϵ_1 predicting the residual, ϵ_2 presumably very small error



Gradient Boosting Overview

- To address the obvious problem of overfitting, we'll dampen the effect of the additional models by only taking a “step” toward the solution in that direction.
- We'll also start (in continuous problems) with a constant function (intercept)
- The **step-sizes** are automatically determined at each round inside the method

$$y = \gamma_1 + \gamma_2 f_2(x) + \gamma_3 f_3(x) + \cdots + \gamma_k f_k(x) + \epsilon_k$$

Gradient Boosted Trees

- Gradient boosting yields a additive ensemble model
- The key to gradient boosting is using “**weak learners**”
 - Typically simple, shallow decision/regression trees
 - Computationally fast and efficient
 - Alone, make poor predictions but ensembled in this additive fashion provide superior results

Gradient Boosting and Overfitting

- In general, the "step-size" is not enough to prevent us from overfitting the training data
- To further aid in this mission, we must use some form of **regularization** to prevent overfitting:
 1. Control the number of trees/classifiers used in the prediction
 - Larger number of trees => More prone to overfitting
 - Choose a number of trees by observing out-of-sample error
 2. Use a *shrinkage parameter* ("learning rate") to effectively lessen the step-size taken at each step. Often called eta, η
 - $y = \gamma_1 + \eta \gamma_2 f_2(x) + \eta \gamma_3 f_3(x) + \dots + \eta \gamma_k f_k(x) + \epsilon_k$
 - Smaller values of eta => Less prone to overfitting
 - eta = 1 => no regularization

Gradient Boosting Summary

➤ Advantages

- Exceptional model – one of most accurate available, generally superior to Random Forests when well trained
- Can provide information on variable importance for the purposes of variable selection

➤ Disadvantages

- Model lacks interpretability in the classical sense aside from variable importance
- The trees must be trained sequentially so computationally this method is slower than Random Forest
- Extra tuning parameter over Random Forests, the regularization or shrinkage parameter, etc.

The RGF algorithm is a variation of GBDT in which the structure search and the optimization are decoupled. More specifically, the main differences are given as follows:

- RGF introduces an explicit regularization term that takes advantage of individual tree structures.

$$\hat{h} = \operatorname{argmin}_{h \in H} [\ell(h(\mathbf{x}); y) + R(h)] \quad (4)$$

- RGF employs a *fully-corrective* greedy algorithm which iteratively modifies the weights of all the leaf nodes (decision rules) currently obtained while new rules are added into the forest by greedy search. Here, an explicit regularization is also included to avoid overfitting and very large models.
- RGF utilizes the concept of structured sparsity to perform greedy search directly over the forest nodes based on the forest structure.

Algorithm 2 Regularized Greedy Forest framework

$F \leftarrow \{\}$

while stopping criterion not met **do**

 Fix weights and adjust forest structure s :

$\hat{s} \leftarrow \operatorname{argmin}_{s \in S(F)} Q(s(F))$ (the optimum s that minimizes $Q(F)$ among all the structures that can be obtained by applying one structure-changing operation to F).

if some criterion is met **then**

 Fix the structure and change the weights in F s.t. the loss is minimized in $Q(F)$ (it can be optimized using a standard procedure (such as coordinate descent) if the regularization penalty is standard e.g., *L2-loss*)

end if

end while

Optimize leaf weights in F to minimize loss in $Q(F)$

return $h_F(\mathbf{x})$

Notes about EM

- EM has node for Random Forest (HP tab=> HP Forest)
 - Uses CHAID unlike other implementations
 - Does not perform bootstrap sampling
 - Does not appear to work as well as the randomForest package in R
- EM has node for gradient boosting
 - Personally I recommend the "extreme gradient boosting" implementation of this method, which is called *xgboost* both in R and python.
 - This implementation appears to be stronger and faster than the one in SAS

XGBOOST: A SCALABLE TREE BOOSTING SYSTEM

(T. CHEN, C. GUESTRIN, 2016)

NATALLIE BAIKEVICH

**HARDWARE ACCELERATION FOR
DATA PROCESSING SEMINAR**

ETH ZÜRICH

MOTIVATION

- ✓ **Effective**
statistical
models
- ✓ **Scalable** system
- ✓ **Successful**
real-world
applications



XGBoost
eXtreme
Gradient
Boosting

A BIT OF HISTORY

AdaBoost, 1996

Random Forests, 1999

Gradient Boosting Machine, 2001



A BIT OF HISTORY

AdaBoost, 1996

Random Forests, 1999

Gradient Boosting Machine, 2001

**Various improvements in tree
boosting**

XGBoost package



A BIT OF HISTORY

AdaBoost, 1996

Random Forests, 1999

Gradient Boosting Machine, 2001

Various improvements in tree boosting

XGBoost package

1st **Kaggle** success: Higgs Boson Challenge

17/29 winning solutions in 2015



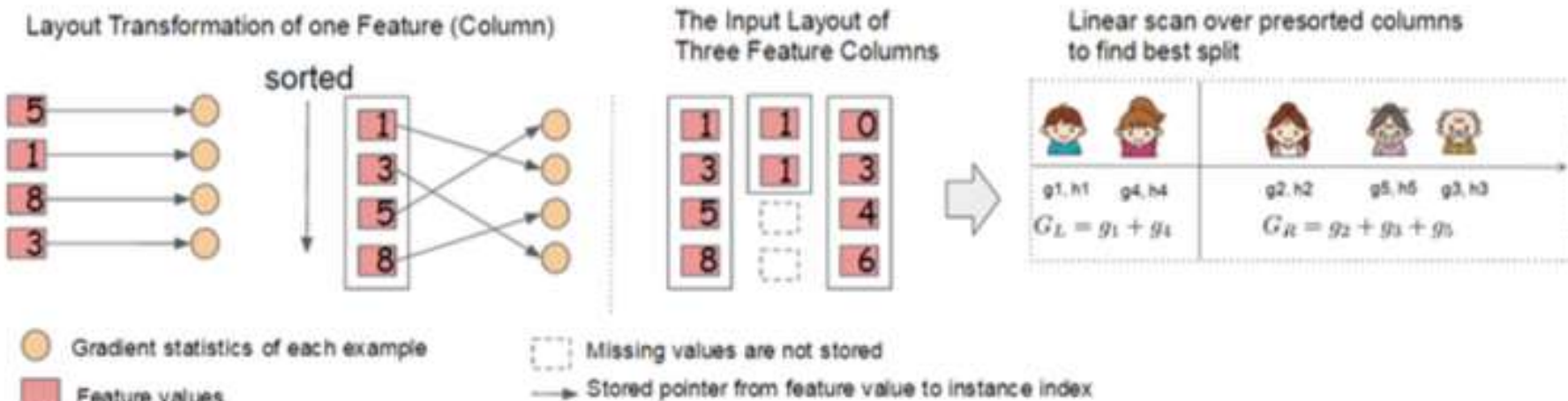
WHY DOES XGBOOST WIN "EVERY" MACHINE LEARNING COMPETITION?

- (MASTER THESIS, D. NIELSEN, 2016)

- Maksims Volkovs, Guangwei Yu and Tomi Poutanen, 1st place of the [2017 ACM RecSys challenge](#). Link to [paper](#).
- Vlad Sandulescu, Mihai Chiru, 1st place of the [KDD Cup 2016 competition](#). Link to the [arxiv paper](#).
- Marios Michailidis, Mathias Müller and HJ van Veen, 1st place of the [Dato Truly Native? competition](#). Link to the [Kaggle interview](#).
- Vlad Mironov, Alexander Guschin, 1st place of the [CERN LHCb experiment Flavour of Physics competition](#). Link to the [Kaggle interview](#).
- Josef Slavicek, 3rd place of the [CERN LHCb experiment Flavour of Physics competition](#). Link to the [Kaggle interview](#).
- Mario Filho, Josef Feigl, Lucas, Gilberto, 1st place of the [Caterpillar Tube Pricing competition](#). Link to the [Kaggle interview](#).
- Qingchen Wang, 1st place of the [Liberty Mutual Property Inspection](#). Link to the [Kaggle interview](#).
- Chenglong Chen, 1st place of the [Crowdfunder Search Results Relevance](#). Link to the [winning solution](#).
- Alexandre Barachant ("Cat") and Rafał Cycoń ("Dog"), 1st place of the [Grasp-and-Lift EEG Detection](#). Link to the [Kaggle interview](#).
- Halla Yang, 2nd place of the [Recruit Coupon Purchase Prediction Challenge](#). Link to the [Kaggle interview](#).
- Owen Zhang, 1st place of the [Avito Context Ad Clicks competition](#). Link to the [Kaggle interview](#).
- Keiichi Kuroyanagi, 2nd place of the [Airbnb New User Bookings](#). Link to the [Kaggle interview](#).
- Marios Michailidis, Mathias Müller and Ning Situ, 1st place [Homesite Quote Conversion](#). Link to the [Kaggle interview](#).

Source: <https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions>

SYSTEM DESIGN: BLOCK STRUCTURE



Max depth

Sorted structure → linear scan

$$O(Kd \|x\|_0 \log n)$$



$$O(Kd \|x\|_0 + \|x\|_0 \log B)$$

trees

non-missing entries

Blocks can be

✓ **Distributed** across machines

✓ **Stored** on disk in out-of-core setting

SYSTEM DESIGN: CACHE-AWARE ACCESS

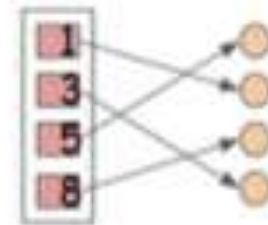
Improved split finding



Non-continuous memory access

- ✓ Allocate internal buffer
- ✓ Prefetch gradient statistics

Block Structure



Instructions

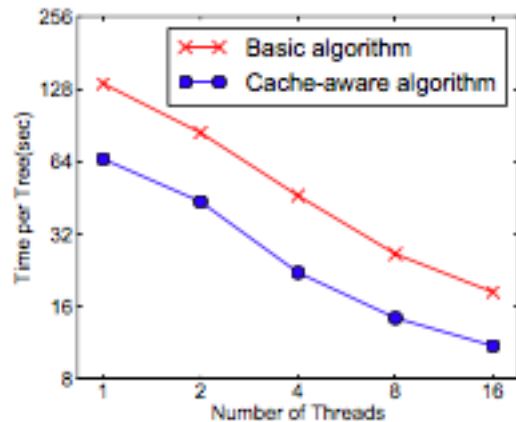
$G = G + g[\text{ptr}[i]]$

$H = H + h[\text{ptr}[i]]$

calculate score....

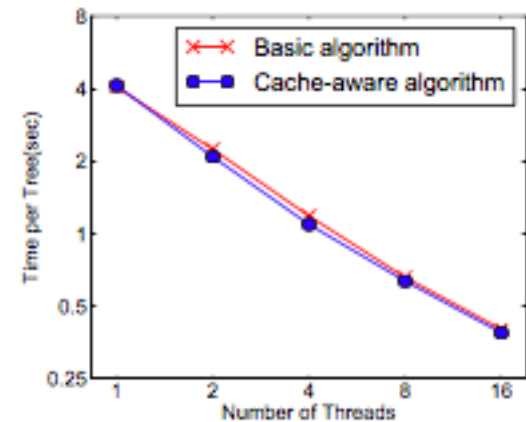
$G = G + g[\text{ptr}[i]]$

$H = H + h[\text{ptr}[i]]$



(b) Higgs 10M

Datasets:
Larger vs Smaller



(d) Higgs 1M

SYSTEM DESIGN: BLOCK STRUCTURE

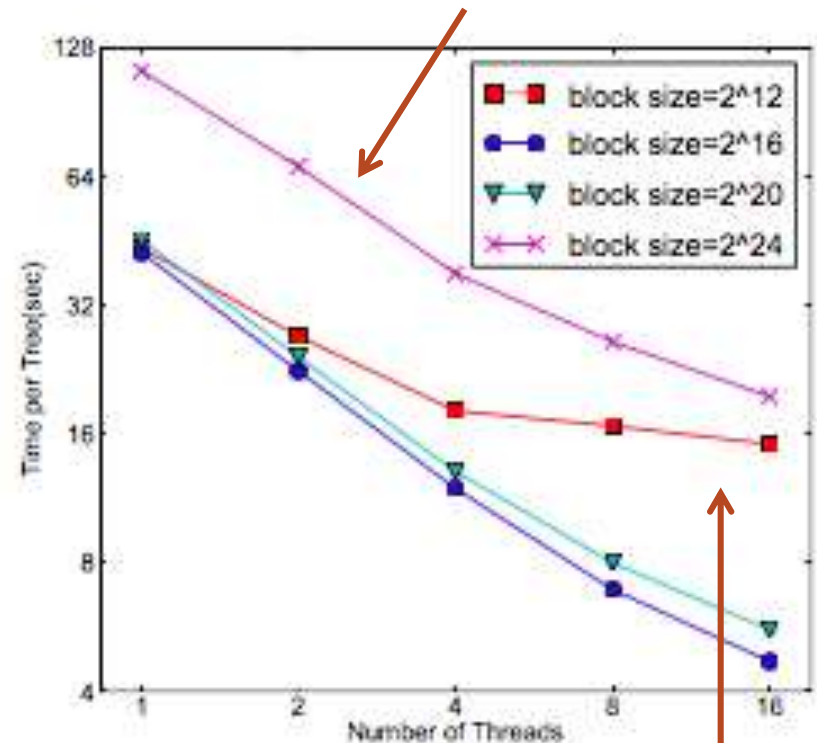
Prefetch
in independent thread

Compression by
columns (**CSC**):

Decompression
vs
Disk Reading

Block **sharding**:
Use multiple disks

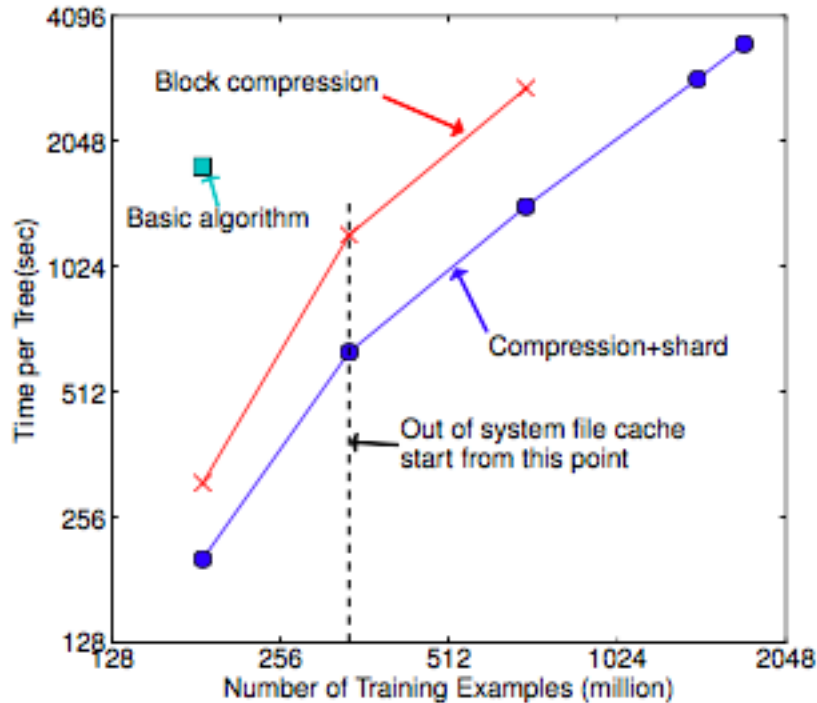
Too large blocks, cache misses



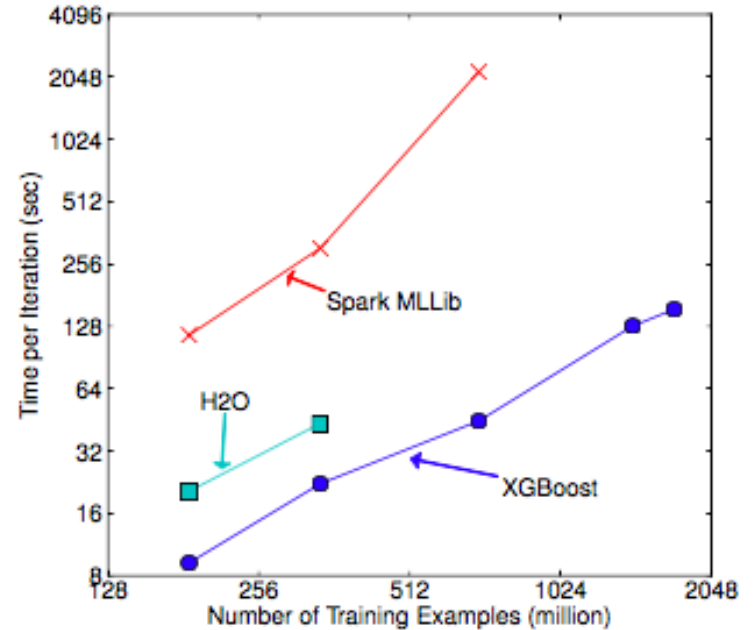
(a) Allstate 10M

Too small, inefficient
parallelization

EVALUATION



**AWS c3.8xlarge machine:
32 virtual cores, 2x320GB SSD,
60 GB RAM**



(b) Per iteration cost exclude data loading

**32 m3.2xlarge machines, each:
8 virtual cores, 2x80GB SSD,
30GB RAM**

DATASETS

Dataset	n	m	Task
Allstate	10M	4227	Insurance claim classification
Higgs Boson	10M	28	Event classification
Yahoo LTRC	473K	700	Learning to rank
Criteo	1.7B	67	Click through rate prediction

WHAT'S NEXT?

XGBoost

Scalability

Weighted quantiles

Sparsity-awareness

Cache-awareness

Data compression



Tuning

Hyperparameter optimization

Parallel Processing

GPU

FPGA

Model Extensions

DART (+ Dropouts)

LinXGBoost

More Applications