Boosting Methods

Jesse Ellin, Ohad Nir, Ritvik Vaish, Thomas Su

Stochastic Gradient Boosting

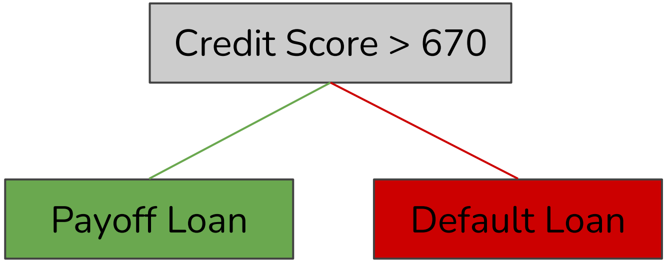
Jerome Friedman - Departments of Statistics and Stanford Linear Accelerator Center

Published Feb 28, 2002 in Computational Statistics and Data Analysis

What are Boosted Methods?

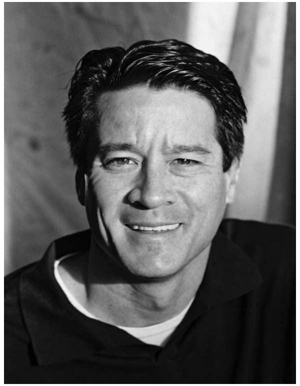
Weak Learnability and Weak Learners

- Put simply, a predictor that performs slightly better than random guessing.
- Unlike strong learners, we can’t guarantee convergence.
- Examples of Weak Learners:
  - Stump
  - Shallow network
- Examples of Strong Learners:
  - ERM using (Stochastic) Gradient Descent
  - Multi-layered Neural Networks



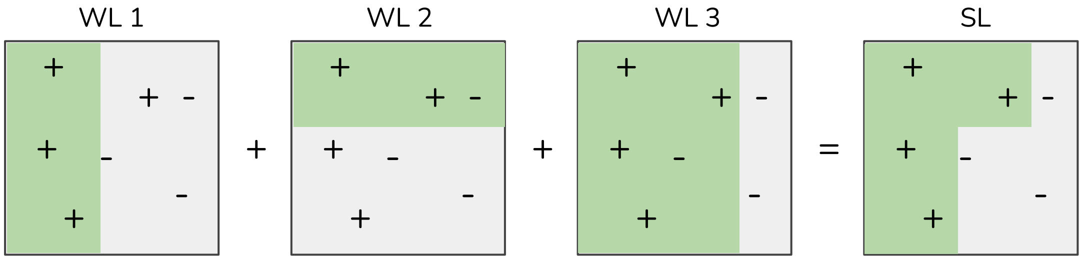
The History of the Boosting Paradigm

- Paradigms driving question:
  - Can an efficient weak learner can be “boosted” into an efficient strong learner?
  - Question proposed by Michael Kearns and Leslie Valiant in 1988.
    - Paper: Cryptographic Limitations on Learning Boolean Formulae and Finite Automata.
- Solutions:
  - Solved by Robert Schapire in 1990.
    - Paper: The strength of weak learnability.
  - Adaboost, the first practical algorithm for boosting was proposed by Robert Schapire and Yoav Freund in 1995 and later won the Godel Prize.
    - Paper: A Short Introduction to Boosting.



What is Boosting?

- Trains weak learners to be strong learners.
  - Does this through combining weak learners.
- Addresses two issues:
  - Bias-complexity tradeoff:
    - The error of an ERM could be decomposed into approximation error and estimation error.
      - Approximation error - Determined by the quality of our prior knowledge.
      - Estimation error - Error due to overfitting.
    - The larger the hypothesis class, the smaller the approximation error and the larger the estimation error.
    - Boosted methods allow the learner to control this tradeoff.
  - Computational complexity of learning:
    - In general, complex predictors are better predictors, but they are harder to train.
    - What if we take a linear combination of simple predictors?



Gradient Boosting vs Stochastic Boosting

Regular Boosting

- When training, we take the full training data set per epoch.
- More likely to overfit to the training data.
- Long training time.

Stochastic Boosting

- When training model, we sample random subsets of data (without replacement) over multiple epochs.
- Increases variance which make it less likely to overfit.
- Faster training time.

Algorithm 1: Gradient_TreeBoost

$F_0(\mathbf{x}) = \arg\min_\gamma \sum_{i=1}^{N} \Psi\left(y_i, \gamma\right)$

For $m = 1$ to M do:

$\quad \tilde{y}_{im} = -\left[\frac{\partial \Psi(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \ i = 1, N$

$\quad \{R_{lm}\}_1^L = L - \text{terminal node } tree(\{\tilde{y}_{im}, \mathbf{x}_i\}_1^N)$

$\quad \gamma_{lm} = \arg\min_\gamma \sum_{\mathbf{x}_i \in R_{lm}} \Psi\left(y_i, F_{m-1}(\mathbf{x}_i) + \gamma\right)$

$\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot \gamma_{lm} 1(\mathbf{x} \in R_{lm})$

endFor

Algorithm 2: Stochastic Gradient_TreeBoost

$F_0(\mathbf{x}) = \arg\min_\gamma \sum_{i=1}^{N} \Psi\left(y_i, \gamma\right)$

For $m = 1$ to M do:

$\quad \{\pi(i)\}_1^N = \text{rand\_perm } \{i\}_1^N$

$\quad \tilde{y}_{\pi(i)m} = -\left[\frac{\partial \Psi(y_{\pi(i)}, F(\mathbf{x}_{\pi(i)}))}{\partial F(\mathbf{x}_{\pi(i)})}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \ i = 1, \tilde{N}$

$\quad \{R_{lm}\}_1^L = L - \text{terminal node } tree(\{\tilde{y}_{\pi(i)m}, \mathbf{x}_{\pi(i)}\}_1^{\tilde{N}})$

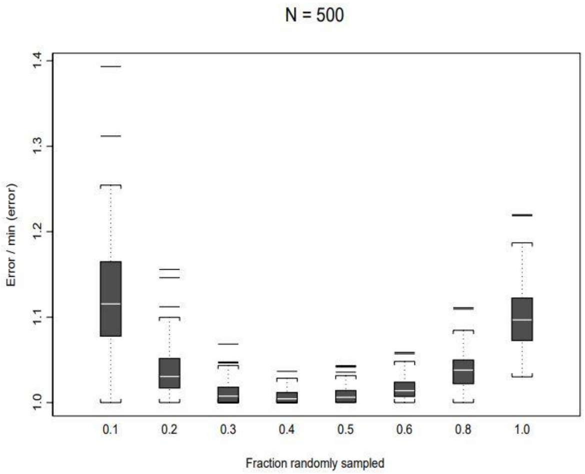
$\quad \gamma_{lm} = \arg\min_\gamma \sum_{\mathbf{x}_{\pi(i)} \in R_{lm}} \Psi\left(y_{\pi(i)}, F_{m-1}(\mathbf{x}_{\pi(i)}) + \gamma\right)$

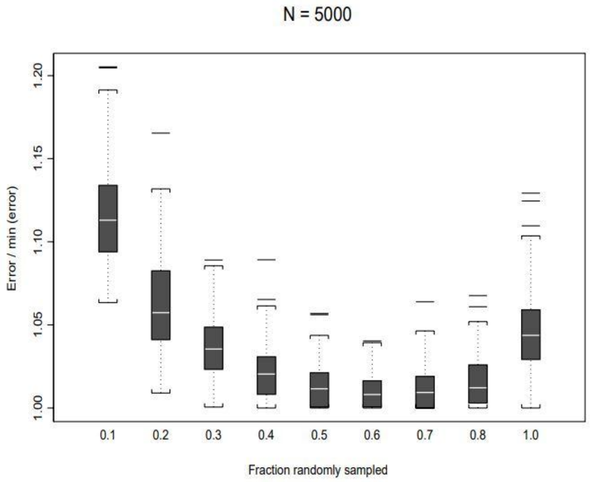
$\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot \gamma_{lm} 1(\mathbf{x} \in R_{lm})$

endFor

Papers Results on Regression Problem

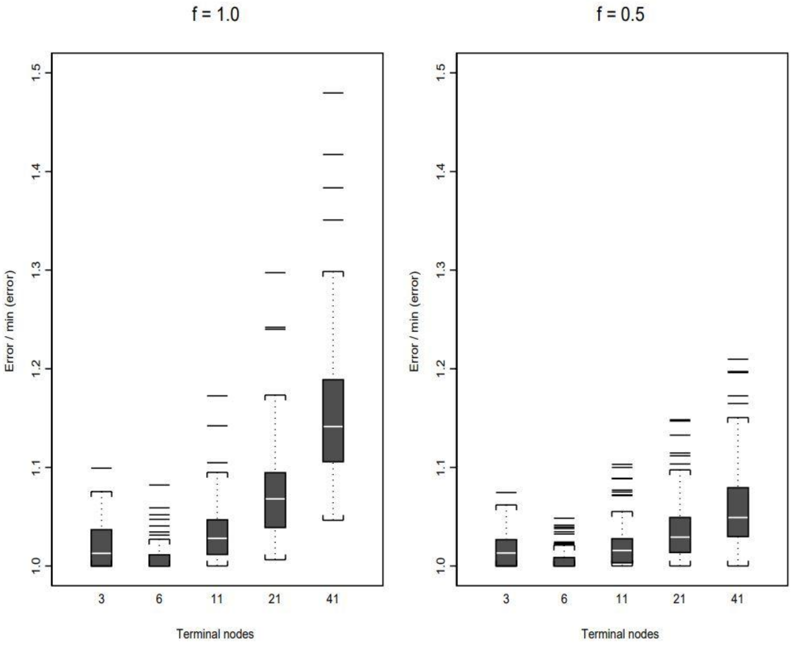
- Used M_TreeBoost procedure (Huber M-regression Loss).
  - Was found to be the best regression procedures in Friedman (1999).
- Tests were done one a small dataset with N=500 data points and a larger dataset with N=5000 data points.
- Different degrees of randomness where applied.
  - Controlled through the fraction f=Ñ/N.





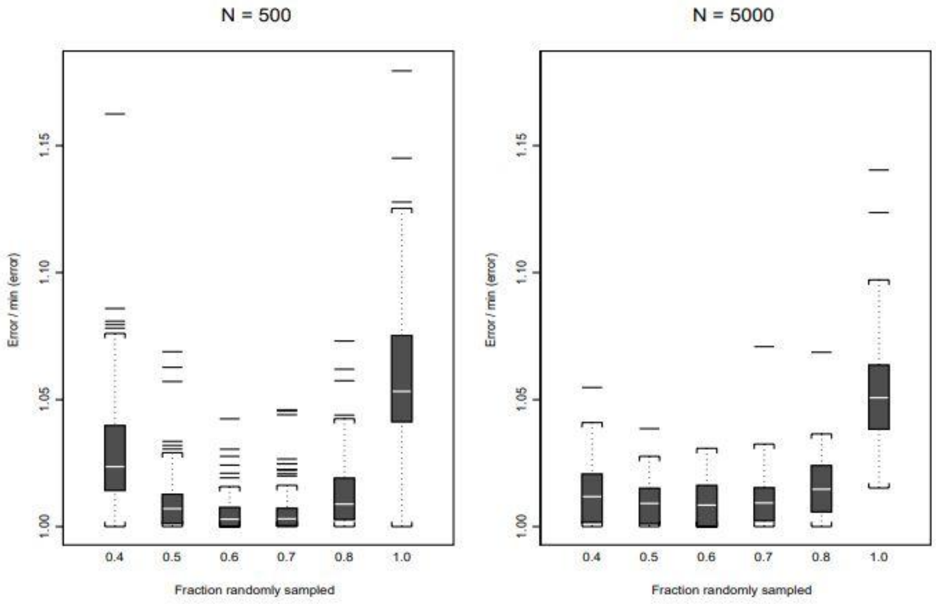
More Results on Regression Problem

- Used N=500 dataset using M_TreeBoost procedure.
- Shows Error over different number of terminal nodes (leaf nodes) base learners.



Papers Results on Classification Problem

- Experiment was done on a dual classification problem.
- Loss criteria was on twice binomial negative log-likelihood.
- Sampled an equal amount of data for both classes.



The Experiment

Our Motivation

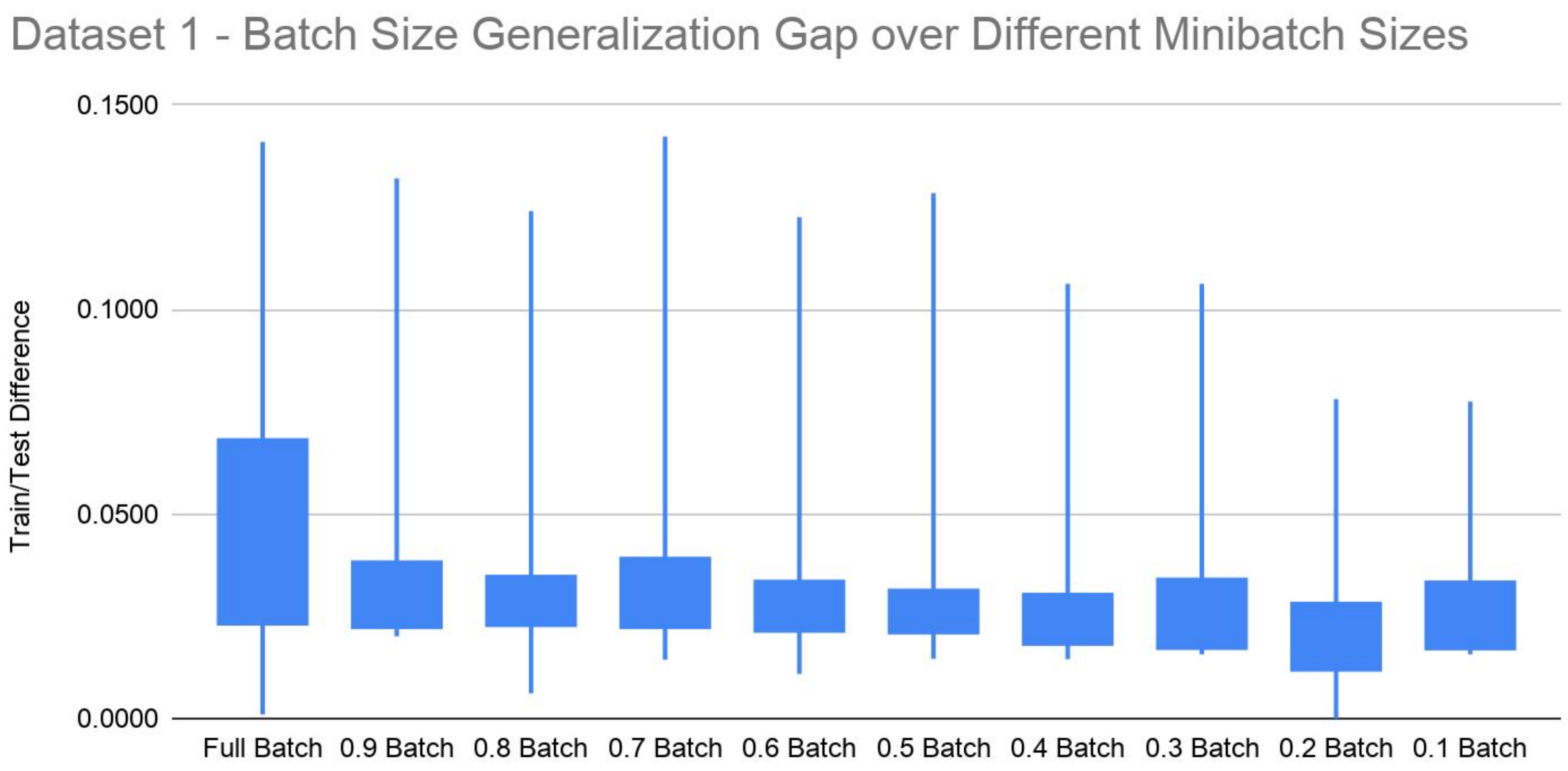
We want to compare Adaboost, a known and standard Boosting architecture, to a stochastic Boosting architecture to see how adding stochasticity improves or hinders the performance of Boosting.

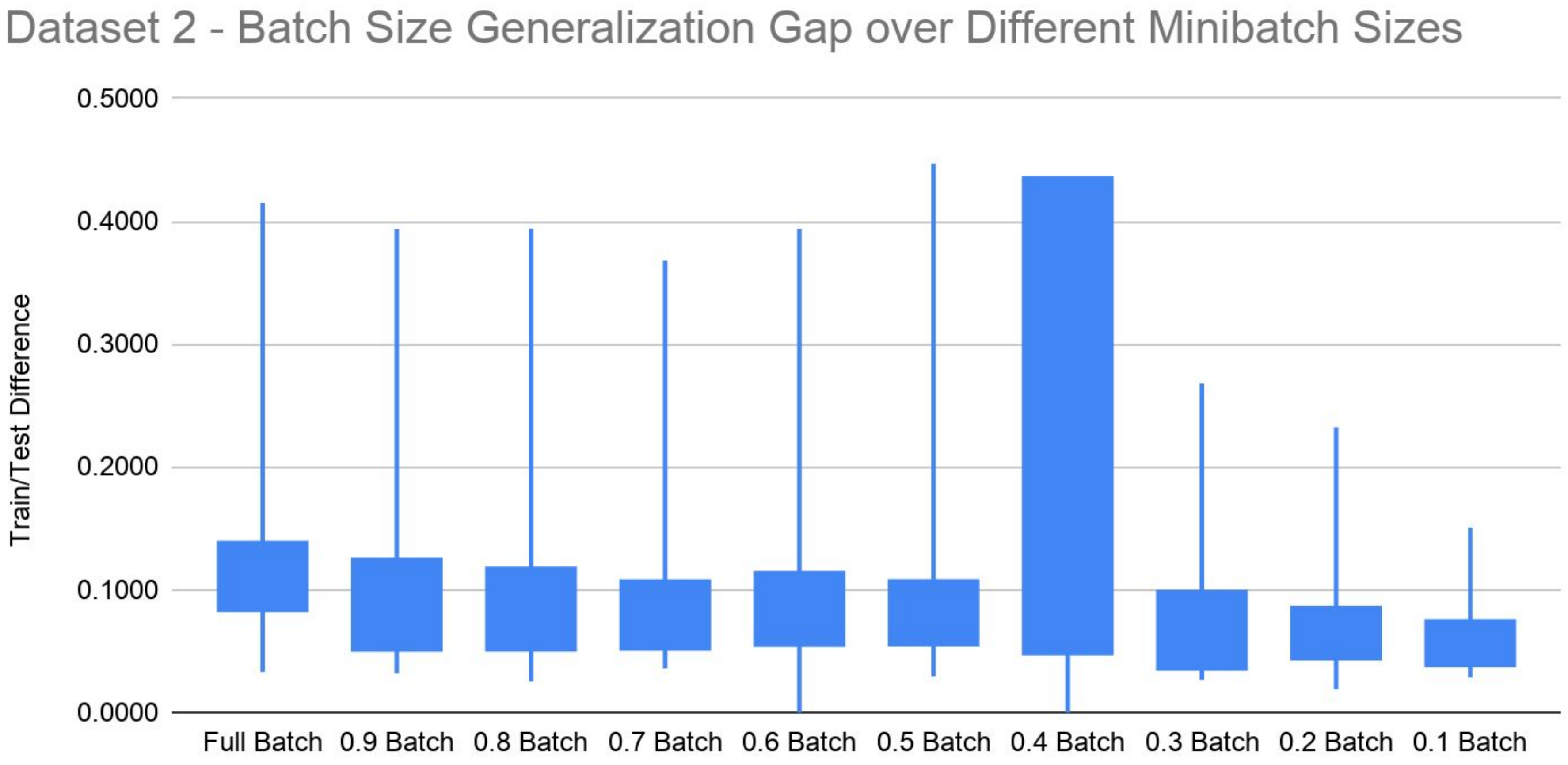
This will hopefully give us a stronger intuition of how Boosting performs.

Our Experiment

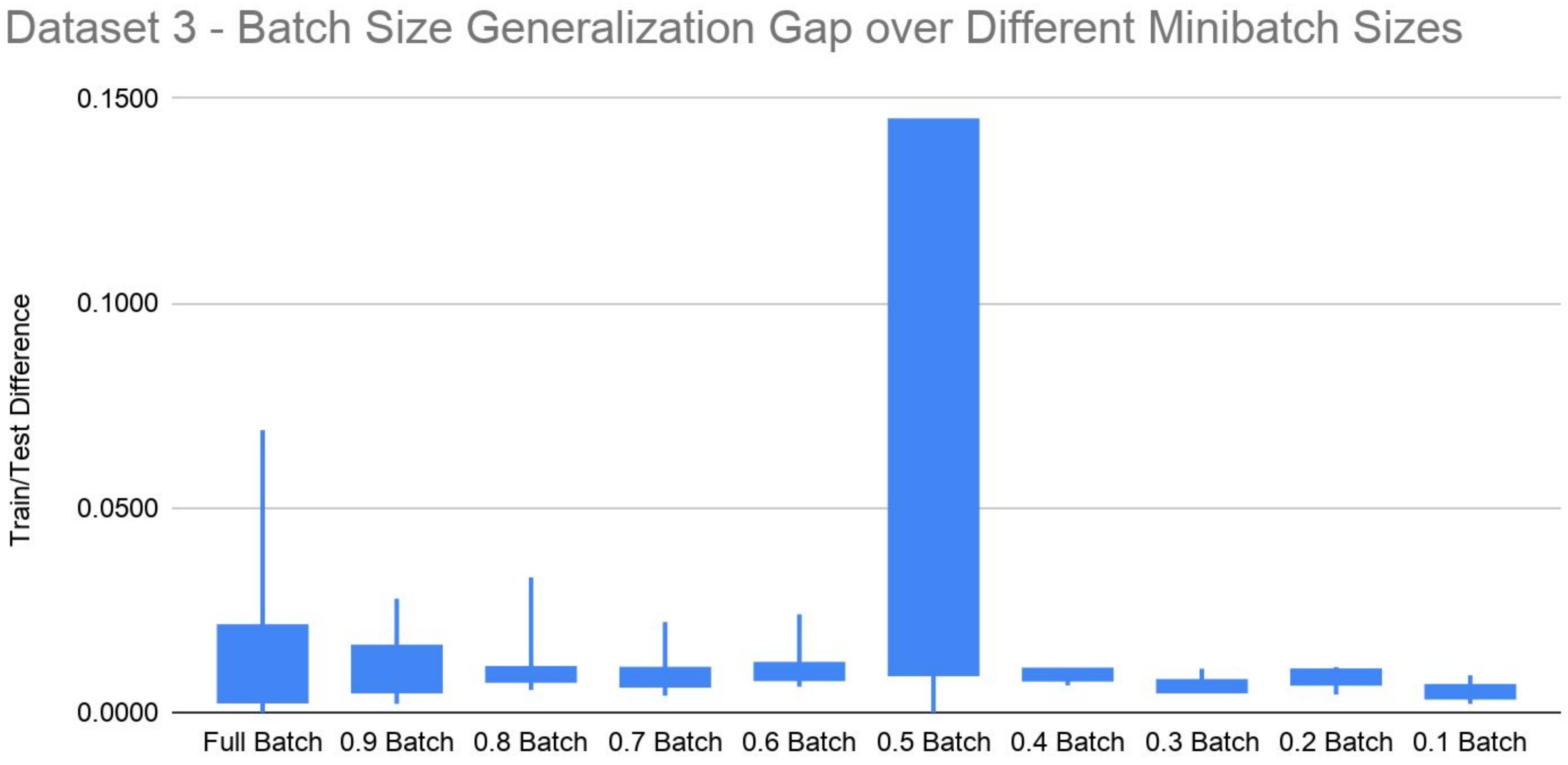
- Experiment with the behavior of boosting with different training data sizes and tree sizes.
    - See if adding stochastic processes results in similar improvement to the paper.
    - See how changing tree depth impacts overfitting trends
    - Is there a general change in runtime (determined by relative training times) when we change batch size or tree size
- Dataset:
    - Dataset 1 - Sign-Language MNIST
    - Dataset 2 - Chinese MNIST
    - Dataset 3 - Fashion MNIST

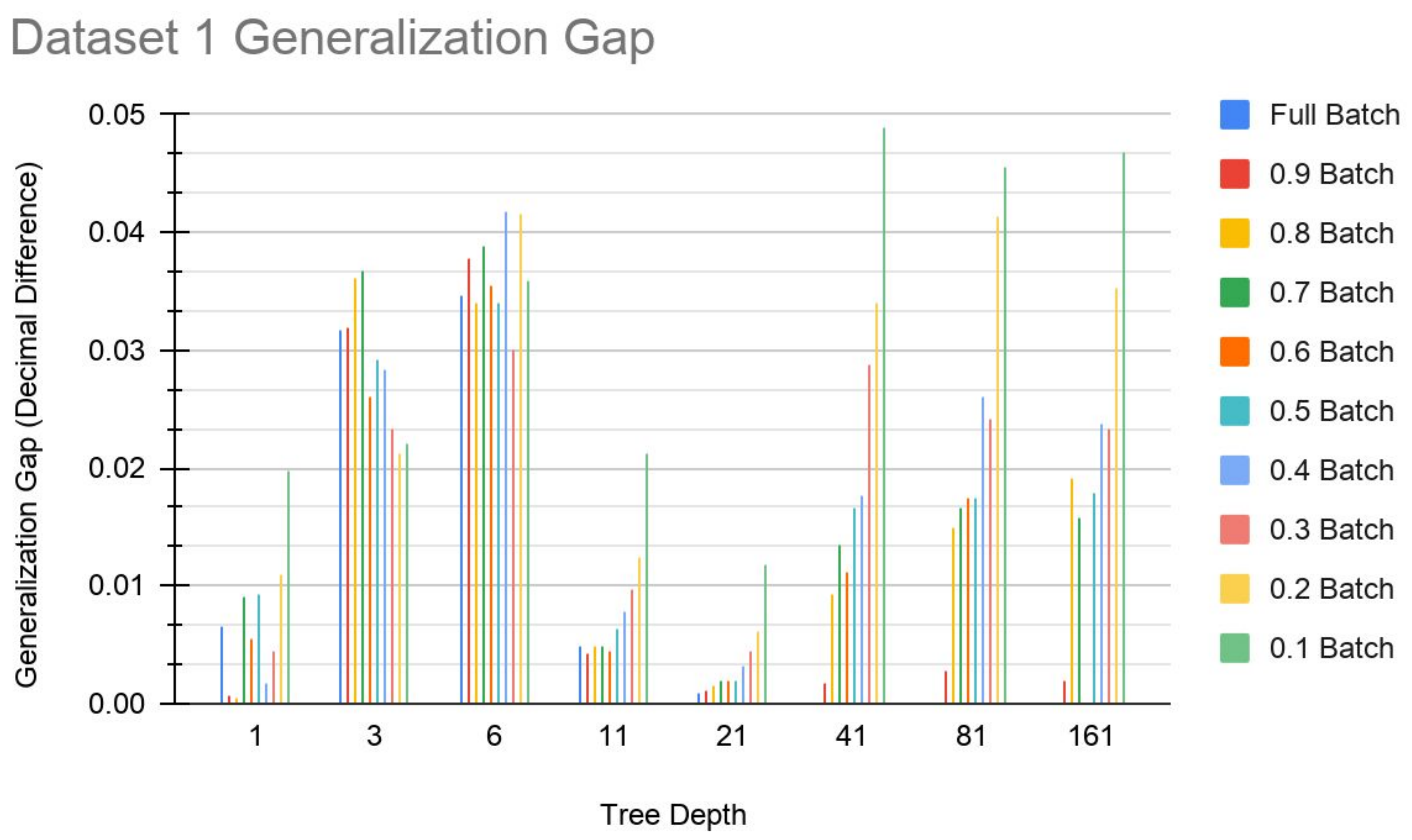
Results

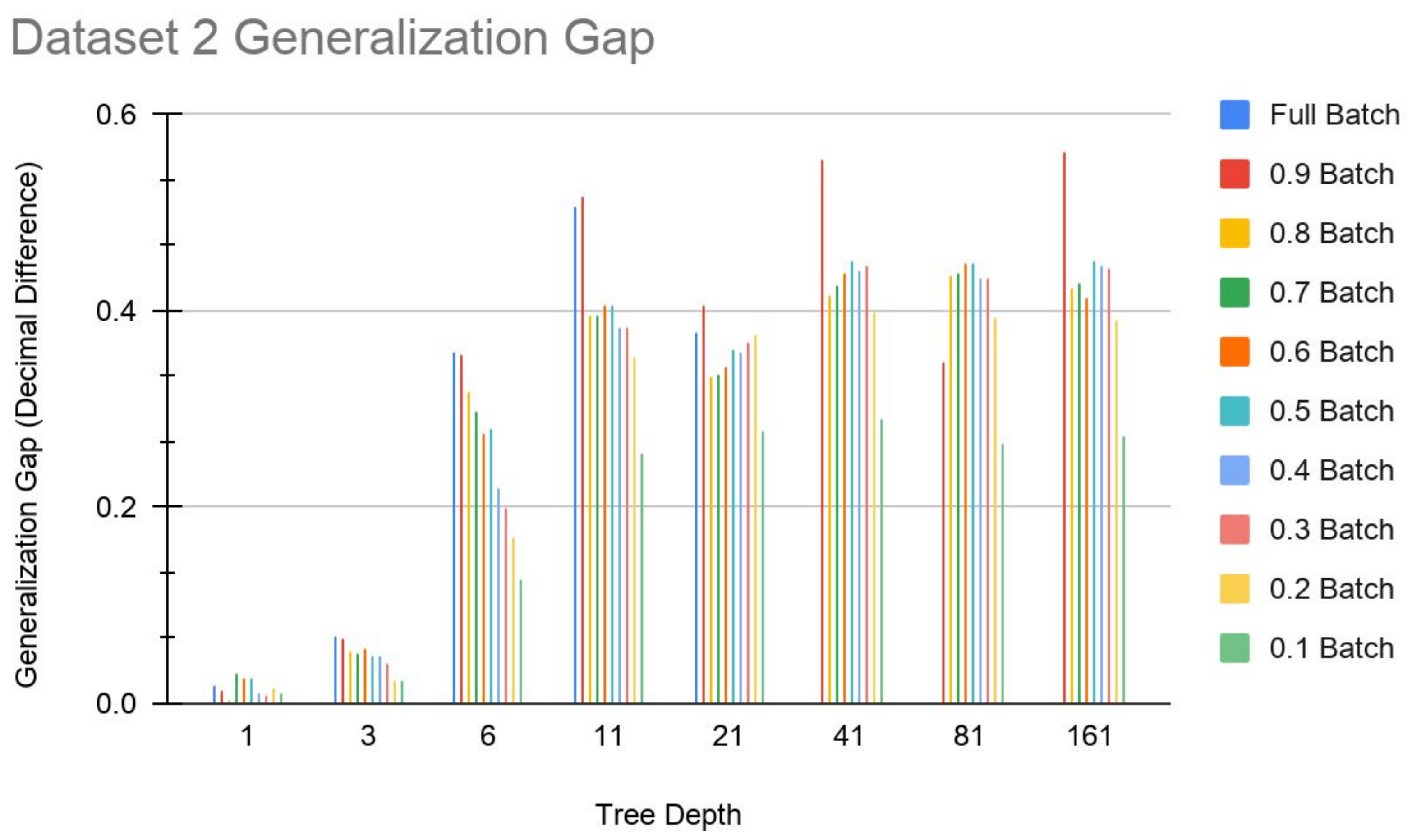

Dataset 1 - Batch Size Generalization Gap over Different Minibatch Sizes
Train/Test Difference
0.1500
0.1000
0.0500
0.0000
Full Batch
0.9 Batch
0.8 Batch
0.7 Batch
0.6 Batch
0.5 Batch
0.4 Batch
0.3 Batch
0.2 Batch
0.1 Batch

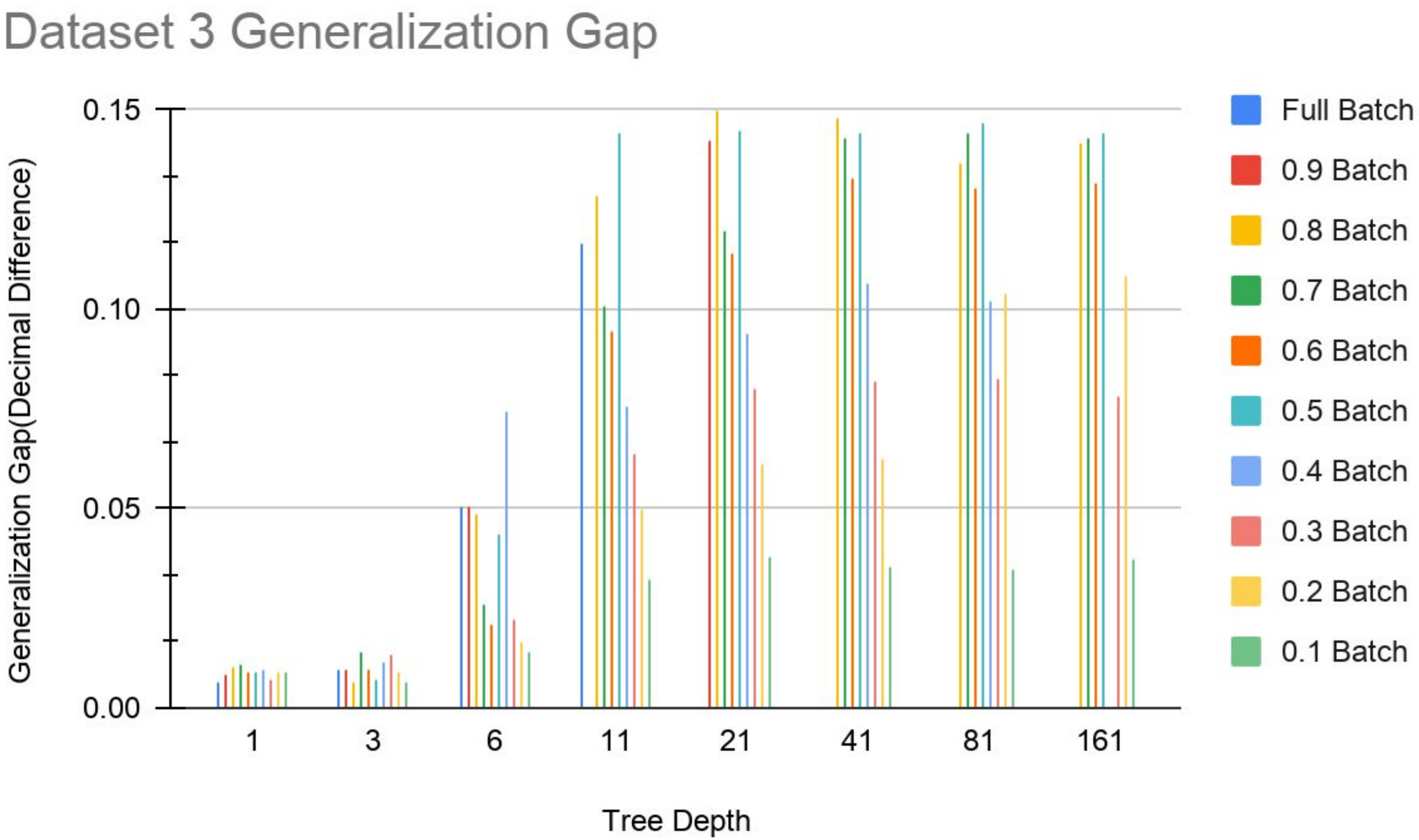


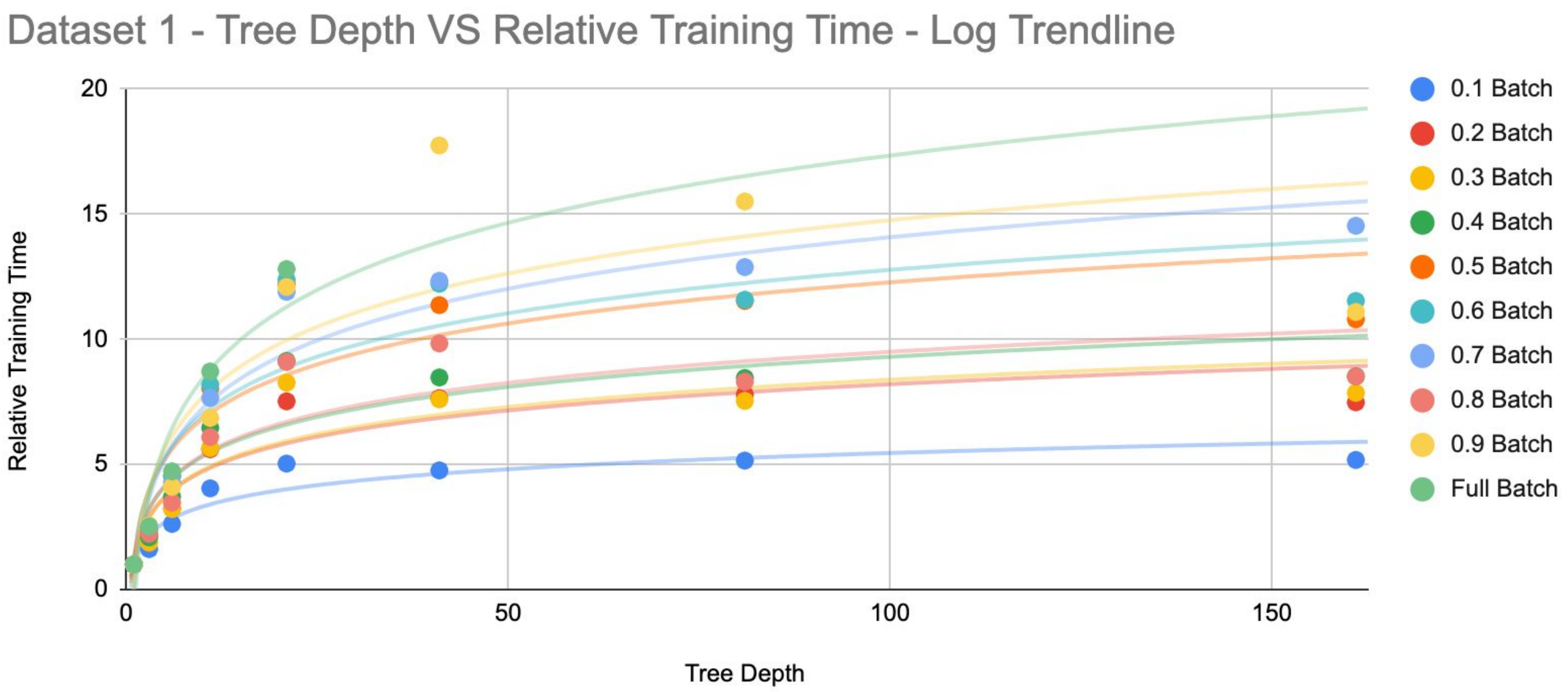
Dataset 2 - Batch Size Generalization Gap over Different Minibatch Sizes
Train/Test Difference
0.5000
0.4000
0.3000
0.2000
0.1000
0.0000
Full Batch
0.9 Batch
0.8 Batch
0.7 Batch
0.6 Batch
0.5 Batch
0.4 Batch
0.3 Batch
0.2 Batch
0.1 Batch


Dataset 3 - Batch Size Generalization Gap over Different Minibatch Sizes
Train/Test Difference
0.1500
0.1000
0.0500
0.0000
Full Batch
0.9 Batch
0.8 Batch
0.7 Batch
0.6 Batch
0.5 Batch
0.4 Batch
0.3 Batch
0.2 Batch
0.1 Batch

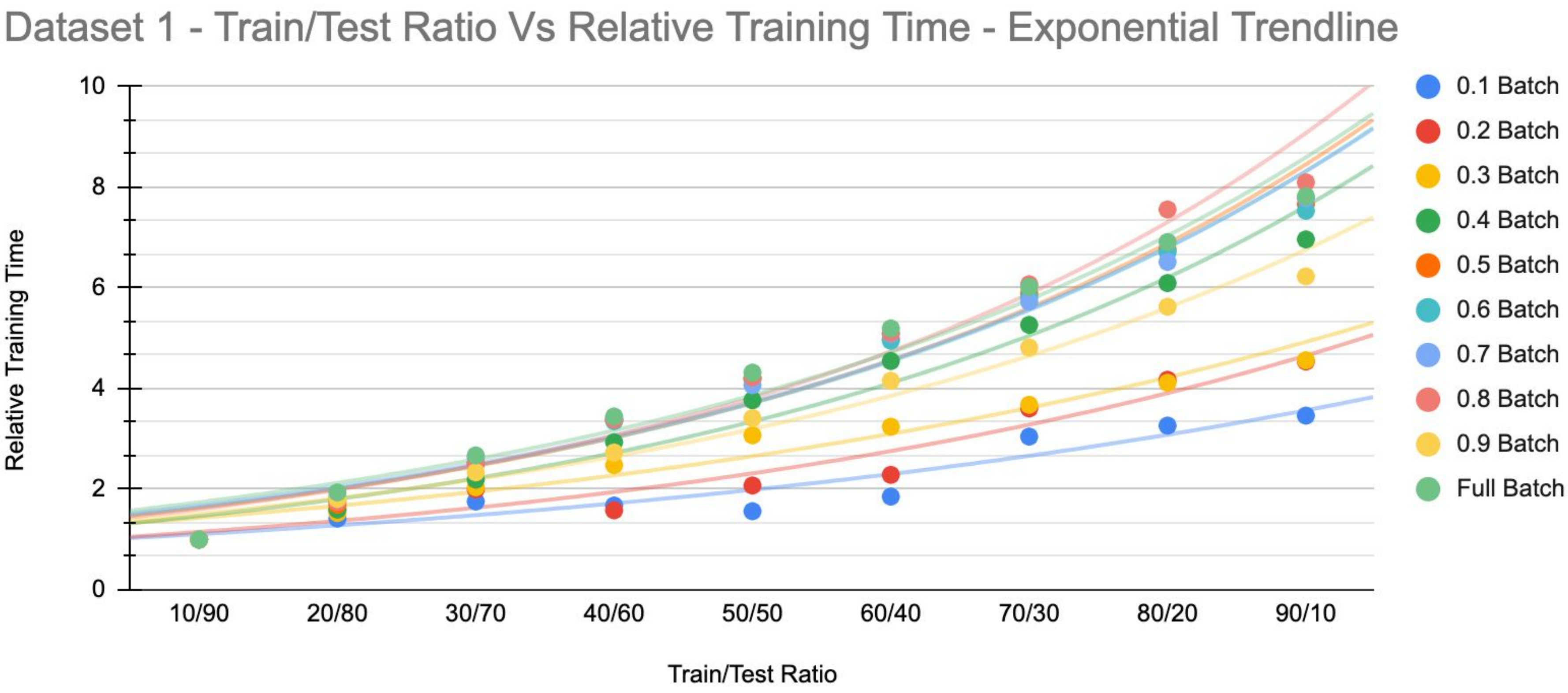

Dataset 1 Generalization Gap
Generalization Gap (Decimal Difference)
0.05
0.04
0.03
0.02
0.01
0.00
1
3
6
11
21
41
81
161
Tree Depth
Full Batch
0.9 Batch
0.8 Batch
0.7 Batch
0.6 Batch
0.5 Batch
0.4 Batch
0.3 Batch
0.2 Batch
0.1 Batch


Dataset 2 Generalization Gap
Generalization Gap (Decimal Difference)
0.6
0.4
0.2
0.0
1
3
6
11
21
41
81
161
Tree Depth
Full Batch
0.9 Batch
0.8 Batch
0.7 Batch
0.6 Batch
0.5 Batch
0.4 Batch
0.3 Batch
0.2 Batch
0.1 Batch


Dataset 3 Generalization Gap
Generalization Gap(Decimal Difference)
0.15
0.10
0.05
0.00
1
3
6
11
21
41
81
161
Tree Depth
Full Batch
0.9 Batch
0.8 Batch
0.7 Batch
0.6 Batch
0.5 Batch
0.4 Batch
0.3 Batch
0.2 Batch
0.1 Batch



Dataset 1 - Tree Depth VS Relative Training Time - Log Trendline
Relative Training Time
Tree Depth
0
5
10
15
20
0
50
100
150
0.1 Batch
0.2 Batch
0.3 Batch
0.4 Batch
0.5 Batch
0.6 Batch
0.7 Batch
0.8 Batch
0.9 Batch
Full Batch


Dataset 1 - Train/Test Ratio Vs Relative Training Time - Exponential Trendline
Relative Training Time
10
8
6
4
2
0
10/90
20/80
30/70
40/60
50/50
60/40
70/30
80/20
90/10
Train/Test Ratio
0.1 Batch
0.2 Batch
0.3 Batch
0.4 Batch
0.5 Batch
0.6 Batch
0.7 Batch
0.8 Batch
0.9 Batch
Full Batch

Homework

Homework Instruction

- Implementing a framework for a multiclass version of Adaboost.
  - Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME).
  - Stochastic
- Train your framework on two data sets:
  - Speaker Accent Recognition DataSet.
  - Deepfakes: Medical Image Tamper Detection DataSet.
- Will use two kinds of weak learners.
  - Gaussian Naive Bayes
  - Shallow Decision Tree
- Compare:
  - Regular boosting and stochastic boosting.
  - Weak learners

SAMME Multiclass AdaBoost Algorithm

Algorithm 2 *SAMME*

1. *Initialize the observation weights* $w_i = 1/n,\ i = 1, 2, \ldots, n$.
2. *For* $m = 1$ *to* M*:*
   (a) *Fit a classifier* $T^{(m)}(\boldsymbol{x})$ *to the training data using weights* w_i.
   (b) *Compute*
   $$err^{(m)} = \sum_{i=1}^{n} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{n} w_i.$$
   (c) *Compute*
   $$\alpha^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1).$$
   (d) *Set*
   $$w_i \leftarrow w_i \cdot \exp\left(\alpha^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),\ i = 1, \ldots, n.$$
   (e) *Re-normalize* w_i.
3. *Output*
   $$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \alpha^{(m)} \cdot \mathbb{I}(T^{(m)}(\boldsymbol{x}) = k).$$

Questions?

Resources

- Friedman, Jerome H. “Stochastic Gradient Boosting.” Statweb.stanford.edu, 26 Mar. 1999, statweb.stanford.edu/~jhf/ftp/stobst.pdf.
- Shalev-Shwartz, Shai, and Shai Ben-David. Understanding Machine Learning: from Theory to Algorithms. Cambridge University Press, 2019.
- Zhu, Ji, et al. “Multi-Class AdaBoost.” Web.stanford.edu, 12 Jan. 2006, web.stanford.edu/~hastie/Papers/samme.pdf.

Data

Experiment:

- Keras.datasets, FashionMNIST, https://keras.io/api/datasets/fashion_mnist/
- Kaggle, Chinese MNIST, https://www.kaggle.com/gpreda/chinese-mnist
- Kaggle, Sign Language MNIST, https://www.kaggle.com/datamunge/sign-language-mnist

Homework:

- UCI, Speaker Accent Recognition, https://archive.ics.uci.edu/ml/datasets/Speaker+Accent+Recognition#
- UCI, Deepfakes: Medical Image Tamper Detection, https://archive.ics.uci.edu/ml/datasets/Deepfakes%3A+Medical+Image+Tamper+Detection