Distributed Confidence-Weighted Classification on MapReduce

Nemanja Djuric\textsuperscript{1}, Mihajlo Grbovic\textsuperscript{2}, Slobodan Vucetic\textsuperscript{1}

\textsuperscript{1} Temple University, Philadelphia, USA
\textsuperscript{2} Yahoo! Labs, Sunnyvale, USA

IEEE Big Data, Santa Clara, October 7\textsuperscript{th}, 2013
Outline of the talk

1. Introduction
   - Motivation behind the proposed approach
   - Machine Learning using MapReduce

2. Related work
   - Confidence-Weighted (CW) classification
   - AROW training of CW classifiers

3. Proposed approach
   - Distributed training of CW classifiers (AROW-MR)

4. Experiments and conclusion
   - Validate the proposed method on synthetic data
   - Evaluate on real-world, industrial-size Ad Latency task
Introduction

- Big Data is pervasive; data sets with millions of examples and features are now a rule rather than an exception
  - Crowdsourcing, remote sensing, social networks, etc.

- Globally-recognized, strategic importance of Big Data
  - Focus of all major internet companies
  - “Big Data Research and Development Initiative” by US govt.

- Many challenges to machine learning and data mining researchers due to its large-scale nature
Introduction

- Explosive growth in data size, complexity, and rates resulted in data of unprecedented scales
  - Standard classification tools are not capable of addressing these large-scale tasks
  - Even linear time and space complexity of efficient SVM solvers is not tractable for modern data sets

- We propose a linear SVM solver for large-scale training of recently proposed Confidence-Weighted (CW) classifiers
  - Distributed, sub-linear training using MapReduce framework
  - Significant improvement over state-of-the-art linear classifiers
  - Evaluated on real-world, large-scale Ad Latency task
Hadoop and MapReduce

- Combines distributed filesystem with MapReduce framework
- Hadoop Distributed Filesystem (HDFS)
  - Distributes data files among servers automatically
  - Default replication factor of 3
- MapReduce
  - Easier to send code to data than vice versa with big data
  - Each job is a sequence of map and reduce operations
  - Mappers load data, perform basic transformations
  - Reducers process mapper output records with a single key
  - Complex operations typically happen in mappers
Hadoop and MapReduce

Input Files:
- Apple Orange Mango
- Orange Grapes Plum
- Apple Plum Mango
- Apple Apple Plum

Each line passed to individual mapper instances:
- Apple Orange Mango
- Orange Grapes Plum
- Apple Plum Mango
- Apple Apple Plum

Map Key Value Splitting:
- Apple, 1
- Orange, 1
- Mango, 1
- Apple, 1
- Orange, 1
- Plum, 1
- Apple, 1
- Plum, 1
- Mango, 1
- Apple, 1
- Mango, 1
- Orange, 1
- Plum, 1
- Plum, 1

Sort and Shuffle:
- Apple, 1
- Apple, 1
- Apple, 1
- Mango, 1
- Mango, 1
- Plum, 1
- Plum, 1
- Apple, 1
- Apple, 1
- Orange, 1
- Orange, 1

Reduce Key Value Pairs:
- Apple, 4
- Grapes, 1
- Mango, 2
- Orange, 2
- Plum, 3

Final Output:
- Apple, 4
- Grapes, 1
- Mango, 2
- Orange, 2
- Plum, 3
Hadoop and MapReduce

- Java-based
  - Compatible with any language using JVM
  - Can “stream” data into shell commands for other languages

- Parallelism
  - Typically 1 mapper per input file (can split further)
  - Number of reducers must be specified (summary operation)

- Significant overhead with launching jobs
  - Highly iterative algorithms suffer greatly
Four ways of using MapReduce for machine learning

**Option 1**: Learn 1 model on 1 reducer (1 job)
- Reading the data in multiple mappers
- Learning a model on a single reducer in an online learning manner without storing the points that are being streamed
- Learning a model takes as long as learning on a single machine
- The only benefit is in data storage
Option 2: Learn 1 model in batch mode on $M$ mappers

- Mappers compute gradients and the reducer sums them
- One MapReduce job is analogous to one batch GD update
- Requires running several MapReduce jobs

Disadvantage: this is ineffective

1. Each iteration has large overheads (e.g., job scheduling, data transfer, data parsing)
2. At least a dozen iterations (i.e., MapReduce jobs) often need to be conducted to ensure convergence
Option 3: learn 1 model in mini-batches on $M$ mappers (1 job)

- AllReduce abstraction
- A spanning tree for communicating between mappers
- Local gradients are summed up the tree, and then broadcast down to all mappers

Disadvantage: this is not robust

1. If one mapper fails job is stuck
2. All mappers need to run at the same time (sometimes not possible – think 1,000 mappers on a busy queue) – if not possible the job is stuck
Option 4: learn $M$ models in $M$ mappers and combine models on 1 reducer (1 job)

- Learning of $M$ models – one on each mapper
- Combine $M$ models into 1 model on the reducer

**Advantage:** mappers are independent of each other (they don’t need to communicate or run concurrently)

**Disadvantage:** not many algorithms out there
Confidence-Weighted classification

- Proposed by Dredze et al., ICML 2009

- Confidence-Weighted (CW) binary classifier, in addition to the margin, outputs confidence in the prediction
  - Assumes a multivariate Gaussian over separating hyperplanes
  - Given a trained CW model, this induces a Gaussian distribution over the prediction margin for a new point \((x, y)\)

\[
\hat{y} \sim \mathcal{N}(y(\mu^T x), x^T \Sigma x)
\]

- Following the assumption of Gaussianity, we can compute the prediction confidence as follows

\[
\Pr(\text{sign}(\mu^T x) = y) = \frac{1}{2} (1 + \text{erf}(\frac{y(\mu^T x)}{\sqrt{2} x^T \Sigma x}))
\]
CW training

- The CW classifier is trained in an online manner
  - New parameter estimates should be close to those from the previous iteration
  - Maximize prediction confidence for current training example
- The authors solve the following optimization problem
  \[
  (\mu_{t+1}, \Sigma_{t+1}) = \arg \min_{\mu, \Sigma} D_{KL}(\mathcal{N}(\mu, \Sigma) \mid \mid \mathcal{N}(\mu_t, \Sigma_t))
  \]
  subject to \quad \mathbb{P}(y_t (\mu^T x_t \geq 0)) \geq \eta
- CW classifier is susceptible to noise: performs too aggressive updates due to the constraint
AROW training

- Adaptive Regularization of Weight Vectors (AROW) proposed by Crammer et al., NIPS 2009

- Online training algorithm is derived having in mind the following constraint
  - Margin for a new training point should be maximized, while uncertainty minimized

- Solve the following optimization problem at each iteration

\[
(\mu_{t+1}, \Sigma_{t+1}) = \arg \min_{\mu, \Sigma} D_{KL}(\mathcal{N}(\mu, \Sigma) \| \mathcal{N}(\mu_t, \Sigma_t)) + \\
\lambda_1 \left( \max(0, 1 - y_t \mu^T x_t) \right)^2 + \lambda_2 (x_t^T \Sigma x_t)
\]
After finding derivatives of the objective function with respect to mean and covariance matrix, we obtain the following update rule whenever misclassification occurs:

\[ \mu_{t+1} = \mu_t + \alpha_t y_t \Sigma_t x_t, \]
\[ \Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t x_t x_t^T \Sigma_t \]

where \( \alpha_t = \beta_t \max(0, 1 - y_t \mu^T x_t) \)
\[ \beta_t = (x_t^T \Sigma x_t + r)^{-1} \]
\[ r = 1/(2\lambda_1), \text{ for } \lambda_1 = \lambda_2 \]

The training proceeds in rounds until convergence.
We utilize MapReduce framework to significantly speed up the training of CW classifiers.

- **Map phase** – Train a number of independent CW classifiers on each mapper, send the learned parameters to reducer.
- **Reduce phase** – Aggregate local, mapper-specific classifiers into a single CW classifier on a reducer.
AROW training on MapReduce

- Train a CW classifier on each of $M$ mappers to obtain local, mapper-specific parameters $\mu_m$ and $\Sigma_m$, $m = 1, \ldots, M$

- Minimize the following objective function on the reducer:
  \[
  \mathcal{L} = \mathbb{E}_{\mathcal{N}(\mu, \Sigma)} [D_{KL}^S(\mathcal{N}(\mu_*, \Sigma_*)||\mathcal{N}(\mu, \Sigma))]
  \]
  or its empirical estimate:
  \[
  \mathcal{L} = \sum_{m=1}^{M} P(\mathcal{N}(\mu_m, \Sigma_m)) \ D_{KL}^S(\mathcal{N}(\mu_*, \Sigma_*)||\mathcal{N}(\mu_m, \Sigma_m))
  \]

- We can obtain closed-form updates for mean vector and covariance matrix of the multivariate Gaussian
Finding derivative of the loss function with respect to the mean and covariance matrix, we obtain updates

\[
\mu_* = \left( \sum_{m=1}^{M} \left( \mathbb{P}(\mathcal{N}(\mu_m, \Sigma_m)) \left( \Sigma_*^{-1} + \Sigma_m^{-1} \right) \right) \right)^{-1} \left( \sum_{m=1}^{M} \left( \mathbb{P}(\mathcal{N}(\mu_m, \Sigma_m)) \left( \Sigma_*^{-1} + \Sigma_m^{-1} \right) \right) \mu_m \right)
\]

\[
\Sigma_* \left( \sum_{m=1}^{M} \mathbb{P}(\mathcal{N}(\mu_m, \Sigma_m)) \Sigma_m^{-1} \right) \Sigma_* = \sum_{m=1}^{M} \mathbb{P}(\mathcal{N}(\mu_m, \Sigma_m)) \left( \Sigma_m + (\mu_* - \mu_m)(\mu_* - \mu_m)^T \right)
\]

The 2\textsuperscript{nd} equation is an algebraic Riccati equation of the form $XAX=B$, solved as

\[
X = U^{-0.5} B^{0.5} (U^T)^{-0.5}, \text{ with } A = U^T U
\]
Experiments – Synthetic data

- waveform data set (50,000 training, 5,000 test examples)
  - Increased no. of mappers from 1 to 100, repeated 10 times
  - We report results of AROW, the proposed AROW-MR, and AROW-single (local mapper model used by AROW-MR)

- Distributed AROW-MR obtains significantly improved training time and test accuracy
Experiments – Ad Latency

- Real-world, industrial-size Ad Latency data set
  - 1.3 billion data examples, 21 measured features
- Online advertising domain
  - Improve online experience through timely delivery of relevant ads to the users
  - Can we detect if the ad will be late before it is served?
- Features:
  - **user features** (browser type, device type, ISP, location, connection speed, etc.)
  - **ad features** (ad type, ad size, ad dimensions, etc.),
  - **vendor features** (where is the ad served from, hardware used, etc.)
Experiments – Ad Latency

We compared AROW-MR to non-distributed AROW, as well as to the state-of-the-art Vowpall Wabbit (VW).

- Increased no. of mappers to evaluate effects of parallelization

AROW-MR decreased training time from 17 days to 25 minutes, with further accuracy gains!

Outperformed linear VW classifier with comparable training times
Inadequacy of standard machine learning tools in large-scale setting is apparent

- Novel methods are necessary in order to address a plethora of Big Data problems

We proposed AROW-MR, a large-scale, efficient linear SVM solver based on the state-of-the-art CW classifiers

- AROW-MR validated on synthetic, as well as real-world, industrial-size Ad Latency data sets
- Outperformed state-of-the-art, large-scale linear classifiers
Thank you!

- Questions?