# Distributed Confidence-Weighted Classification on MapReduce

Nemanja Djuric[1], Mihajlo Grbovic[2], Slobodan Vucetic[1]

[1] Temple University, Philadelphia, USA

[2] Yahoo! Labs, Sunnyvale, USA

IEEE Big Data, Santa Clara, October 7th, 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

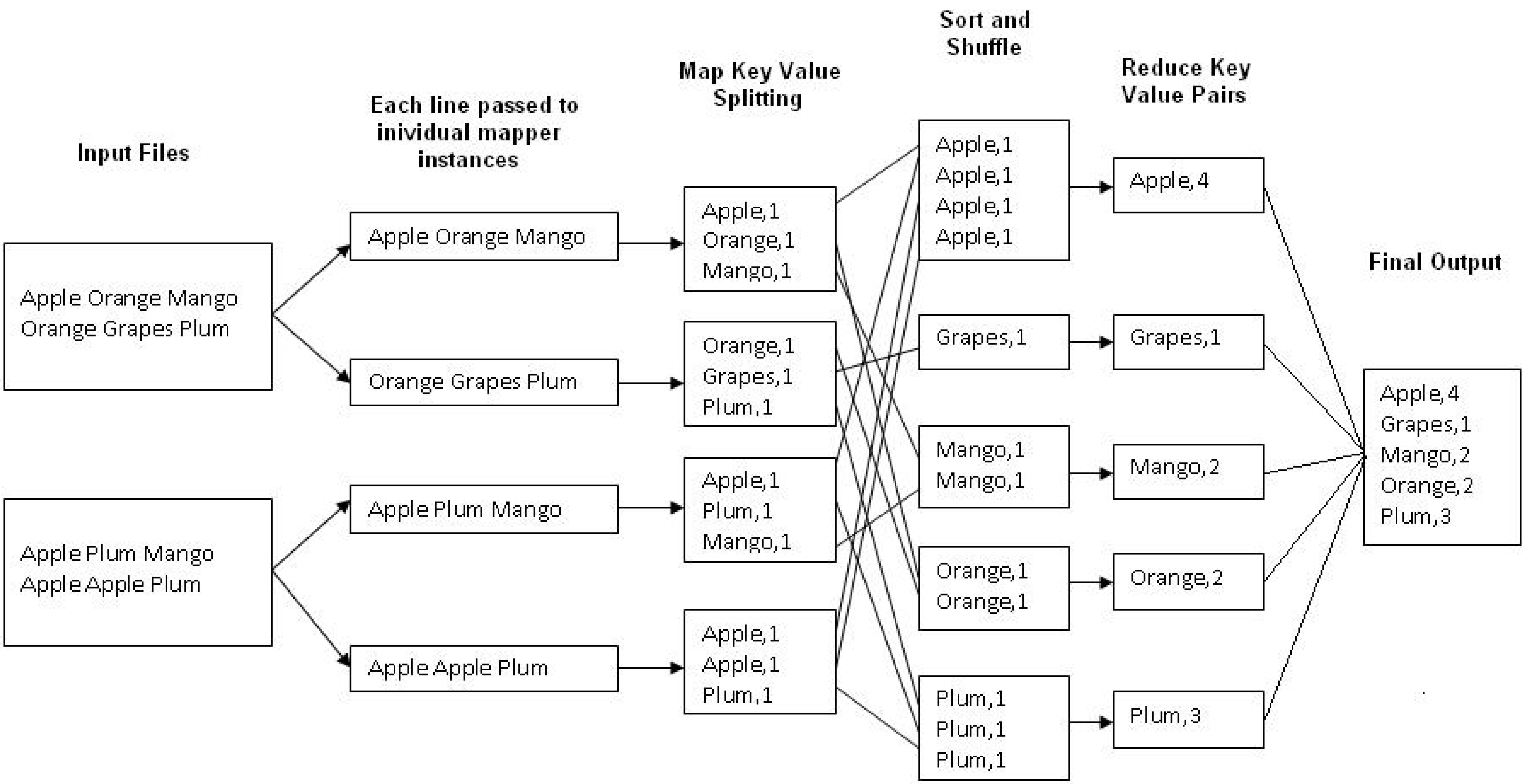
# Introduction

- 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



# 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

# Machine Learning and MapReduce

- 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

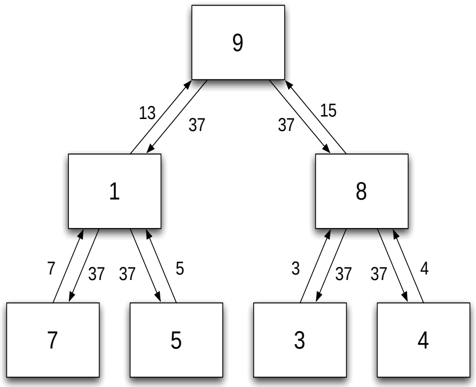
# Machine Learning and MapReduce

- **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

# Machine Learning and MapReduce

- **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



# Machine Learning and MapReduce

- **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 $(\mathbf{x}, y)$

$$\hat{y} \sim \mathcal{N}\big(y(\boldsymbol{\mu}^{\mathrm{T}}\mathbf{x}), \mathbf{x}^{\mathrm{T}}\Sigma\mathbf{x}\big)$$

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

$$\mathbb{P}(\mathrm{sign}(\boldsymbol{\mu}^{\mathrm{T}}\mathbf{x}) = y) = \frac{1}{2}\big(1 + \mathrm{erf}(\frac{y(\boldsymbol{\mu}^{\mathrm{T}}\mathbf{x})}{\sqrt{2\mathbf{x}^{\mathrm{T}}\Sigma\mathbf{x}}})\big)$$

# 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

$$(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}) = \arg\min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} D_{KL}\big(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \| \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)\big)$$
$$\text{subject to} \quad \mathbb{P}\big(y_t(\boldsymbol{\mu}^{\mathrm{T}} \mathbf{x}_t \geq 0)\big) \geq \eta$$

- 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

$$
\begin{aligned}
(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1}) = \arg\min_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} & D_{KL}\big(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \| \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)\big) + \\
& \lambda_1 \big(\max(0, 1 - y_t \boldsymbol{\mu}^{\mathrm{T}} \mathbf{x}_t)\big)^2 + \lambda_2 (\mathbf{x}_t^{\mathrm{T}} \Sigma \mathbf{x}_t)
\end{aligned}
$$

# AROW training

- After finding derivatives of the objective function with respect to mean and covariance matrix, we obtain the following update rule whenever misclassification occurs

$$\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}_t + \alpha_t y_t \boldsymbol{\Sigma}_t \mathbf{x}_t,$$
$$\boldsymbol{\Sigma}_{t+1} = \boldsymbol{\Sigma}_t - \beta_t \boldsymbol{\Sigma}_t \mathbf{x}_t \mathbf{x}_t^{\mathrm{T}} \boldsymbol{\Sigma}_t$$

where
$$\alpha_t = \beta_t \max(0, 1 - y_t \boldsymbol{\mu}^{\mathrm{T}} \mathbf{x}_t)$$
$$\beta_t = (\mathbf{x}_t^{\mathrm{T}} \Sigma \mathbf{x}_t + r)^{-1}$$
$$r = 1/(2\lambda_1), \text{ for } \lambda_1 = \lambda_2$$

- The training proceeds in rounds until convergence

# AROW training on MapReduce

- 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 $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$, $m = 1, \ldots, M$
- Minimize the following objective function on the reducer

$$\mathcal{L} = \mathbb{E}_{\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[D_{KL}^{S}(\mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \| \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}))]$$

or its empirical estimate

$$\mathcal{L} = \sum_{m=1}^{M} \mathbb{P}(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m))\ D_{KL}^{S}(\mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \| \mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m))$$

- We can obtain closed-form updates for mean vector and covariance matrix of the multivariate Gaussian

# AROW training on MapReduce

- Finding derivative of the loss function with respect to the mean and covariance matrix, we obtain updates

$$\boldsymbol{\mu}_* = \Big(\sum_{m=1}^{M} \left(\mathbb{P}(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)) \; (\boldsymbol{\Sigma}_*^{-1} + \boldsymbol{\Sigma}_m^{-1})\right)\Big)^{-1} \Big(\sum_{m=1}^{M} \left(\mathbb{P}(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)) \; (\boldsymbol{\Sigma}_*^{-1} + \boldsymbol{\Sigma}_m^{-1})\right)\boldsymbol{\mu}_m\Big)$$

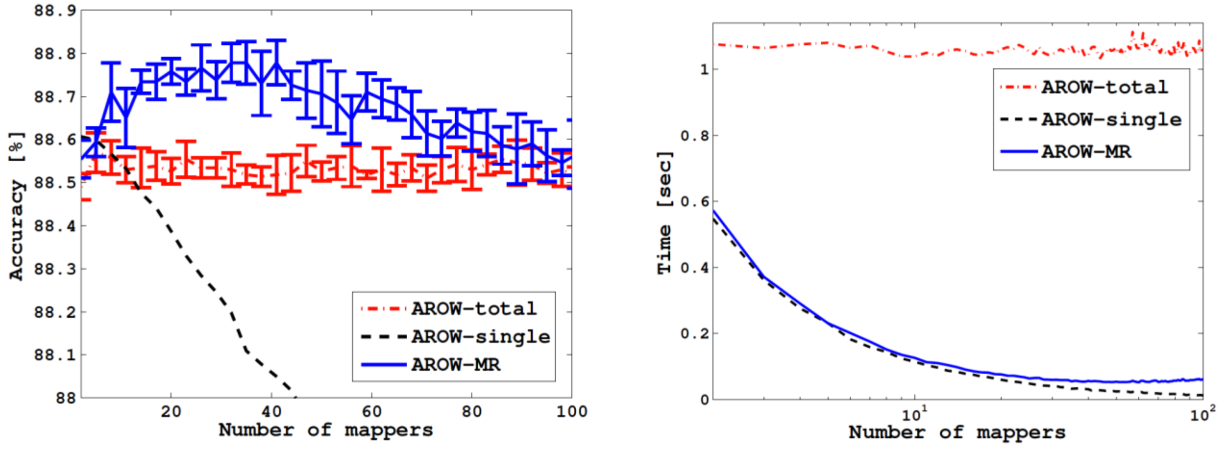
$$\boldsymbol{\Sigma}_*\Big(\sum_{m=1}^{M} \mathbb{P}(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)) \; \boldsymbol{\Sigma}_m^{-1}\Big)\boldsymbol{\Sigma}_* = \sum_{m=1}^{M} \mathbb{P}(\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)) \; \Big(\boldsymbol{\Sigma}_m + (\boldsymbol{\mu}_* - \boldsymbol{\mu}_m)(\boldsymbol{\mu}_* - \boldsymbol{\mu}_m)^{\mathrm{T}}\Big)$$

- The 2nd equation is an algebraic Riccati equation of the form **XAX**=**B**, solved as

$$\mathbf{X} = \mathbf{U}^{-0.5}\mathbf{B}^{0.5}(\mathbf{U}^{\mathrm{T}})^{-0.5}, \text{ with } \mathbf{A} = \mathbf{U}^{\mathrm{T}}\mathbf{U}$$

# 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




(a) Classification accuracy (b) Training time

# 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

Table 1. Increasing number of mappers

| # mappers | # reducers | Avg. map time | Reduce time | AUC |
|---|---|---|---|---|
| 1 | 0 | 408h | n/a | 0.8442 |
| 100 | 1 | 30.5h | 1 min | 0.8552 |
| 500 | 1 | 34 min | 4 min | 0.8577 |
| 1,000 | 1 | 17.5 min | 7 min | 0.8662 |
| 10,000 | 1 | 2 min | 1h | 0.8621 |

Table 2. Performance of VW

| # mappers | # reducers | Avg. map time | Reduce time | AUC |
|---|---|---|---|---|
| 1 | 0 | 7h | n/a | 0.8506 |
| 100 | 0 | 1h | n/a | 0.8508 |
| 500 | 0 | 8 min | n/a | 0.8501 |
| 1,000 | 0 | 6 min | n/a | 0.8498 |

- AROW-MR decreased training time from 17 days to 25 minutes, with further accuracy gains!
- Outperformed linear VW classifier with comparable training times

# Conclusion

- 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?