Data Driven Resource Allocation for Distributed Machine Learning

Venkata Krishna Pillutla
www.cs.cmu.edu/~vpillutl
Thesis Committee

• Nina Balcan, Chair
• Alex Smola
• Christos Faloutsos
Collaborators
Machine Learning is Changing the World

“A breakthrough in machine learning would be worth ten Microsofts”
(Bill Gates, Chairman, Microsoft)

“Machine learning is the next Internet”
(Tony Tether, former director, DARPA)

“Machine learning is the hot new thing”
(John Hennessy, President, Stanford)
The World is Changing ML

Outbreak of the “Data Epidemic”

New Applications
Introduction/Motivation
Machine Learning

• Traditional ML is centralized
• All the data is assumed to be on one machine
Distributed ML

Big Data in Google

- 100 hours/min
- 100 petabytes
- 500+ million users
- 900+ million devices
Distributed ML

Massive data is inherently distributed!

Also stored in a distributed manner. Eg: Yahoo! PNUTS
Distributed ML

In other cases, massive data centrally collected

This talk focuses on this scheme
Distributed ML

Communication: important resource (in addition to computation)
Typical Example: Learning Task

- Spam vs Not Spam
Another Example: Learning Task

Online advertising: Speed is of essence
Another Example: Learning Task

Online advertising: Speed is of essence
Another Example: Learning Task

Online advertising: Speed is of essence

User

RESPONSE
Another Example: Learning Task

Online advertising: Speed is of essence
How to partition the data?
Random Partitioning

Machine 1

Machine 2

Machine 3
Random Partitioning

• Advantages
  – Easy to implement
  – Clean theory

• Disadvantages
  – Statistically sub-optimal

• Can we do better?
Our idea: Data dependent partitioning

Vapnik:
Locally Simple but
Globally Complex!
Pros and Cons

• Advantages
  – Distributed
  – More expressive concept class!
  – Better performance at same communication

• Possible Concern
  – More expressive dispatch rule is required
Data Dependent Partitioning

• How? Clustering
Data Dependent Partitioning

• For efficiency, cluster an initial sample
Requirements I

- Load-Balancing
Requirements II

• Fault-tolerance
Requirements III

• Efficient dispatch during deployment
Contributions*

• Balanced Clustering with Fault Tolerance
  – NP-hard
  – Approximation algorithm with strong guarantees

• Nearest Neighbor Dispatch
  – Efficient, Online Dispatch
  – Provably good

• Experiments
  – Classification accuracy after data dependent partitioning
  – Scalability

*Joint work with: Travis Dick, Mu Li, Colin White, Maria-Florina Balcan, Alex Smola
Under submission at AISTATS 2016
Balanced Clustering with Fault Tolerance
## Requirements

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load balancing:</td>
<td>Well studied [KS, ABC+, ABG+]</td>
</tr>
<tr>
<td>Upper bound on cluster size:</td>
<td>$L$ fraction</td>
</tr>
<tr>
<td>Load balancing:</td>
<td>Not studied; very tricky</td>
</tr>
<tr>
<td>Lower bound on cluster size:</td>
<td>$l$ fraction</td>
</tr>
<tr>
<td>Fault tolerance:</td>
<td>$p$ replication</td>
</tr>
</tbody>
</table>

29
Lower bounds are tricky

• Typically: $OPT_k$ decreases as $k$ increases
• With lower bounds:
  – Arbitrary number of local maxima [DLP+]
Handling Size Constraints
Handling Size Constraints

Not center based!
Algorithm Overview

• Notation:
  – $y_i$: point $i$ is a center: opening
  – $x_{i,j}$: point $i$ is the center corresponding to $j$: assignments
  – $V$: set of points

• Works for any metric space $(\mathcal{X}, d)$
LP Relaxation

\[ K\text{-median: } c_{i,j} = d(i, j) \]
\[ K\text{-means: } c_{i,j} = d(i, j)^2 \]

\[
\min \sum_{i,j \in V} c_{ij} x_{ij} \\
\text{subject to: } \sum_{i \in V} x_{ij} = p, \quad \forall j \in V
\]
\[
\ell y_i \leq \sum_{j \in V} \frac{x_{ij}}{n} \leq Ly_i, \quad \forall i \in V
\]
\[
\sum_{i \in V} y_i \leq k; \\
0 \leq x_{ij} \leq y_i \leq 1, \quad \forall i, j \in V.
\]

\[ y_i : \text{opening} \]
\[ x_{ij} : \text{assignment} \]
LP Relaxation

• May open $2k$ half centers- requires rounding
Algorithm Overview

• Step 1: Solve LP

• Step 2: Round opening
  – Greedy Coarse Clustering to get \( \leq k \) coarse clusters: Monarch Procedure
  – Round centers locally within each coarse cluster

• Step 3: Round assignments
  – Round assignments globally with min-cost flow
Algorithm Overview: Step 1

- Solve LP
- Example: 8 points
Algorithm Overview: Step 2

- Perform coarse clustering: Monarch procedure
- Greedy
- Good guarantees
Algorithm Overview: Step 2

- Perform coarse clustering: Monarch procedure
- Greedy
- Good guarantees
Algorithm Overview: Step 2

• Round opening within each coarse cluster
Algorithm Overview: Step 2

• Round opening within each coarse cluster
Algorithm Overview: Step 2

- Round opening within cluster
Algorithm Overview: Step 3

- Round Assignments with min-cost flow.
Step 2: Monarch Procedure

- Greedily pick $\leq k$ points as monarchs
Step 2: Monarch Procedure

- Empires: Voronoi partitions about monarchs
Step 2: Monarch Procedure

- Greedy rule: pick point with highest contribution to the objective (as long as it does not have a monarch nearby)
Step 2: Monarch Procedure

- Why this greedy rule?
Step 2: Monarch Procedure Guarantees

- Points within an empire are close
Step 2: Monarch Procedure Guarantees

• Monarchs are far apart
Step 2: Monarch Procedure Guarantees

• Each empire has opening $\geq p/2$: Markov Inequality
Step 2: Monarch Procedure Guarantees

- Each empire has opening at least 1! (for $p>1$)
- Round *locally* within each empire!
Step 2: Rounding within Empire

- Pick \( \lceil Y_\mathcal{E} \rceil \) central points from each empire, each with opening \( Y_\mathcal{E}/\lceil Y_\mathcal{E} \rceil \); make centers

\[
Y_\mathcal{E} := \sum_{i \in \mathcal{E}} y_i
\]
LP Relaxation

\[ \begin{align*}
  K\text{-median: } & \quad c_{i,j} = d(i, j) \\
  K\text{-means: } & \quad c_{i,j} = d(i, j)^2 \\
  \min & \quad \sum_{i,j \in V} c_{i,j} x_{ij} \\
  \text{subject to: } & \quad \sum_{i \in V} x_{ij} = p, \quad \forall j \in V \\
  & \quad \ell y_i \leq \sum_{j \in V} \frac{x_{ij}}{n} \leq L y_i, \quad \forall i \in V \\
  & \quad \sum_{i \in V} y_i \leq k; \\
  & \quad 0 \leq x_{ij} \leq y_i \leq 1, \quad \forall i, j \in V.
\end{align*} \]
Step 2: Rounding within Empire

- Same factor appears as violation of cluster size constraint: $\frac{Y_\varepsilon}{\lfloor Y_\varepsilon \rfloor} \leq \frac{p+2}{p}$
Step 2: Rounding Guarantee

- Obtain a feasible solution with integral $y$
- Cost bounded by triangle inequality
Step 3: Rounding assignments

• Easy: Min cost flow

\[ \text{cost} = c_{ij} \]
\[ \text{capacity} = 2 \]

\[ \text{cost} = 0 \]
\[ \text{capacity} = \left[ \frac{nL(p + 2)}{p} \right] \]

\[ \text{supply} = k\ell n - np \]

\[ \text{supply} = -n\ell \]

\[ \text{supply} = p \]
Step 3: Rounding assignments

- Fractional LP solution implies a feasible flow

\[ \text{cost} = c_{ij} \]
\[ \text{capacity} = 2 \]

\[ \text{cost} = 0 \]
\[ \text{capacity} = \left\lfloor nL(p + 2)/p \right\rfloor \]

\[ \text{supply} = knL - np \]

\[ \text{supply} = -nL \]

\[ \text{supply} = p \]
Step 3: Rounding assignments

- By Integral Flow Theorem, there is an optimal integral flow

![Graph Diagram]

- Cost: $c_{ij}$
- Capacity: $2$
- Supply: $p$
- Capacity: $\left[ nL(p + 2)/p \right]$
- Supply: $kn\ell - np$
Step 3: Rounding assignments

• Can be computed by standard algorithms

\[
\begin{align*}
  \text{cost} &= c_{ij} \\
  \text{capacity} &= 2 \\
  \text{supply} &= p
\end{align*}
\]

\[
\begin{align*}
  \text{cost} &= 0 \\
  \text{capacity} &= \left\lceil nL(p + 2)/p \right\rceil \\
  \text{supply} &= kn\ell - np
\end{align*}
\]

\[
\begin{align*}
  \text{supply} &= -n\ell
\end{align*}
\]
To sum up...

**Theorem:** There exist poly time approximation algorithms for balanced $k$-clustering with fault tolerance

- that output
  - 5 approx. for $k$-center
  - 11 approx. for $k$-median
  - 95 approx. for $k$-means, and

- cluster size constraint is violated by $(p+2)/p$
- replication between $p$ and $p/2$. 
Nearest Neighbor Dispatch
Requirements

• Dispatch a new point correctly and efficiently
Goal

• PAC Assumption: data are drawn iid from some fixed unknown distribution $\mu$
Goal

• Given an iid sample from $\mu$, cluster the distribution

• Balance constraints: Probability mass of each cluster is within $(l,L)$. 
Our solution

• Cluster a sample (previous section)
• Extend clustering to the distribution
• How?
Clustering a Distribution

Assignments \( f : \mathcal{X} \to \binom{k}{p} \)

Centers \( c : [k] \to \mathcal{X} \)

\( K \)-median: \( \min_{f,c} \mathbb{E}_{x \sim \mu} \left[ \sum_{i \in f(x)} \| x - c(i) \| \right] \)
Find the Nearest Center?

- Doesn’t work because of size constraints
An Idea: NN Extension

- Find nearest point from the original sample
An Idea: NN Extension

- Find nearest point from the original sample
An Idea: NN Extension

• Find nearest point from the original sample
NN Extension of a Clustering

\[ \bar{g}_n(x) := g_n(\text{NN}_S(x)) \]

Defined on distribution

Defined on sample

![Diagram showing clustering and extension rules]
NN Extension

• Each point represents its Voronoi cell
• Sample level objective:

\[ g_n : S \rightarrow \binom{k}{p} \]

\[ c_n : [k] \rightarrow S \]

\[
\min_{g_n, c_n} \sum_{j=1}^{n} w_j \left[ \sum_{i \in g_n(x_j)} \|x_j - c_n(i)\| \right] \\
\text{where } w_j = \mathbb{P}_{x \sim \mu}(\text{NN}_S(x) = x_j)
\]
NN Extension

• Weights are unknown
• Estimate weights from another sample drawn iid from $\mu$.
• Cluster sample with estimated weights
• Use approx algo discussed earlier
NN Dispatch Algorithm

- Draw a second sample $S'$ of size $n'$.
- Approximate weights $w_j$ with estimates:
  
  \[
  \hat{w}_j = \frac{|S' \cap V_j|}{n'}
  \]

- Find a balanced clustering $(g_n, c_n)$ using estimated weights
- Return its NN extension
  \[
  \bar{g}_n(x) = g_n(\text{NN}_S(x))
  \]
NN Dispatch

• Guarantee: NN Dispatch returns a good clustering of the distribution.

• Sub-optimality depends on
  – Quality of approximation on sample
  – Average ‘radius’ of Voronoi cell
    \[ \alpha(S) = \mathbb{E}_{x \sim \mu}(\|x - \text{NN}_S(x)\|) \]
  – Bias from returning clustering that are constant over Voronoi partitions
    \[ \beta(S) = \min_{h,c}(Q(\bar{h}, c) - Q(f^*, c^*)) \]
    s.t. \( h \) satisfies size constraints \( l, L \)
NN Dispatch

Theorem:

- If \( n' = O((n + \ln 1/\delta)/\epsilon^2) \)
- Algo on \( S \) returns solution within \( r \cdot \text{OPT} + s \)
- Then \( \text{w.p.} \geq 1 - \delta \)

\(- (\bar{g}_n, c_n) \) output satisfies sizes \( (l - \epsilon, L + \epsilon) \)

\(- Q(\bar{g}_n, c_n) \leq r \cdot Q(f^*, c^*) + s + 2(r + 1)pD\epsilon \\
\quad + p(r + 1) \cdot \alpha(S) \\
\quad + r \cdot \beta(S) \\
\quad (f^*, c^* = \text{OPT}(l + \epsilon, L - \epsilon)) \)
NN Dispatch

• Can bound other terms
• Worst case exponential in dimension
  – Curse of dimensionality
• Better bounds with niceness assumptions
  – E.g., Doubling Measure
Experiments
Learning: Approximations

Balanced Clustering

- K-means++, with rebalancing

NN Dispatch

- Estimated weight = 1/n
- Random Partition Trees for Approximate NN Search
Algorithm

• Cluster a small sample
• Extend the clustering to the rest of the training set with NN Dispatch
• Learn
  – independent model for each cluster or
  – in tandem, with partial or complete communication
• Testing
  – Query the appropriate model with NN Dispatch
Learning

• No communication:
  – Each cluster learns an independent model
  – Embarrassingly parallel

• Compare against:
  – Random partitioning with no communication
  – Random partitioning with full communication (global model)
Experimental Setup

• Run on a cluster with
  – 15 machines
  – 8 cores per machine, each of 2.4GHz
  – 32 GB shared memory per machine
Datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Number of examples</th>
<th>Dimensionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST-8M</td>
<td>8 million</td>
<td>784</td>
</tr>
<tr>
<td>CIFAR-10-early</td>
<td>2.5 million</td>
<td>160</td>
</tr>
<tr>
<td>CIFAR-10-late</td>
<td>2.5 million</td>
<td>144</td>
</tr>
<tr>
<td>CTRc</td>
<td>0.8 million</td>
<td>232</td>
</tr>
<tr>
<td>CTRa</td>
<td>0.3 million</td>
<td>13 million</td>
</tr>
<tr>
<td>Criteo-Kaggle</td>
<td>45 million</td>
<td>34 million</td>
</tr>
</tbody>
</table>

CTR: Click Through Rate
Learning with no communication: MNIST-8M

Accuracy vs. \# of clusters (k)

- Global
- Random
- Ours
Learning with no communication: CIFAR-10

![Graph showing accuracy vs. number of clusters (k) for different methods.](image-url)
Learning with no communication: CTRc

![Graph showing accuracy vs. number of clusters (k)]

- Accuracy:
  - global
  - random
  - ours
  - ours (p=2)
  - ours (p=4)

- # of clusters (k):
  - 4
  - 8
  - 16
  - 32
  - 64
  - 128
  - 256
  - 512
  - 1024
  - 2048
Learning with no communication: Criteo-Kaggle

![Graph showing accuracy vs. number of clusters (k)]
Learning with no communication: Scalability

![Graph showing scalability with different datasets and varying number of workers. The x-axis represents the number of workers (8, 16, 32, 64), and the y-axis represents the time relative to 64 workers. Different datasets are indicated by different markers: Cifar10 in3c (circles), Cifar10 in4d (squares), CTR1S (triangles), and Mnist8m (stars).]
Learning with communication

• High dimensional datasets
• Feature occurrence: approx. power law
Learning with communication

- Tail features cannot be reliably learnt
- Scheme 1: Synchronize on tail features only across all clusters
- Scheme 2: Synchronize on all features, also store a local correction for head features
- Asynchronous Stochastic Gradient Descent
Scheme 1: Partial Communication

• Local model for head and synchronized model for the tail

\[ f(x) = w_{i(x)} \cdot x_h + w_t \cdot x_t \]

• Communication: not very high for relatively small size of head
Scheme 2: Full communication

- Each cluster stores a “correction” to the globally synchronized model

\[ f(x) = w_i(x) \cdot x_h + w_g \cdot x \]

- Communication: Equal to communication of fully synchronized global model
Performance on CTR data

Accuracy for CTRa

- Random
- ours, no comm
- ours, partial comm
- ours, full comm
- global

#clusters(k)

accuracy

0.56
0.58
0.60
0.62

0.56
How many local features?

CTRa, partial communication, k=8

CTRa, partial communication, k=64
Which scheme should I use?

- Dense data, images: No communication
- High dimensional data: With communication
Conclusion

• Data-dependent partitioning is good in both theory and practice!
• Balanced Clustering
• Nearest Neighbor Extension
• Experimental Evaluation
Thank You! Questions?
Collaborators