Distributed Machine Learning

Maria-Florina Balcan
12/09/2015

Machine Learning is Changing the World

"Machine learning is the hot new thing" (John Hennessy, President, Stanford)

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

"Web rankings today are mostly a matter of machine learning" (Prabhakar Raghavan, VP Engineering at Google)

The World is Changing Machine Learning

New applications

Explosion of data

The World is Changing Machine Learning

New approaches, E.g.,
- Semi-supervised learning
- Interactive learning
- Distributed learning
- Multi-task/transfer learning
- Deep Learning
- Never ending learning

Many competing resources & constraints, E.g.,
- Computational efficiency (noise tolerant algos)
- Human labeling effort
- Statistical efficiency
- Communication
- Privacy/Incentives
The World is Changing Machine Learning

New approaches. E.g.,

- Semi-supervised learning
- Interactive learning
- Distributed learning
- Multi-task/transfer learning
- Deep Learning
- Never ending learning

Key challenges: how to best utilize the available resources most effectively in these new settings.

Distributed Learning

Data inherently distributed: **massive amounts** of data **distributed** across multiple locations.

Modern applications: massive amounts of data distributed across multiple locations. E.g.,

- scientific data
- video data

Key new resource communication.

E.g., medical data
**Distributed Learning**

- Data distributed across multiple locations.
- Each has a piece of the overall data pie.
- To learn over the combined $D$, must communicate.
- Communication is expensive.

*President Obama cites Communication-Avoiding Algorithms in FY 2012 Department of Energy Budget Request to Congress*

**Important question: how much communication?**
*Plus, privacy & incentives.*

**Distributed PAC Learning** (Balcan-Blum-Fine-Mansour, COLT 2012)

- $X$ = instance space. $s$ players.
- Player $i$ can sample from $D_i$, samples labeled by $c^*$.
- Goal: find $h$ that approximates $c^*$ w.r.t. $D=1/s \left( D_1 + \ldots + D_s \right)$

**Goal: learn good $h$ over $D$, as little communication as possible**

Efficient algos for problems when centralized algos exist.

**Main Results**
- Broadly applicable communication efficient distr. boosting.
- Tight results for interesting cases [intersection closed, parity fn, linear separators over “nice” distrib].
- Privacy guarantees.

**Interesting special case to think about**

$s=2$. One has the positives and one has the negatives.
- How much communication, e.g., for linear separators?

**Generic Results**

- **Baseline** $d/s \log(1/\varepsilon)$ examples, 1 round of communication

- Each player sends $d/(s \varepsilon) \log(1/\varepsilon)$ examples to player 1.
- Player 1 finds consistent $h \in C$, whp error $\leq \varepsilon$ w.r.t $D$

**Distributed Boosting**

*Only $O(d \log 1/\varepsilon)$ examples of communication*
Communication Aware Distributed Boosting

Baseline
\(d/\varepsilon \log(1/\varepsilon)\) examples, 1 round of communication

- Each player sends \(d/(\varepsilon s) \log(1/\varepsilon)\) examples to player 1.
- Player 1 finds consistent \(h \in C\), whp error \(\leq \varepsilon\) wrt \(D\)

Distributed Boosting

Only \(O(d \log 1/\varepsilon)\) examples of communication!

Recap of Adaboost

- Boosting: algorithmic technique for turning a weak learning algorithm into a strong (PAC) learning one.

Input: \(S=\{(x_1, y_1), \ldots, (x_m, y_m)\}\); weak learner \(A\)

- Weak learning algorithm \(A\).
- For \(t=1,2,\ldots,T\)
  - Construct \(D_t\) on \((x_1,\ldots,x_m)\)
  - Run \(A\) on \(D_t\) producing \(h_t\)
- Output \(H_{\text{final}} = \text{sgn}(\sum \alpha_t h_t)\)

Key points:
- \(H_{\text{final}}(x_i)\) depends on \(h_1(x_i),\ldots,h_T(x_i)\) and normalization factor that can be communicated efficiently.
- To achieve weak learning it suffices to use \(O(d)\) examples.
Distributed Adaboost

- Each player $i$ has a sample $S_i$ from $D_i$.
- For $t=1, 2, \ldots, T$
  - Each player sends player 1, enough data to produce weak hyp $h_t$.
  (For $t=1$, $O(d/s)$ examples each.)
  - Player 1 broadcasts $h_t$ to other players.

- Each player sends player 1, enough data to produce weak hyp $h_t$.
  (For $t=1$, $O(d/s)$ examples each.)

- Player 1 broadcasts $h_t$ to other players.

- Each player $i$ reweights its own distribution on $S_i$ using $h_t$ and sends the sum of its weights $w_{i,t}$ to player 1.

- Player 1 determines the # of samples to request from each $i$ (samples $O(d)$ times from the multinomial given by $w_{i,t}/W_t$).

Distributed Adaboost Can learn any class $C$ with $O(\log(1/\epsilon))$ rounds using $O(d)$ examples + $O(s \log d)$ bits per round.

- Per round: $O(d)$ examples, $O(s \log d)$ extra bits for weights, 1 hypothesis.

Proof:
- As in Adaboost, $O(\log 1/\epsilon)$ rounds to achieve error $\epsilon$.

Distributed implementation of Robust halving \cite{Balcan-Hanneke12}.

Error $O(OPT)+\epsilon$ using only $O(s \log |C| \log(1/\epsilon))$ examples.

Not computationally efficient in general.

Distributed implementation of Smooth Boosting (access to agnostic weak learner). \cite{Tewari-Balcan-Chu15}
Intersection-closed when fns can be described compactly.

C is intersection-closed, then C can be learned in one round and k hypotheses of total communication.

Algorithm:
- Each i draws $S_i$ of size $O(d/\log(1/\epsilon))$, finds smallest $h_i$ in C consistent with $S_i$ and sends $h_i$ to player 1.
- Player 1 computes smallest $h$ s.t. $h_i \subseteq h$ for all $i$.

Key point:
- $h_i, h$ never make mistakes on negatives, so $err_{D_i}(h) \leq err_{D_i}(h_i) \leq \epsilon$.

Interesting class: parity functions

s = 2, $X = \{0,1\}^d$, $C =$ parity fns, $f(x) = x_1 \oplus x_2 \cdots \oplus x_d$

Generic methods: $O(d)$ examples, $O(d^2)$ bits.

Classic CC lower bound: $\Omega(d^2)$ bits LB for proper learning.

Improperly learn $C$ with $O(d)$ bits of communication!

Algorithm:
- Can properly PAC-learn $C$.
- Can non-properly learn $C$ in reliable-useful manner (KS88)

Key points:
- Can properly PAC-learn $C$.

Improperly learn $C$ with $O(d)$ bits of communication!

Algorithm:
- Player i properly PAC-learns over $D_i$ to get parity $h_i$. Also improperly R-U learns to get rule $g_i$. Sends $h_i$ to player j.
- Player i uses rule $R_i$: "if $g_i$ predicts, use it; else use $h_j$".

Key point: low error under $D_i$ because $h_i$ has low error under $D_i$ and since $g_i$ never makes a mistake putting it in front does not hurt.
**Distributed PAC learning: Summary**
- First time consider communication as a fundamental resource.
- Broadly applicable communication efficient distributed boosting.
- Improved bounds for special classes (intersection-closed, parity fns, and linear separators over nice distributions).
- Analysis of privacy guarantees achievable.
- Lots of follow-up work analyzing communication aspects in ML.

---

**Privacy**
Natural also to consider privacy in this setting.
- Privacy for individual data items (using usual notion of differential privacy considered in the literature).
- Privacy for the data holders / players (using a notion of distributional privacy).

Q: What is the effect on communication?

---

**Differential Privacy**
Differential privacy: want each player i's messages not to reveal information about individual data items in $S_i$.

- For any $x \in S_i$, prob of output sequence $\sigma$ changes by only a little if modify $x$ to any $x'$.

$$\forall \sigma, \frac{\Pr(A(S_i)=\sigma)}{\Pr(A(S_i\setminus x+x')=\sigma)} \in 1 \pm \epsilon$$

- Substantial literature on how to achieve - e.g., any Stat. Query algorithm can be made to satisfy Diff. Privacy.
- So, if protocols can be implemented s.t. each player interacts with own data via SQs, then no increase in communication.

---

**Distributed Clustering**

[Balcan-Ehrlich-Liang, NIPS 2013]
[Balcan-Kanchanapally-Liang-Woodruff, NIPS 2014]
**Distributed Clustering** [Balcan-Ehrlich-Liang, NIPS 2013]

- **k-median**: find center pts $c_1, c_2, \ldots, c_k$ to minimize $\sum x \min d(x, c_i)$
- **k-means**: find center pts $c_1, c_2, \ldots, c_k$ to minimize $\sum d(x, c_i)^2$

**Goal**: cluster the data, as little communication as possible

- Dataset $S$ distributed across $s$ locations.
- Each has a piece of the overall data pie.

**Key idea**: use coresets, short summaries capturing relevant info w.r.t. all clusterings.

- By combining local coresets, get a global coreset; the size goes up multiplicatively by $s$.
- We show a two round procedure with communication only the true size of a global coreset of dataset $S$.

---

**Coresets**

**Def**: An $\epsilon$-coreset for a set of pts $S$ is a set of points $\tilde{S}$ and weights $w: \tilde{S} \rightarrow \mathbb{R}$ s.t. for any sets of centers $c$:

$$(1 - \epsilon) \text{cost}(S, c) \leq \sum_{p \in \tilde{S}} w_p \text{cost}(p, c) \leq (1 + \epsilon) \text{cost}(S, c)$$

**Centralized Coresets of size $O(kd/\epsilon^2)$** [Feldman-Langberg'11]

1. Find a constant factor approx. $B$, add its centers to coreset
2. Sample $O(kd/\epsilon^2)$ pts according to their contribution to the cost of that approximate clustering $B$. Add them in too.

**Key idea (proof reinterpreted)**:

- Can view $B$ as rough coreset, with $b \in B$ weighted by size of Voronoi cell.
- If $p$ has closest pt $b_p \in B$, then for any center $c$, $|\text{cost}(p, c) - \text{cost}(b_p, c)| \leq ||p - b_p||$ by triangle inequality.
- So, penalty $(p) = \text{cost}(p, c) - \text{cost}(b_p, c)$ for $p$ satisfies $(p) \in [-\text{cost}(p, b_p), \text{cost}(p, b_p)]$.
- Motivates sampling according to $\text{cost}(p, b_p)$. 

---

**Distributed Clustering** [Balcan-Ehrlich-Liang, NIPS 2013]

- Data distributed across $s$ locations.
- Each has a piece of the overall data pie.

**Goal**: cluster the data, as little communication as possible
Distributed Clustering

Key fact: $S_i$ is a coreset for $X_i$, then $\bigcup S_i$ is a coreset for $\bigcup X_i$.

1. Each player finds a coreset of size $O(kd/e^2)$ on their own data using centralized method.
2. Then they all send local coresets to the center.

For $s$ players, total communication is $O(skd/e^2)$.

Can we do better?

Distributed Coresets [Balcan-Ehrlich-Liang, NIPS 2013]

Key idea: in distributed case, show how to do this using only local constant factor approx.

1. Each player $i$ finds a local constant factor approx. $B_i$, and sends cost($B_i, P_i$) and the centers to the center.
2. Center samples $n = O(kd/e^2)$ times $n_i = n_1 + \cdots + n_s$, from multinomial given by these costs. Sends $n_i$ to player $i$.
3. Each player $i$ sends $n_i$ points from $P_i$ sampled according to their contribution to the local approx.

For $s$ players, total communication is only $O \left( \frac{sk}{n} + sk \right)$.

Open questions (Learning and Clustering)

- Efficient algorithms in noisy settings; handle failures, delays.
- Even better dependence on $1/e$ for communication efficiency for clustering via boosting style ideas.
  - Can use distributed dimensionality reduction to reduce dependence on $d$. [Balcan-Kanchanapally-Liang-Woodruff, NIPS 2014]
- More refined trade-offs between communication complexity, computational complexity, and sample complexity.