# Para-active learning

Alekh Agarwal
Microsoft Research

Joint work with Léon Bottou, Miroslav Dudík and John Langford

# Motivation

- Many existing distributed learning approaches
  - Parallelize existing algorithms (e.g. distributed optimization)
  - Variants of existing algorithms (e.g. distributed mini-batches)
  - Bagging, model averaging, ...

# Motivation

- Many existing distributed learning approaches
  - Parallelize existing algorithms (e.g. distributed optimization)
  - Variants of existing algorithms (e.g. distributed mini-batches)
  - Bagging, model averaging, ...
- Model/gradients cheaply communicated, meaningfully averaged

# Motivation

- Many existing distributed learning approaches
  - Parallelize existing algorithms (e.g. distributed optimization)
  - Variants of existing algorithms (e.g. distributed mini-batches)
  - Bagging, model averaging, ...
- Model/gradients cheaply communicated, meaningfully averaged
- Limited use of the statistical problem structure (beyond i.i.d.)

# Peculiarities of models

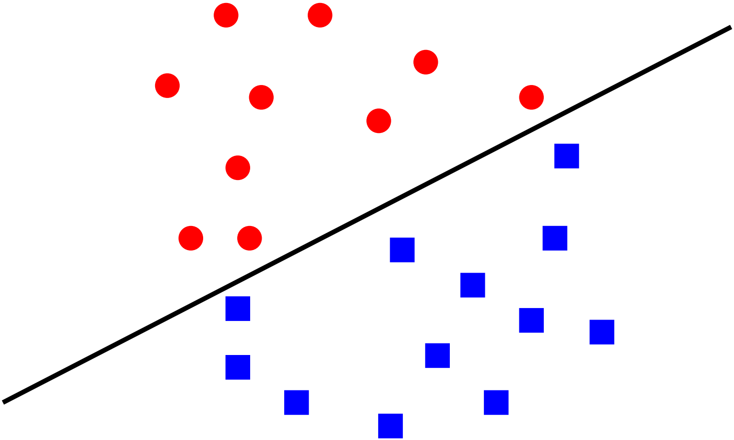
- Models not always parsimoniously described
  - Kernel methods: model/gradient not described without training data
  - High-dimensional/non-parametric models

# Peculiarities of models

- Models not always parsimoniously described
  - Kernel methods: model/gradient not described without training data
  - High-dimensional/non-parametric models
- Models not always meaningfully averaged
  - Matrix factorization: $M = UV = (-U)(-V)$
  - More generic for non-convex models: neural networks, mixture models

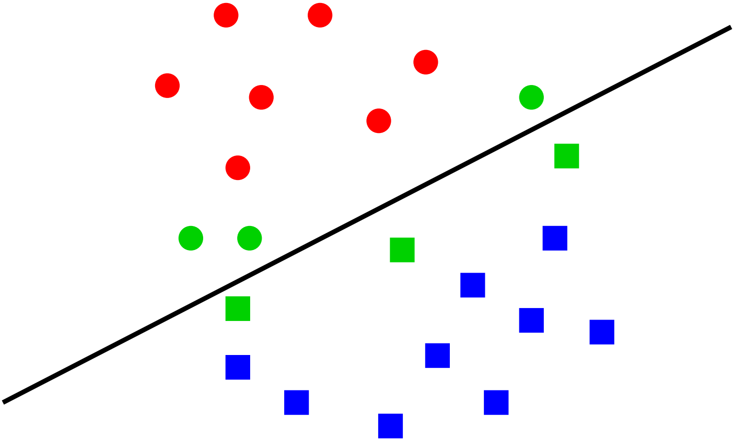
# Data data everywhere, but ...

- Not all data points are equally informative



# Data data everywhere, but . . .

- Not all data points are equally informative
- Small number of support vectors specify SVM solution



# Active learning

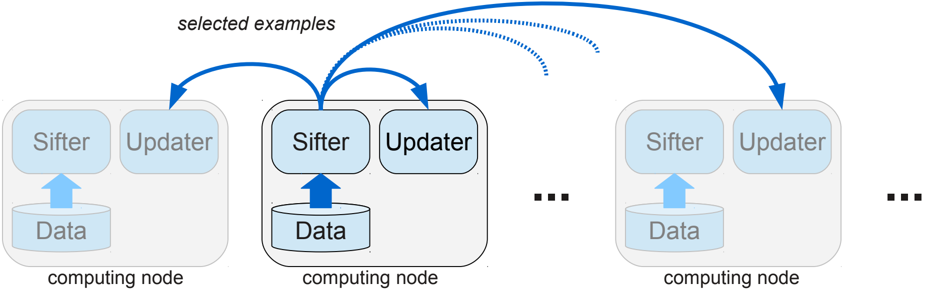
- Active learning identifies *informative examples*
- Similar idea as support vectors, works more generally
- Efficient algorithms (and heuristics) for typical hypothesis classes

# Active learning

- Active learning identifies *informative examples*
- Similar idea as support vectors, works more generally
- Efficient algorithms (and heuristics) for typical hypothesis classes
- Examples
  - Query $x$ with probability $g(|h(x)|)$
  - Query $x$ based on similarity with previously queried samples

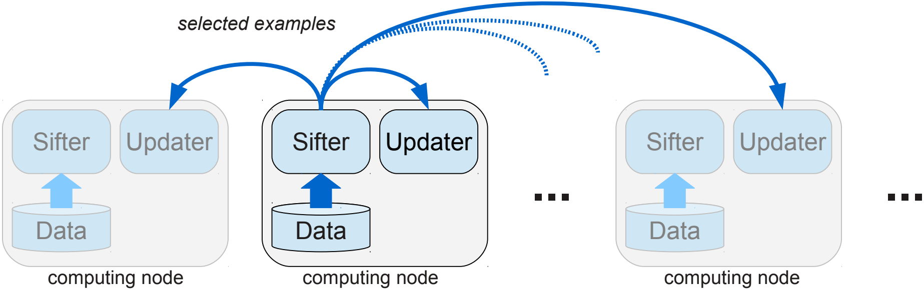
# Para-active learning

- Sift for informative examples in parallel
- Update model on selected examples



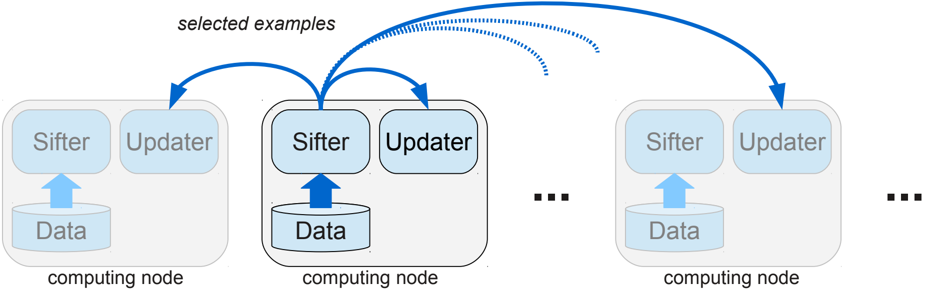
# Synchronous para-active learning

- Initial hypothesis $h_1$, batch size $B$, active sifter $\mathcal{A}$, passive updater $\mathcal{P}$
- For rounds $t = 1, 2, \ldots, T$
  - For all nodes $i = 1, 2, \ldots, k$ *in parallel*
    - Local dataset of size $B/k$
    - $\mathcal{A}$ creates subsampled dataset



# Synchronous para-active learning

- Initial hypothesis $h_1$, batch size $B$, active sifter $\mathcal{A}$, passive updater $\mathcal{P}$
- For rounds $t = 1, 2, \ldots, T$
  - For all nodes $i = 1, 2, \ldots, k$ *in parallel*
    - Local dataset of size $B/k$
    - $\mathcal{A}$ creates subsampled dataset
  - Collect subsampled datasets from each node
  - Update $h_{t+1}$ by running passive updater $\mathcal{P}$ on the collected data

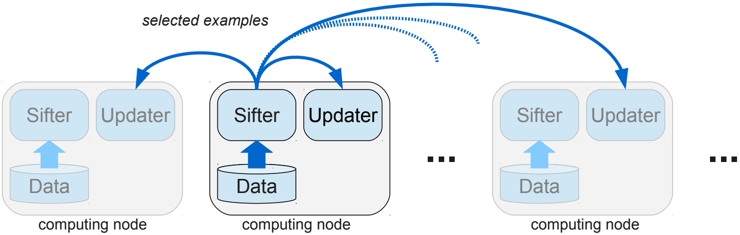


# Synchronous para-active learning

- Initial hypothesis $h_1$, batch size $B$, active sifter $\mathcal{A}$, passive updater $\mathcal{P}$
- For rounds $t = 1, 2, \ldots, T$
  - For all nodes $i = 1, 2, \ldots, k$ *in parallel*
    - Local dataset of size $B/k$
    - $\mathcal{A}$ creates subsampled dataset
  - Collect subsampled datasets from each node
  - Update $h_{t+1}$ by running passive updater $\mathcal{P}$ on the collected data
- Example
  - $h_t$ is kernel SVM on examples selected so far
  - $\mathcal{A}$ samples based on $g(|h_t(x)|)$ at round $t$
  - $\mathcal{P}$ computes $h_{t+1}$ from $h_t$ using online kernel SVM

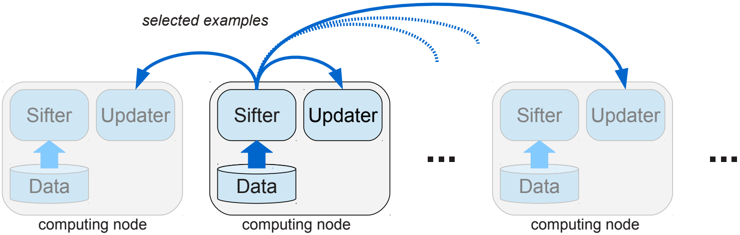
# Asynchronous para-active learning

- Initial hypothesis $h_1$, batch size $B$, active sifter $\mathcal{A}$, passive updater $\mathcal{P}$
- Initialize $Q_S^i = \emptyset$ for each node $i$
- For all nodes $i = 1, 2, \ldots, k$ *in parallel*
  - While $Q_S^i$ is not empty
    - Fetch a selected example from $Q_S^i$
    - Update the hypothesis using $\mathcal{P}$ on this example



# Asynchronous para-active learning

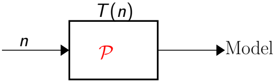
- Initial hypothesis $h_1$, batch size $B$, active sifter $\mathcal{A}$, passive updater $\mathcal{P}$
- Initialize $Q_S^i = \emptyset$ for each node $i$
- For all nodes $i = 1, 2, \ldots, k$ *in parallel*
  - While $Q_S^i$ is not empty
    - Fetch a selected example from $Q_S^i$
    - Update the hypothesis using $\mathcal{P}$ on this example
  - If $Q_F^i$ is non-empty
    - Fetch a candidate example from $Q_F^i$
    - Use $\mathcal{A}$ to decide whether the example is selected or not
    - If selected, broadcast example for addition to $Q_S^j$ for all $j$



# Computational complexity

- Training time for $n$ examples: $T(n)$
- Evaluation time *per example* after $n$ examples: $S(n)$
- Number of subsampled examples out of $n$: $\phi(n)$
- Number of nodes: $k$

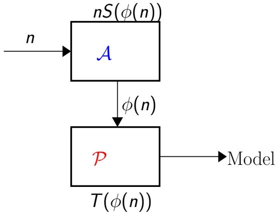
| | **Seq. Passive** | **Seq. Active** | **Para-active** |
|---|---|---|---|
| **Operations** | $T(n)$ | | |
| **Time** | $T(n)$ | | |
| **Broadcasts** | 0 | | |



# Computational complexity

- Training time for $n$ examples: $T(n)$
- Evaluation time *per example* after $n$ examples: $S(n)$
- Number of subsampled examples out of $n$: $\phi(n)$
- Number of nodes: $k$

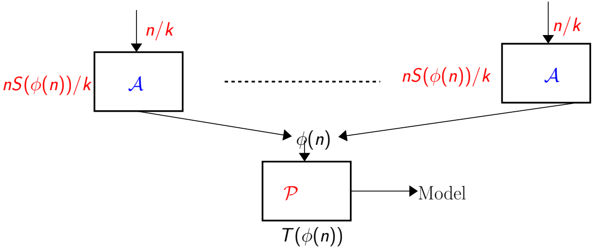
| | **Seq. Passive** | **Seq. Active** | **Para-active** |
|---|---|---|---|
| **Operations** | $T(n)$ | $nS(\phi(n)) + T(\phi(n))$ | |
| **Time** | $T(n)$ | $nS(\phi(n)) + T(\phi(n))$ | |
| **Broadcasts** | 0 | 0 | |



# Computational complexity

- Training time for $n$ examples: $T(n)$
- Evaluation time *per example* after $n$ examples: $S(n)$
- Number of subsampled examples out of $n$: $\phi(n)$
- Number of nodes: $k$

| | **Seq. Passive** | **Seq. Active** | **Para-active** |
|---|---|---|---|
| **Operations** | $T(n)$ | $nS(\phi(n)) + T(\phi(n))$ | $nS(\phi(n)) + kT(\phi(n))$ |
| **Time** | $T(n)$ | $nS(\phi(n)) + T(\phi(n))$ | $nS(\phi(n))/k + T(\phi(n))$ |
| **Broadcasts** | 0 | 0 | $\phi(n)$ |



# Computational complexity

- Training time for $n$ examples: $T(n)$
- Evaluation time *per example* after $n$ examples: $S(n)$
- Number of subsampled examples out of $n$: $\phi(n)$
- Number of nodes: $k$

| | **Seq. Passive** | **Seq. Active** | **Para-active** |
|---|---|---|---|
| **Operations** | $T(n)$ | $nS(\phi(n)) + T(\phi(n))$ | $nS(\phi(n)) + kT(\phi(n))$ |
| **Time** | $T(n)$ | $nS(\phi(n)) + T(\phi(n))$ | $nS(\phi(n))/k + T(\phi(n))$ |
| **Broadcasts** | 0 | 0 | $\phi(n)$ |

**Example 1, kernel SVM:**

- $T(n) \sim \mathcal{O}(n^2)$, $S(n) \sim \mathcal{O}(n)$
- Often $\phi(n) \ll n$
- $T(n) \gg nS(\phi(n)) \gg nS(\phi(n))/k$

# Computational complexity

- Training time for $n$ examples: $T(n)$
- Evaluation time *per example* after $n$ examples: $S(n)$
- Number of subsampled examples out of $n$: $\phi(n)$
- Number of nodes: $k$

| | **Seq. Passive** | **Seq. Active** | **Para-active** |
|---|---|---|---|
| **Operations** | $T(n)$ | $nS(\phi(n)) + T(\phi(n))$ | $nS(\phi(n)) + kT(\phi(n))$ |
| **Time** | $T(n)$ | $nS(\phi(n)) + T(\phi(n))$ | $nS(\phi(n))/k + T(\phi(n))$ |
| **Broadcasts** | 0 | 0 | $\phi(n)$ |

**Example 2, neural nets with backprop:**

- $T(n) \sim \mathcal{O}(nd)$, $S(n) \sim \mathcal{O}(d)$
- Often $\phi(n) \ll n$
- $T(n) \approx nS(\phi(n)) \gg nS(\phi(n))/k$

# Communication complexity

- Communication complexity is query complexity of active learning
- Typically assume examples are queried immediately in active learning
- We have a delay before the model is updated
- **Theorem:** Delay of $\tau$ leads to query complexity at most $\tau + \phi(n - \tau)$

# Experimental evaluation

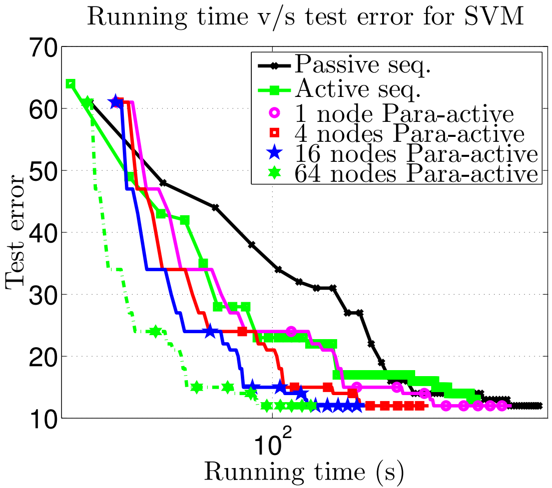
- Large version of MNIST (8.1M examples) with elastic deformations of original images
- Two learning algorithms:
  - Simulation for kernel SVM: RBF kernel, LASVM algorithm
  - Parallel neural nets: 1-hidden layer with 100 nodes
- Active learning: select a point $x$ with probability based on $|f(x)|$ for fixed subsampling rate

# Kernel SVM simulation

- Simulated synchronous para-active learning
- Fixed batch size $B$, split into portions of size $B/k$
- Sift each portion in turn, take largest sifting time
- Update model with new examples, take training time
- Used as an estimate of parallel computation time

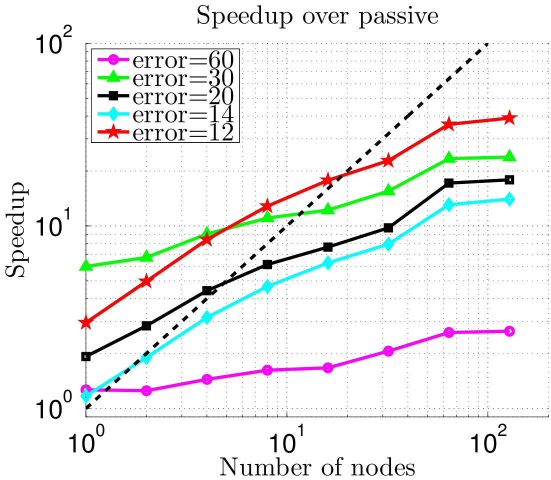
# SVM simulation runtimes

- Classifying $\{3, 1\}$ vs $\{5, 7\}$
- Running time vs test error



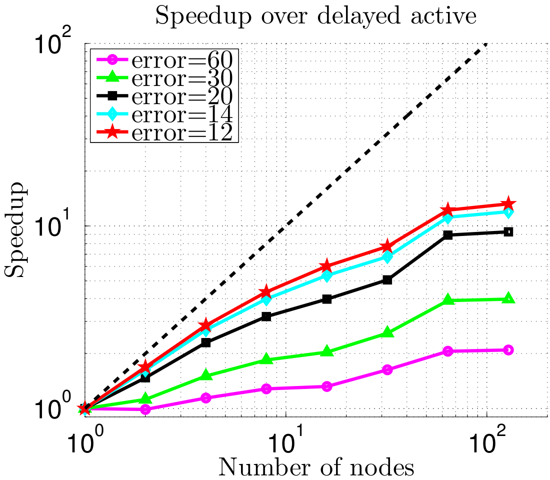
# SVM simulated speedup over passive

- Classifying $\{3, 1\}$ vs $\{5, 7\}$
- Speedup over sequential passive



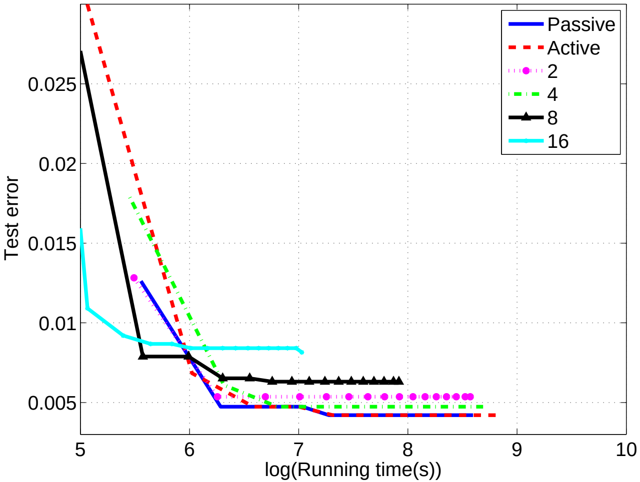
# SVM simulated speedup over delayed active

- Classifying $\{3, 1\}$ vs $\{5, 7\}$
- Speedup over delayed active



# Parallel neural net results

- Classifying 3 vs 5
- Running time vs test error



# Conclusions

- General strategy for distributed learning
- Applicable to diverse hypothesis classes and algorithms
- Particularly appealing for non-parametric and/or non-convex models
- Theoretically justified, empirically promising

# Conclusions

- General strategy for distributed learning
- Applicable to diverse hypothesis classes and algorithms
- Particularly appealing for non-parametric and/or non-convex models
- Theoretically justified, empirically promising
- Real distributed implementation for kernel SVMs
- Other algorithms and datasets
- Better subsampling strategies