Large Scale Learning

Data hypergrowth: an example

- Reuters-21578: about 10K docs (ModApte)

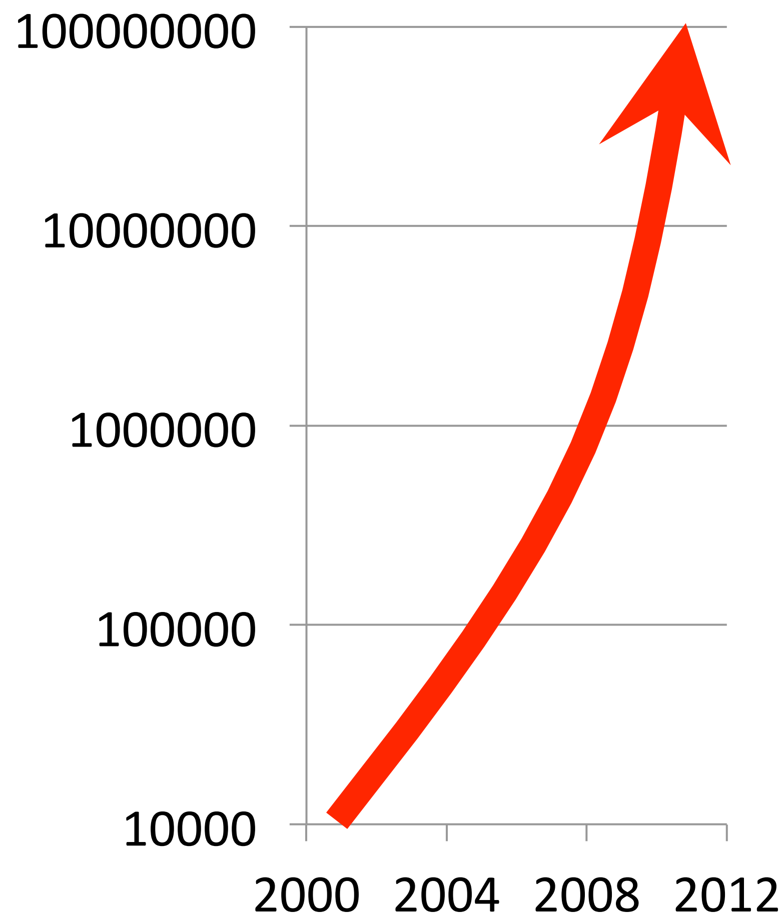
  Bekkerman et al, SIGIR 2001

- RCV1: about 807K docs

  Bekkerman & Scholz, CIKM 2008

- LinkedIn job title data: about 100M docs

  Bekkerman & Gavish, KDD 2011



New age of big data

- The world has gone mobile
  - 5 billion cellphones produce daily data
- Social networks have gone online
  - Twitter produces 200M tweets a day
- Crowdsourcing is the reality
  - Labeling of 100,000+ data instances is doable
    - Within a week ☺

Size matters

- One thousand data instances
- One million data instances
- One billion data instances
- One trillion data instances

Those are not different numbers,

those are different mindsets ☺

One million data instances

- Currently, the most active zone
- Can be crowdsourced
- Can be processed by a quadratic algorithm
  - Once parallelized
- 1M data collection cannot be too diverse
  - But can be too homogenous
- Preprocessing / data probing is crucial

Big dataset cannot be too sparse

- 1M data instances cannot belong to 1M classes
  - Simply because it's not practical to have 1M classes ☺
- Here's a statistical experiment, in text domain:
  - 1M documents
  - Each document is 100 words long
  - Randomly sampled from a unigram language model
    - No stopwords
  - **245M pairs have word overlap of 10% or more**
- Real-world datasets are denser than random

One billion data instances

- Web-scale
- Guaranteed to contain data in different formats
  - ASCII text, pictures, javascript code, PDF documents…
- Guaranteed to contain (near) duplicates
- Likely to be badly preprocessed ☺
- Storage is an issue

One trillion data instances

- Beyond the reach of the modern technology
- Peer-to-peer paradigm is (arguably) the only way to process the data
- Data privacy / inconsistency / skewness issues
  - Can't be kept in one location
  - Is intrinsically hard to sample

Not enough (clean) training data?

- Use existing labels as a *guidance* rather than a directive
  - In a semi-supervised clustering framework
- Or label more data! ☺
  - With a little help from the crowd

Crowdsourcing labeled data

- Crowdsourcing is a tough business ☺
  - People are not machines
- Any worker who can game the system ***will*** game the system
- Validation framework + qualification tests are a must
- Labeling a lot of data can be fairly expensive

Let’s talk about how we can learn with datasets this large...

Stochastic Gradient Descent

Consider Learning with Numerous Data

- Logistic regression objective:

$$J(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n}\underbrace{\left[y_i \log h_{\boldsymbol{\theta}}(\mathbf{x}_i) + (1-y_i)\log\left(1-h_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)\right]}_{\mathrm{cost}_{\boldsymbol{\theta}}(\mathbf{x}_i,\, y_i)}$$

- Fit via gradient descent:

$$\theta_j \leftarrow \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\mathbf{x}_i\right) - y_i\right)x_{ij}$$

- What is the computational complexity in terms of n?

Gradient Descent

Batch Gradient Descent

Initialize θ

Repeat {

$$\theta_j \leftarrow \theta_j - \alpha \underbrace{\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\mathbf{x}_i\right) - y_i\right) x_{ij}}_{\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})} \qquad \text{for } j = 0 \dots d$$

}

Stochastic Gradient Descent

Initialize θ

Randomly shuffle dataset

Repeat { (Typically 1 – 10x)

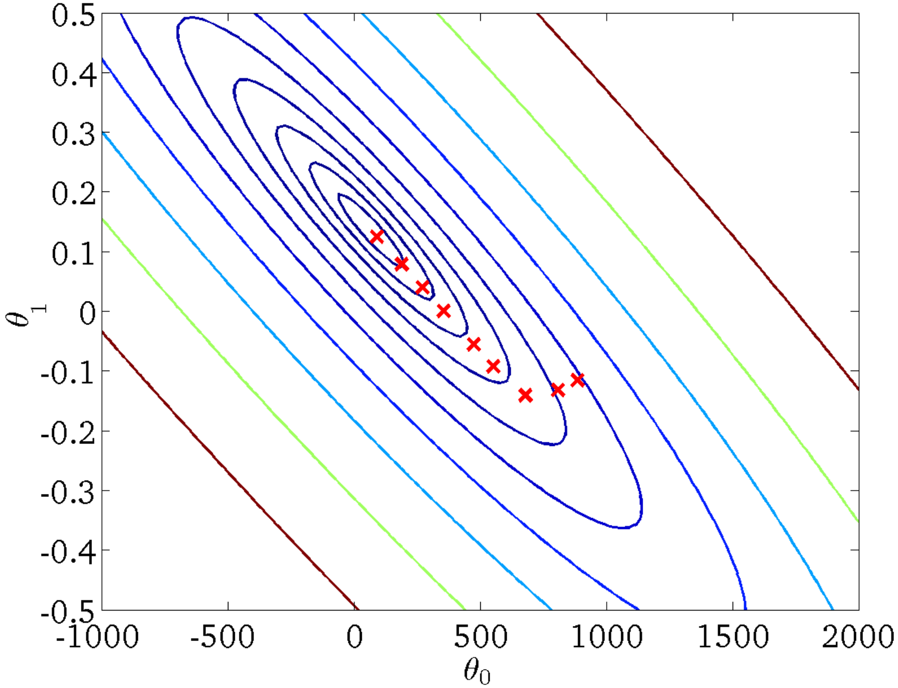
For $i = 1 \dots n$, *do*

$$\theta_j \leftarrow \theta_j - \alpha \underbrace{\left(h_{\boldsymbol{\theta}}\left(\mathbf{x}_i\right) - y_i\right) x_{ij}}_{\frac{\partial}{\partial \theta_j} \mathrm{cost}_{\boldsymbol{\theta}}(\mathbf{x}_i, y_i)} \qquad \text{for } j = 0 \dots d$$

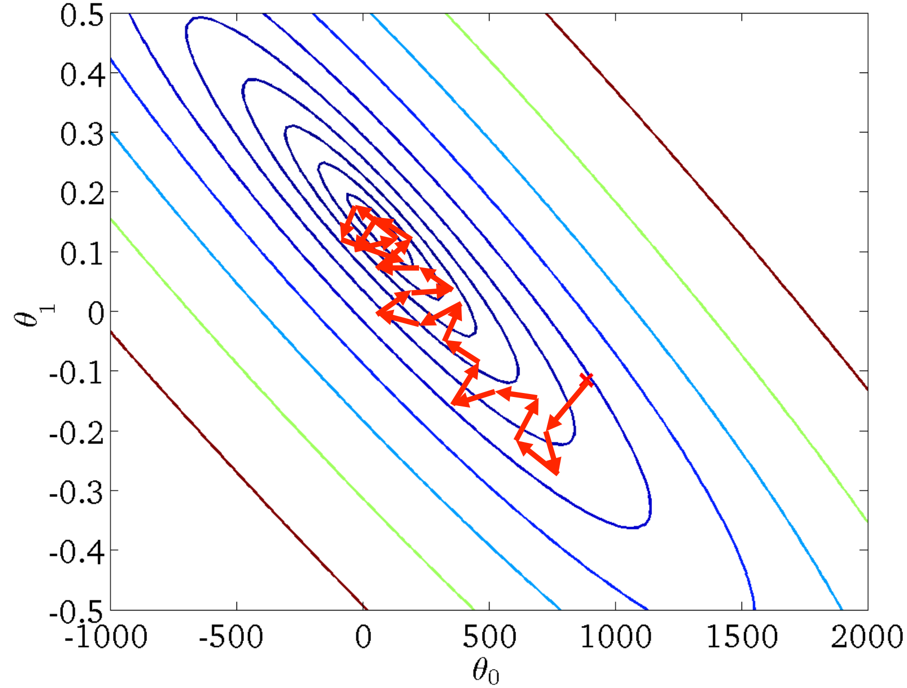
}

Batch vs Stochastic GD

Batch GD



Stochastic GD

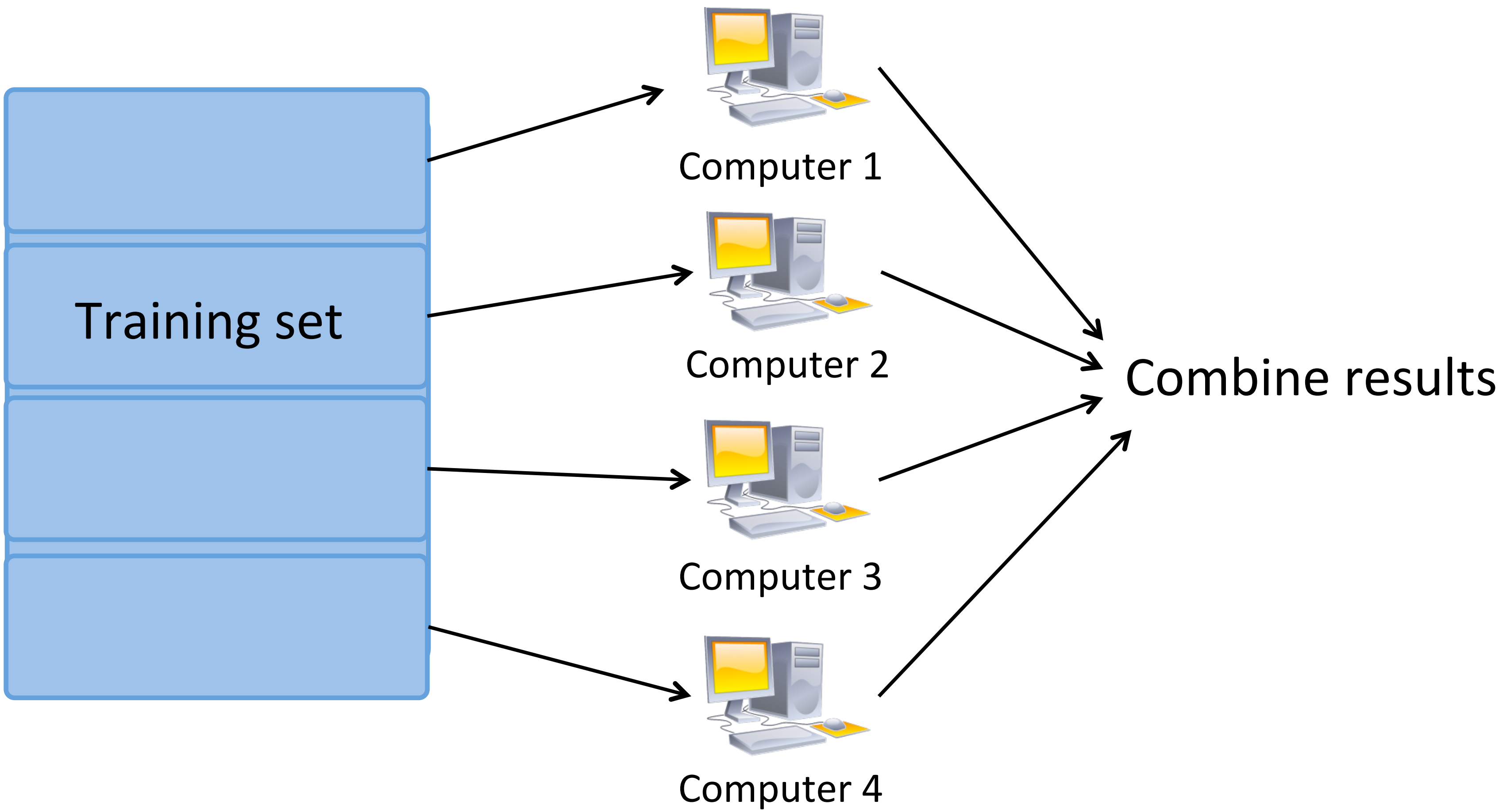


- Learning rate α is typically held constant
- Can slowly decrease α over time to force θ to converge:

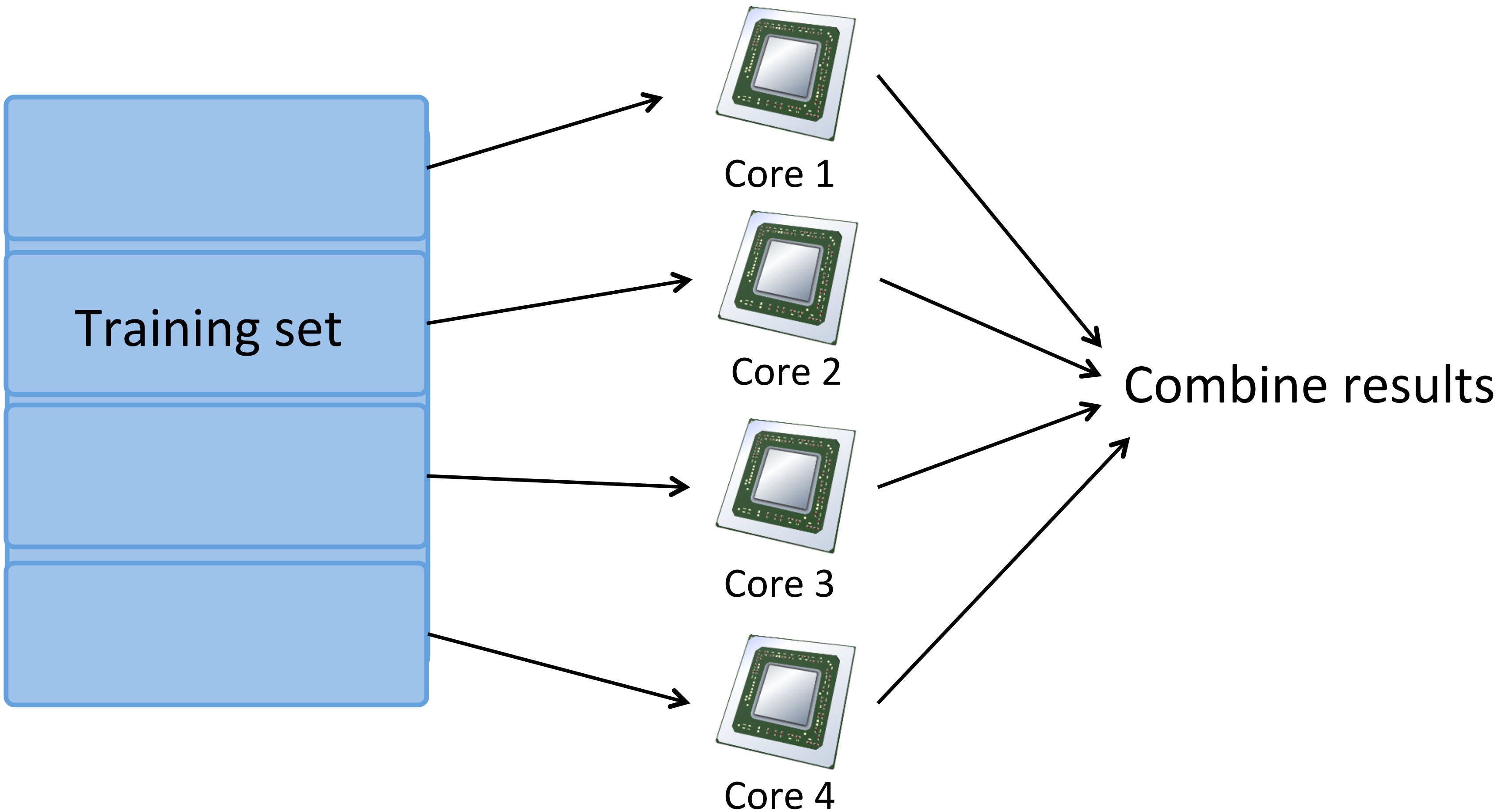
e.g., $$\alpha = \frac{\text{constant1}}{\text{iterationNumber} + \text{constant2}}$$

Graph- and Data-Parallelism

Map-Reduce


Training set
Computer 1
Computer 2
Computer 3
Computer 4
Combine results

Multi-Core Machines



Map-Reduce for Batch GD

Split dataset up into chunks (e.g., with $n = 400$) to compute $\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\mathbf{x}_i \right) - y_i \right) x_{ij}$

$\text{temp1} = \sum_{i=1}^{100} \left(h_{\boldsymbol{\theta}} \left(\mathbf{x}_i \right) - y_i \right) x_{ij}$

$(\mathbf{x}_1, \mathrm{y}_1)$ … $(\mathbf{x}_{100}, \mathrm{y}_{100})$

$\text{temp2} = \sum_{i=101}^{200} \left(h_{\boldsymbol{\theta}} \left(\mathbf{x}_i \right) - y_i \right) x_{ij}$

$(\mathbf{x}_{101}, \mathrm{y}_{101})$ … $(\mathbf{x}_{200}, \mathrm{y}_{200})$

$\text{temp3} = \sum_{i=201}^{300} \left(h_{\boldsymbol{\theta}} \left(\mathbf{x}_i \right) - y_i \right) x_{ij}$

$(\mathbf{x}_{201}, \mathrm{y}_{201})$ … $(\mathbf{x}_{300}, \mathrm{y}_{300})$

$(\mathbf{x}_{301}, \mathrm{y}_{301})$ … $(\mathbf{x}_{400}, \mathrm{y}_{400})$

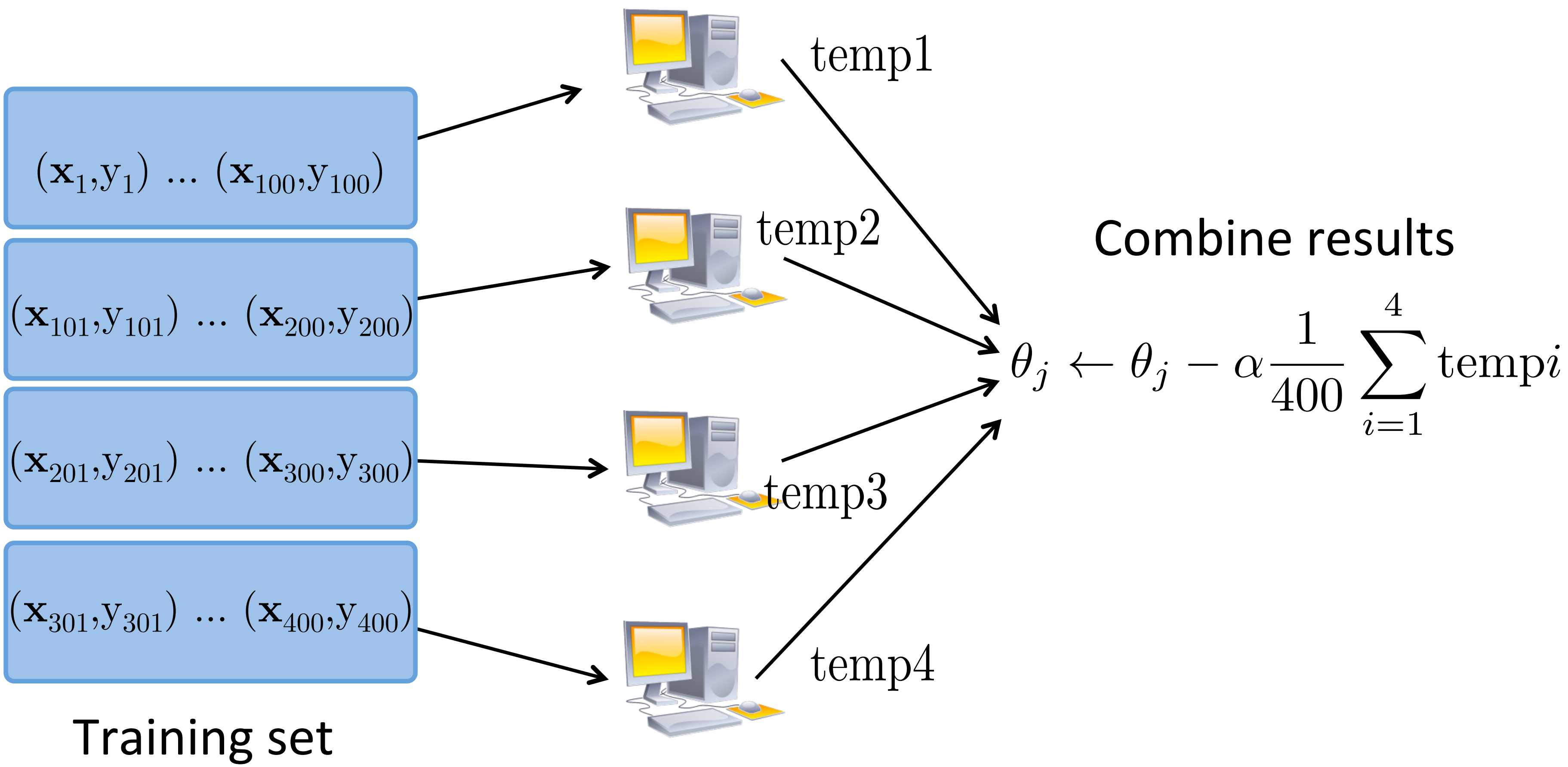
$\text{temp4} = \sum_{i=301}^{400} \left(h_{\boldsymbol{\theta}} \left(\mathbf{x}_i \right) - y_i \right) x_{ij}$

Training set

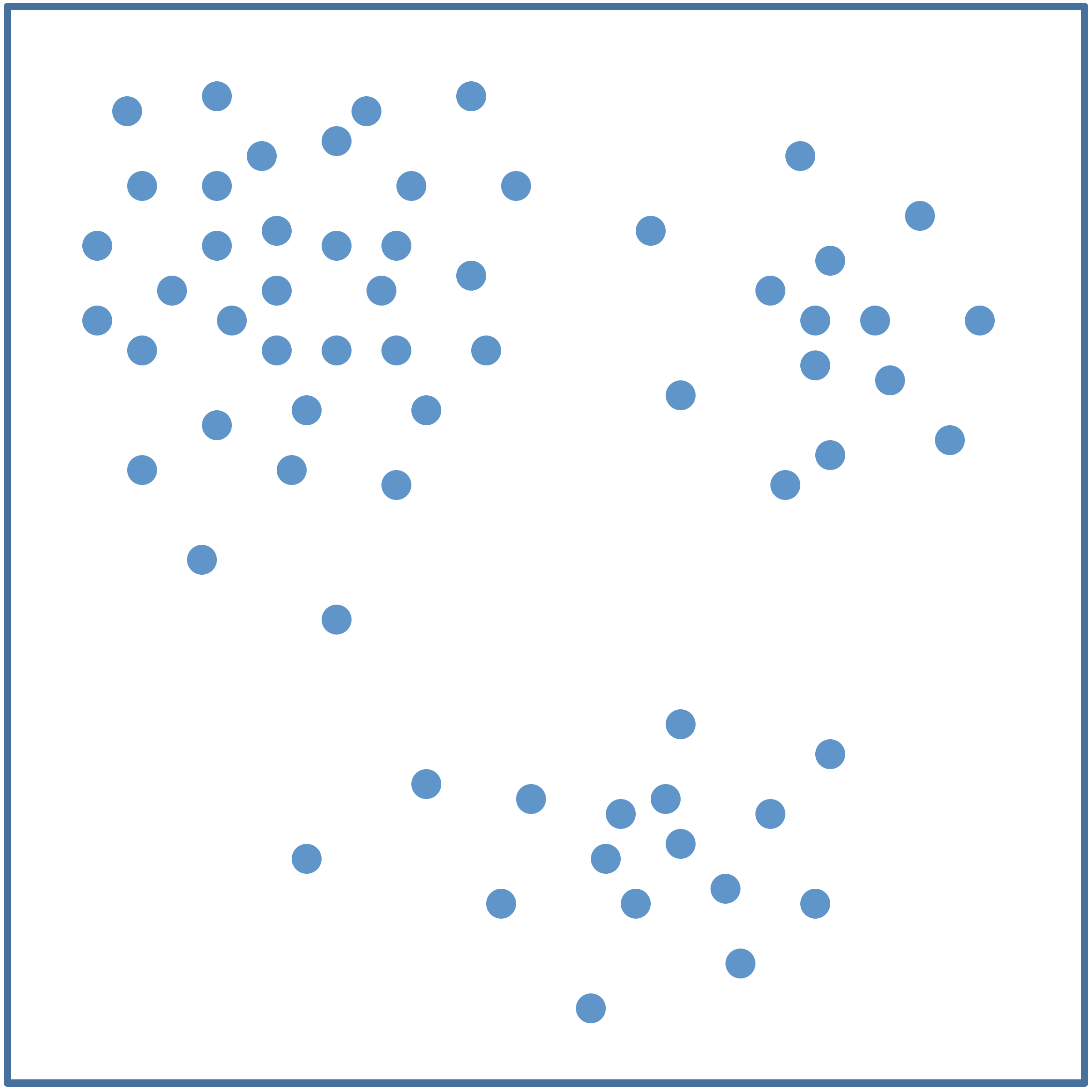
Based on example by Andrew Ng

Map-Reduce for Batch GD

Split dataset up into chunks (e.g., with $n = 400$) to compute $\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\mathbf{x}_i \right) - y_i \right) x_{ij}$

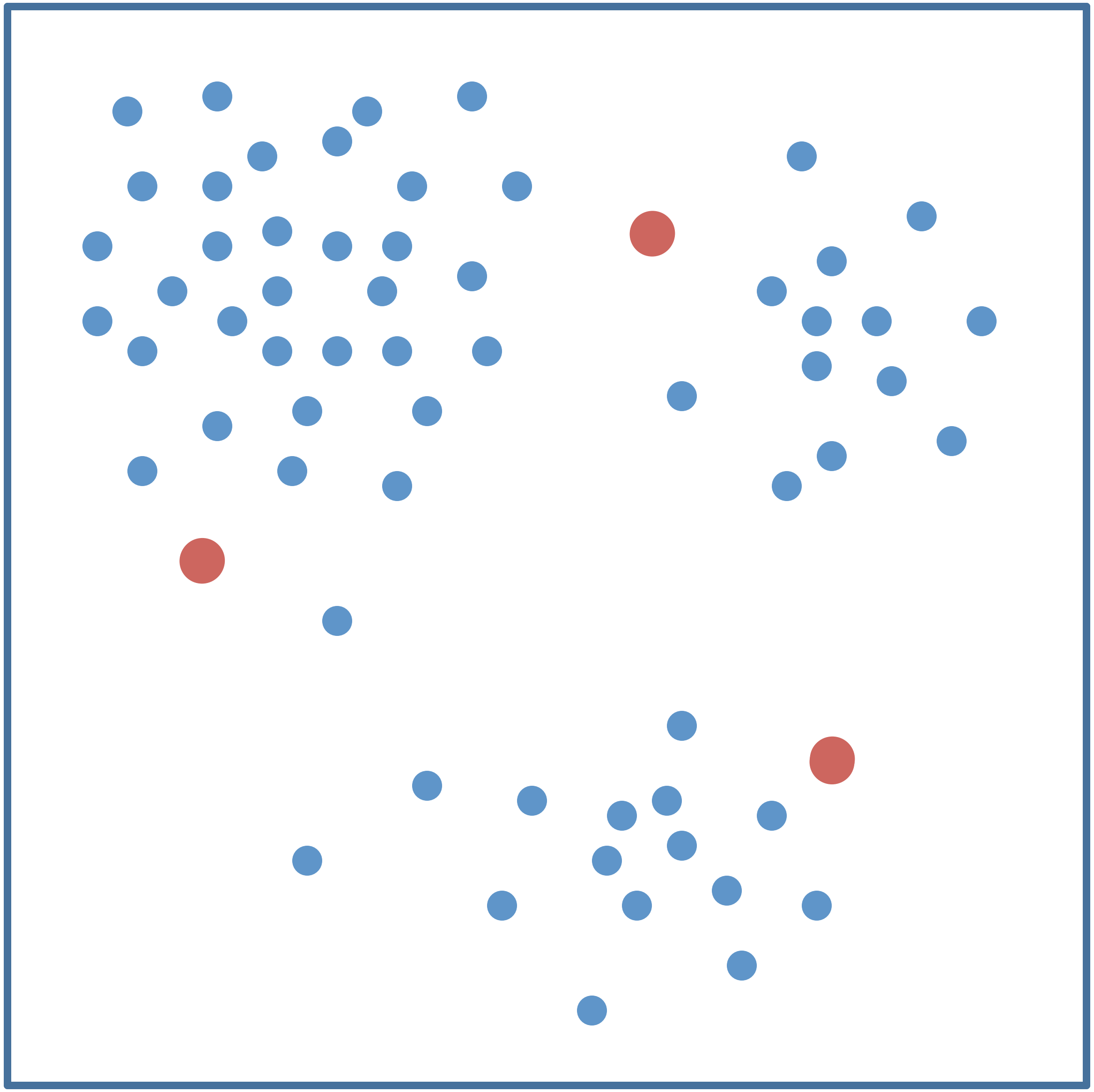


Parallelizing *k*-means



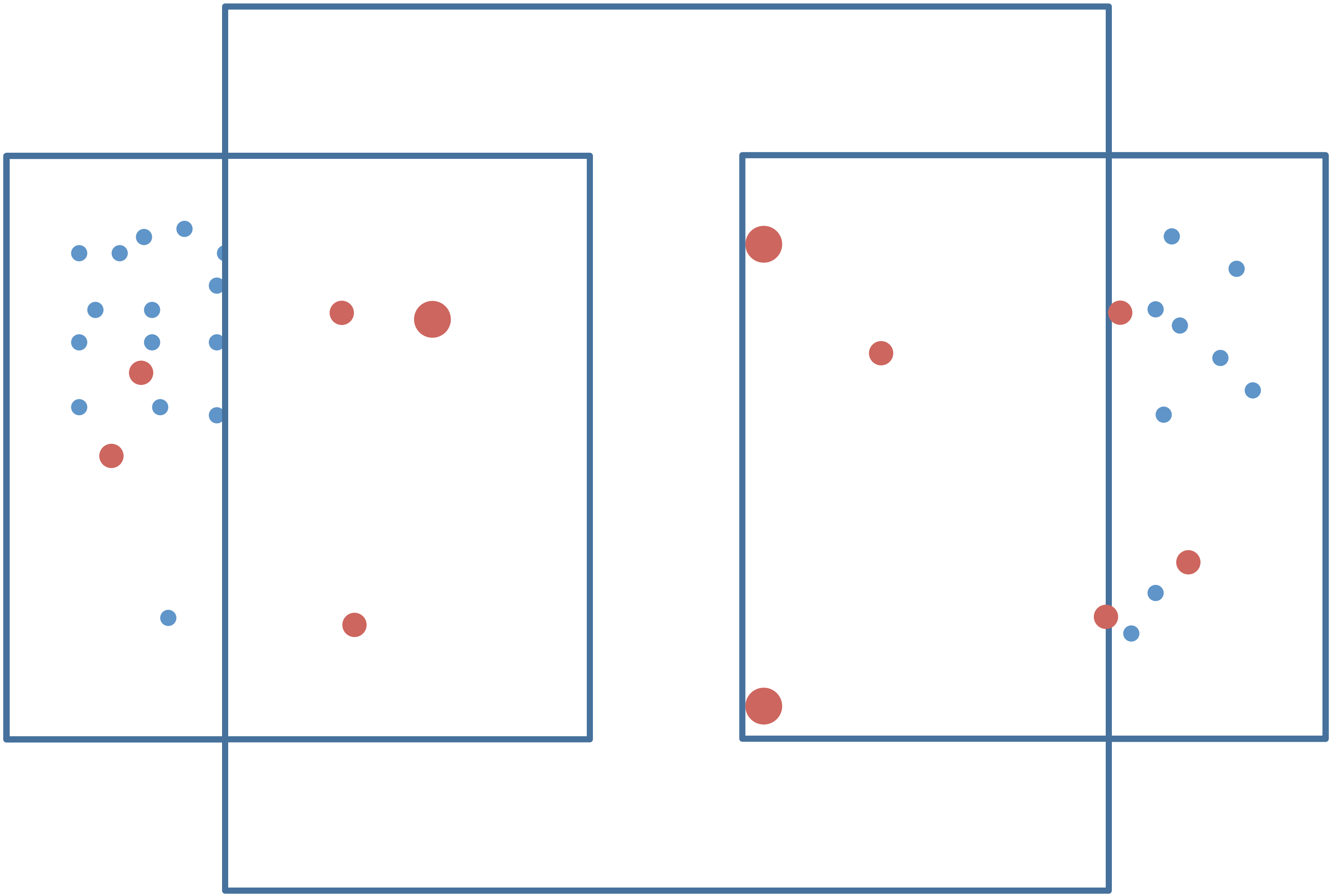
Slide by R. Bekkerman, M. Bilenko, J. Langford

Parallelizing k-means



Slide by R. Bekkerman, M. Bilenko, J. Langford

Parallelizing *k*-means



Slide by R. Bekkerman, M. Bilenko, J. Langford

k-means on MapReduce

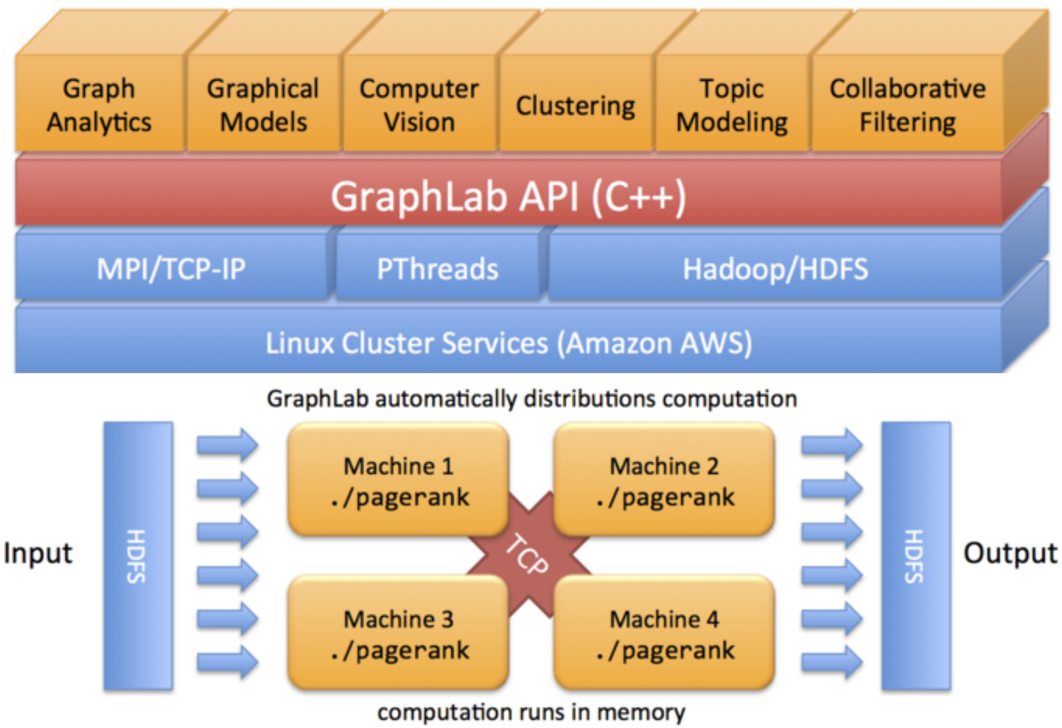
- Mappers read data portions and centroids
- Mappers **assign data instances** to clusters
- Mappers **compute new local centroids** and local cluster sizes
- Reducers **aggregate local centroids** (weighted by local cluster sizes) into new global centroids
- Reducers **write the new centroids**

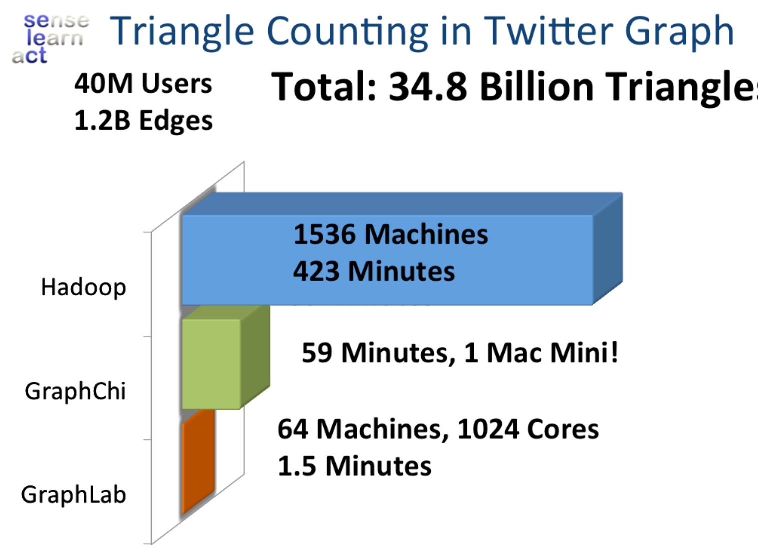
Discussion on MapReduce

- MapReduce is not designed for iterative processing
  - Mappers read the same data again and again
- MapReduce looks too low-level to some people
  - Data analysts are traditionally SQL folks ☺
- MapReduce looks too high-level to others
  - A lot of MapReduce logic is hard to adapt
    - Example: grouping documents by words

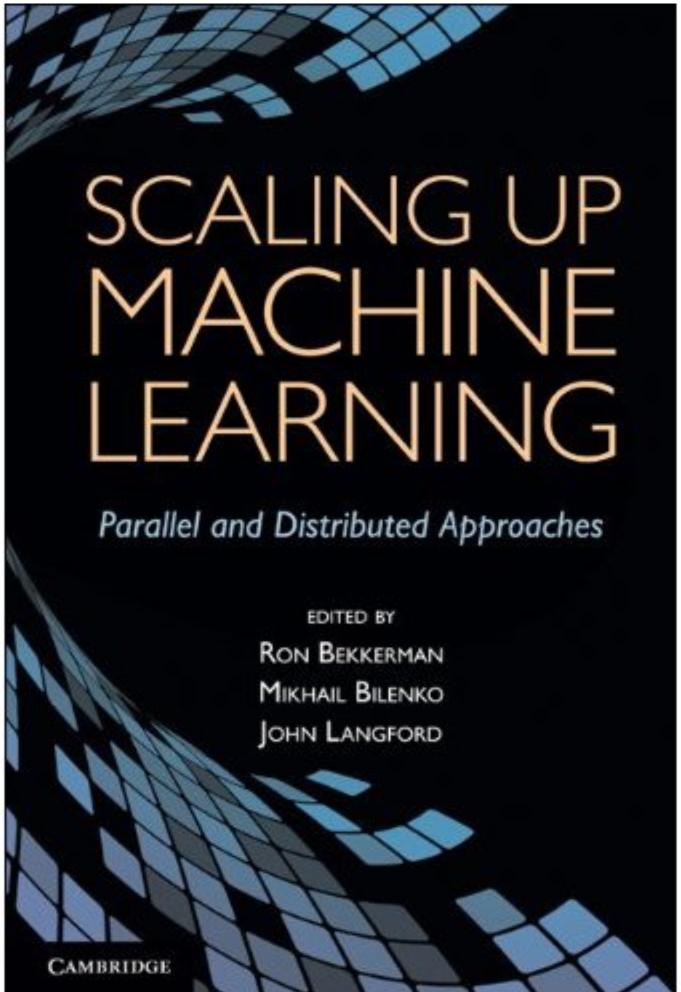
GraphLab

- Open-source parallel machine learning
- Developed at Carnegie Mellon Univ.
- Available at www.graphlab.org





For more information...

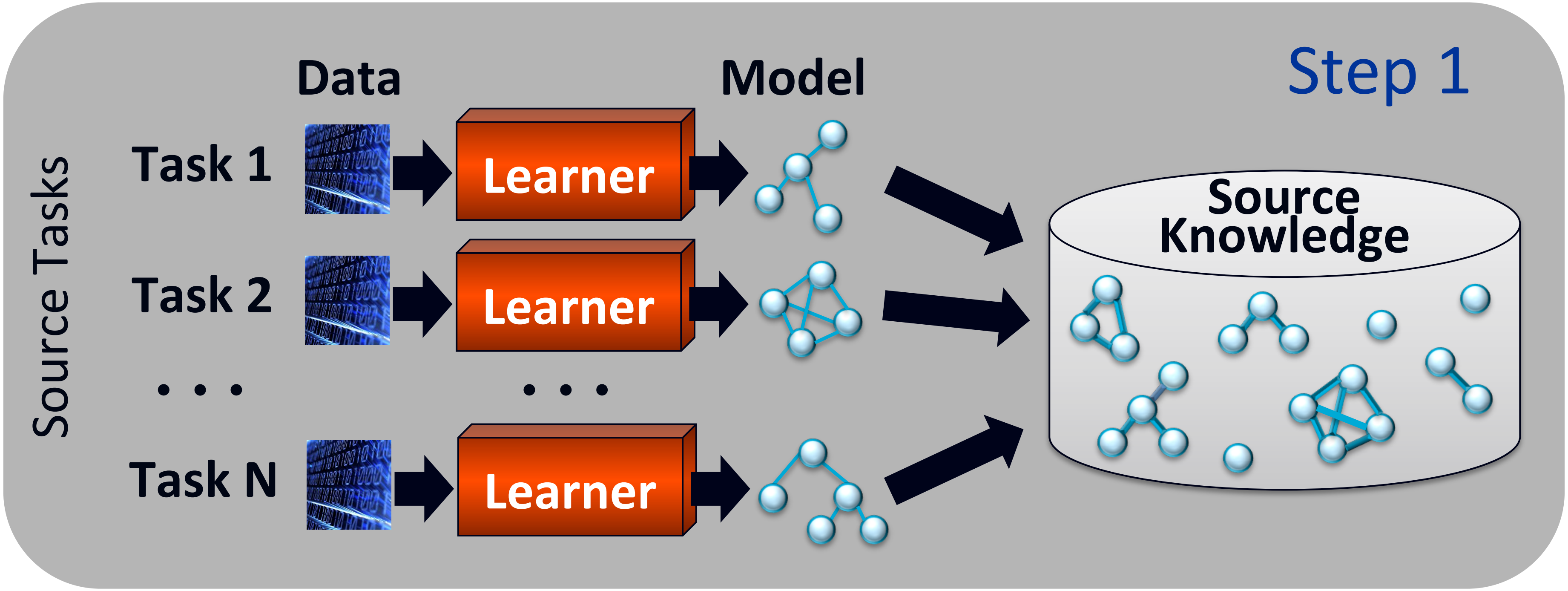


- Cambridge Univ. Press
- Released in 2011
- 21 chapters
- Covering
  - Platforms
  - Algorithms
  - Learning setups
  - Applications

Learning Multiple Tasks via Knowledge Transfer

Transfer Learning

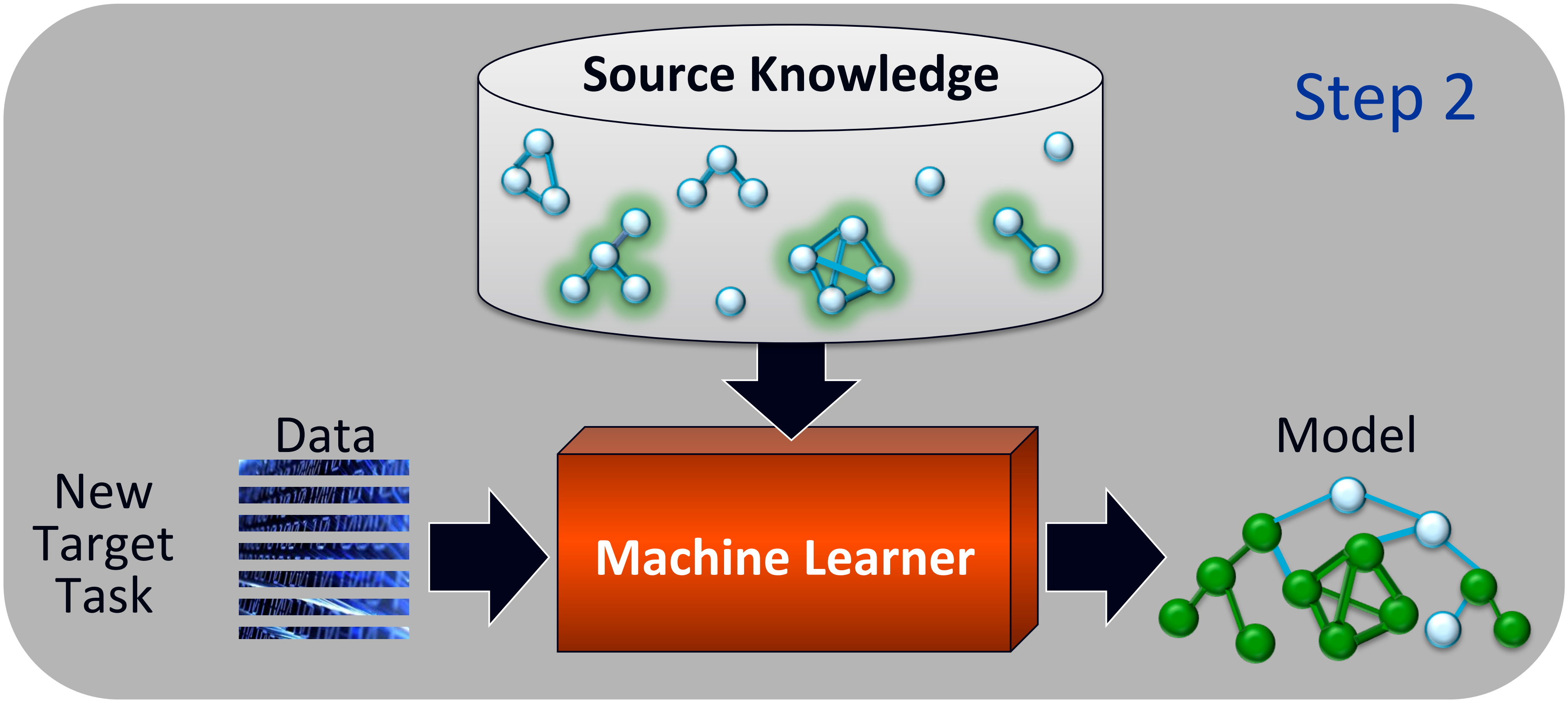
Idea: Transfer information from one or more *source tasks* to improve learning on a *target task*



- Plenty of training data for each source task

Transfer Learning

Idea: Transfer information from one or more *source tasks* to improve learning on a *target task*

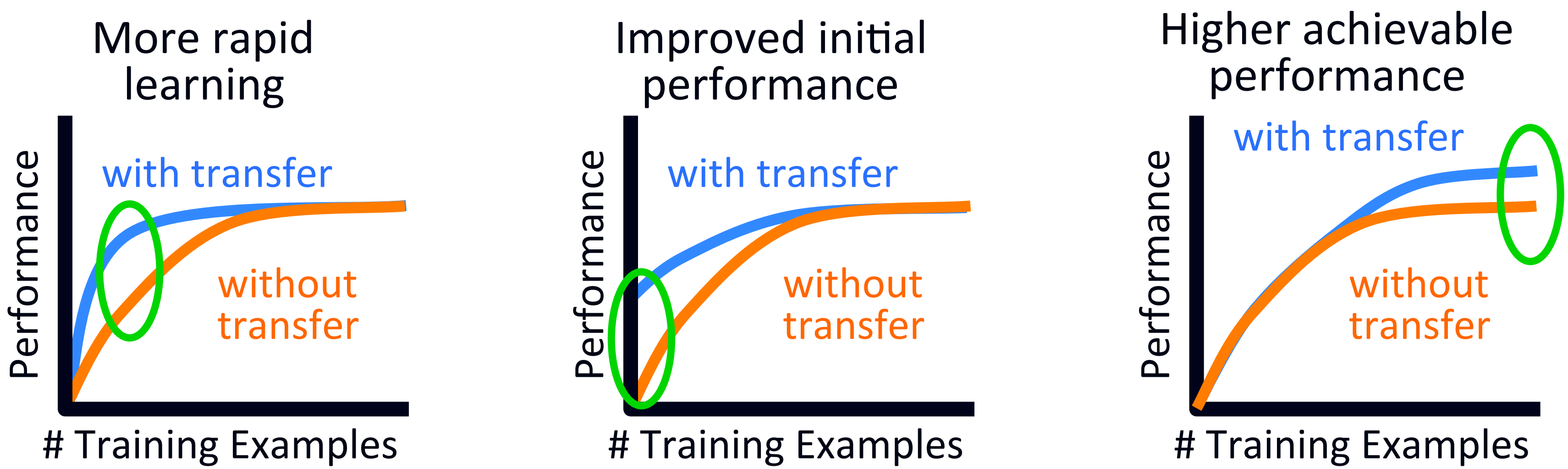


- Insufficient training data on the target task

Benefits of Transfer in Learning

- **Primary goal**: learning the target task T_{new} "better" after first learning related source tasks T_1, ..., T_N

"Better" means some combination of:



Figures adapted from (DARPA/IPTO, 2005)

Secondary goal: creating chunks of reusable knowledge

Multi-Task Learning

- **Idea**: Learn all task models simultaneously, sharing knowledge (Caruana 1997; Zhang et al. 2008; Kumar & Daumé 2012)