



Large Scale Learning

These slides were assembled by Eric Eaton, with grateful acknowledgement of the many others who made their course materials freely available online. Feel free to reuse or adapt these slides for your own academic purposes, provided that you include proper attribution. Please send comments and corrections to Eric.

Data hypergrowth: an example

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

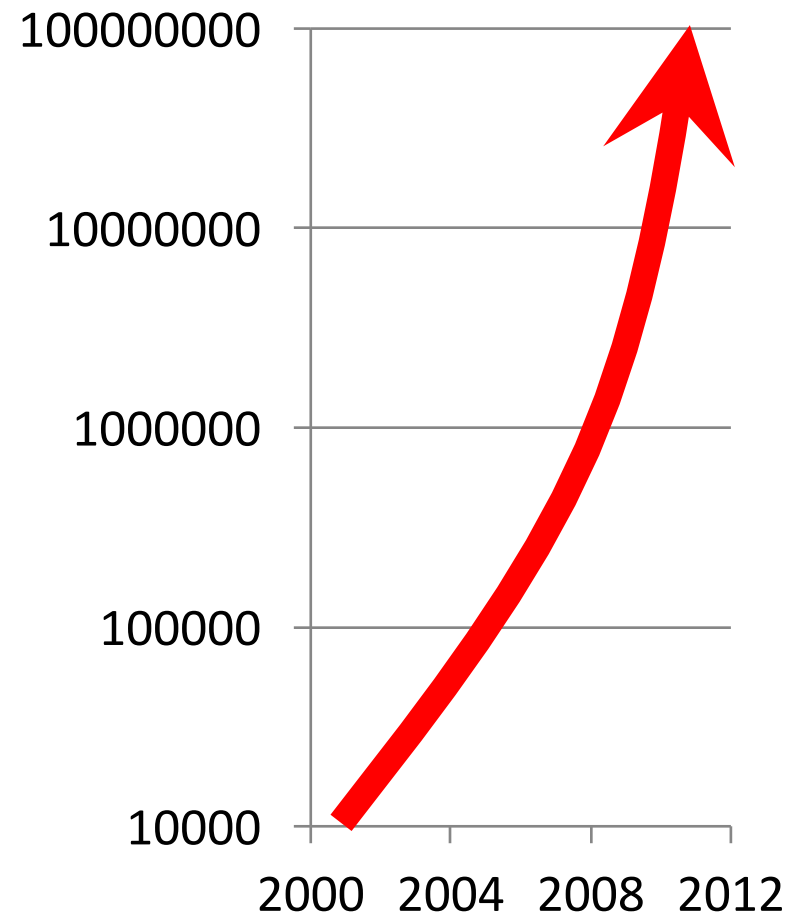
Bekkerman et al, SIGIR 2001

- 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 thousand data instances

- Will process it manually within a day (a week?)
 - No need in an automatic approach
- We shouldn't publish main results on datasets of such size 😊

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

Real-world example

Enron Email Dataset

This dataset was collected and prepared by the [CALO Project](#) (A Cognitive Assistant that Learns and Organizes). It contains data from about 150 users, mostly senior management of Enron, organized into folders. The corpus contains a total of about 0.5M messages. This data was originally [made public, and posted to the web](#), by the [Federal Energy Regulatory Commission](#) during its investigation.

The email dataset was later purchased by [Leslie Kaelbling](#) at MIT, and turned out to have a number of integrity problems. A number of folks at SRI, notably [Melinda Gervasio](#), worked hard to correct these problems, and it is thanks to them (not me) that the dataset is available. The dataset here does not include attachments, and some messages have been deleted "as part of a redaction effort due to information from affected employees". Invalid email addresses were converted to something of the form user@enron.com whenever possible (i.e., recipient is specified in some parse-able format like "Doe, John" or "Mary K. Smith") and to no_address@enron.com when no recipient was specified.

I get a number of questions about this corpus each week, which I am unable to answer, mostly because they deal with preparation issues and such that I just don't know about. If you ask me a question I don't answer, please don't feel slighted.

I am distributing this dataset as a resource for researchers who are interested in improving current email tools, or understanding how email is currently used. This data is valuable; to my knowledge it is the only substantial collection of "real" email that is public. The reason other datasets are not public is because of privacy concerns. In using this dataset, please be sensitive to the privacy of the people involved (and remember that many of these people were certainly not involved in any of the actions which precipitated the investigation.)

- ~~March 2, 2004 Version of dataset~~ and the ~~August 21, 2009 Version of dataset~~ are **no longer being distributed**. If you are using this dataset for your work, you are requested to replace it with the newer version of the dataset below, or make the [the appropriate changes](#) to your local copy. A total of four messages have been removed since the original version of the dataset.
- [August 21, 2009 Version of dataset](#) (about 423Mb, tarred and zipped).

There are also at least two on-line databases that allow you to search the data, at [Enronemail.com](#) and [UCB](#)

Research uses of the dataset

This is a partial and poorly maintained list. If I've left your work out, don't take it personally, and feel free to send me a pointer and/or description.

- [A paper describing the Enron data](#) was presented at the 2004 [CEAS conference](#).
- Some experiments associated with this data are described on [Ron Bekkerman's](#) home page.
- A social-network analysis of the data, including ["useful mappings between the MD5 digest of the email bodies and such things as authors, recipients, etc"](#), is available from [Andres Corrada-Garcia](#).

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

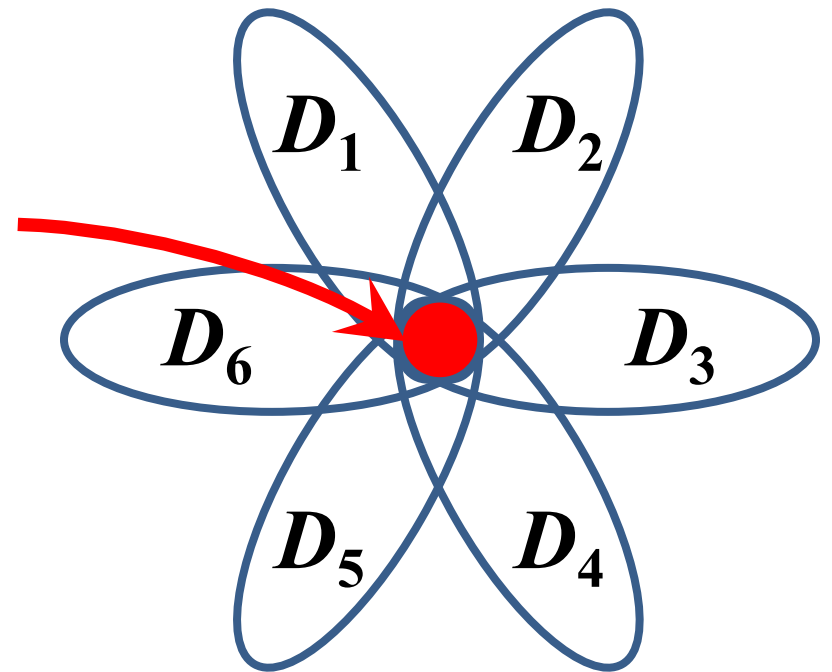
One trillion data instances

- 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

A solution to data privacy problem

Xiang et al, Chapter 16

- n machines with n private datasets
 - All datasets intersect
 - The intersection is shared
- Each machine learns a separate model
- Models get consistent over the data intersection



- **Check out Chapter 16 to see this approach applied in a recommender system!**

Skewness of training data

- Usually, training data comes from users
- **Explicit user feedback** might be misleading
 - Feedback providers may have various incentives
- Learning from **implicit feedback** is a better idea
 - E.g. clicking on Web search results
- In large-scale setups, skewness of training data is hard to detect

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:

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

- Fit via gradient descent:

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

- What is the computational complexity in terms of n ?

Gradient Descent

Batch Gradient Descent

Initialize θ

Repeat {

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

}

Stochastic Gradient Descent

Initialize θ

Randomly shuffle dataset

Repeat { (Typically 1 – 10x)

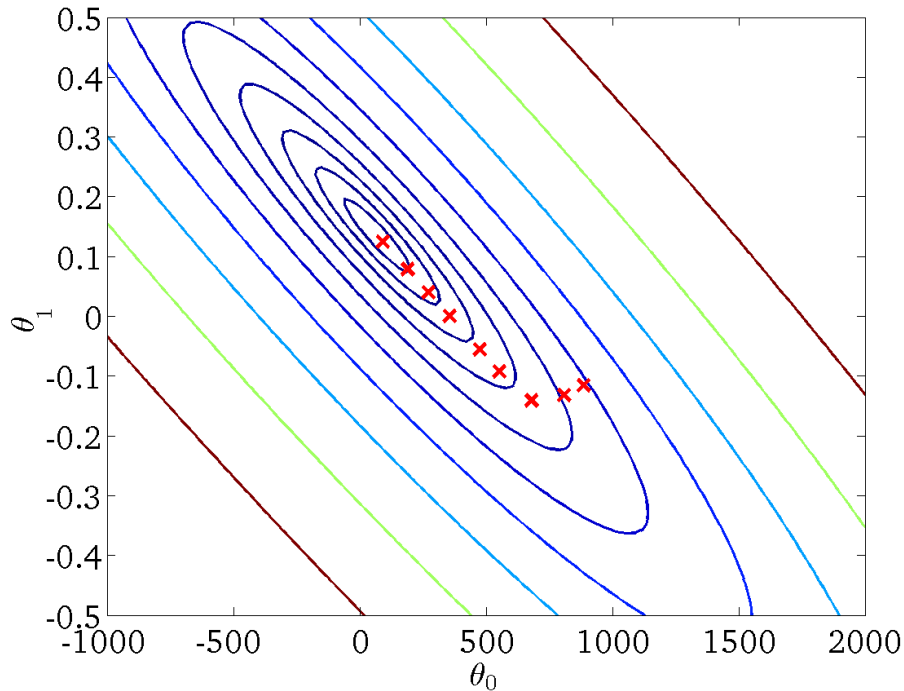
For $i = 1 \dots n$, do

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

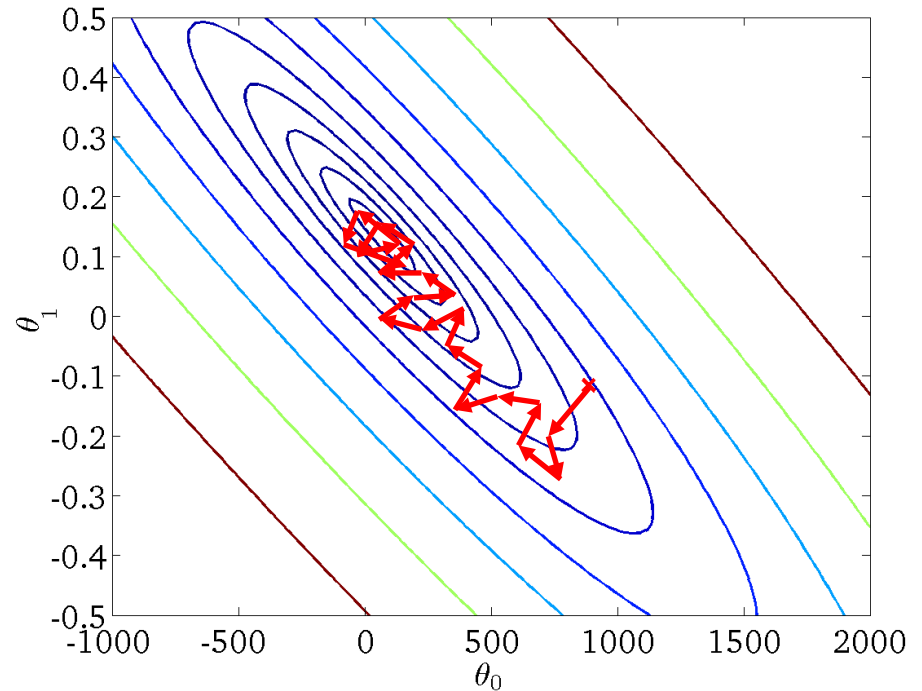
}

Batch vs Stochastic GD

Batch GD



Stochastic GD

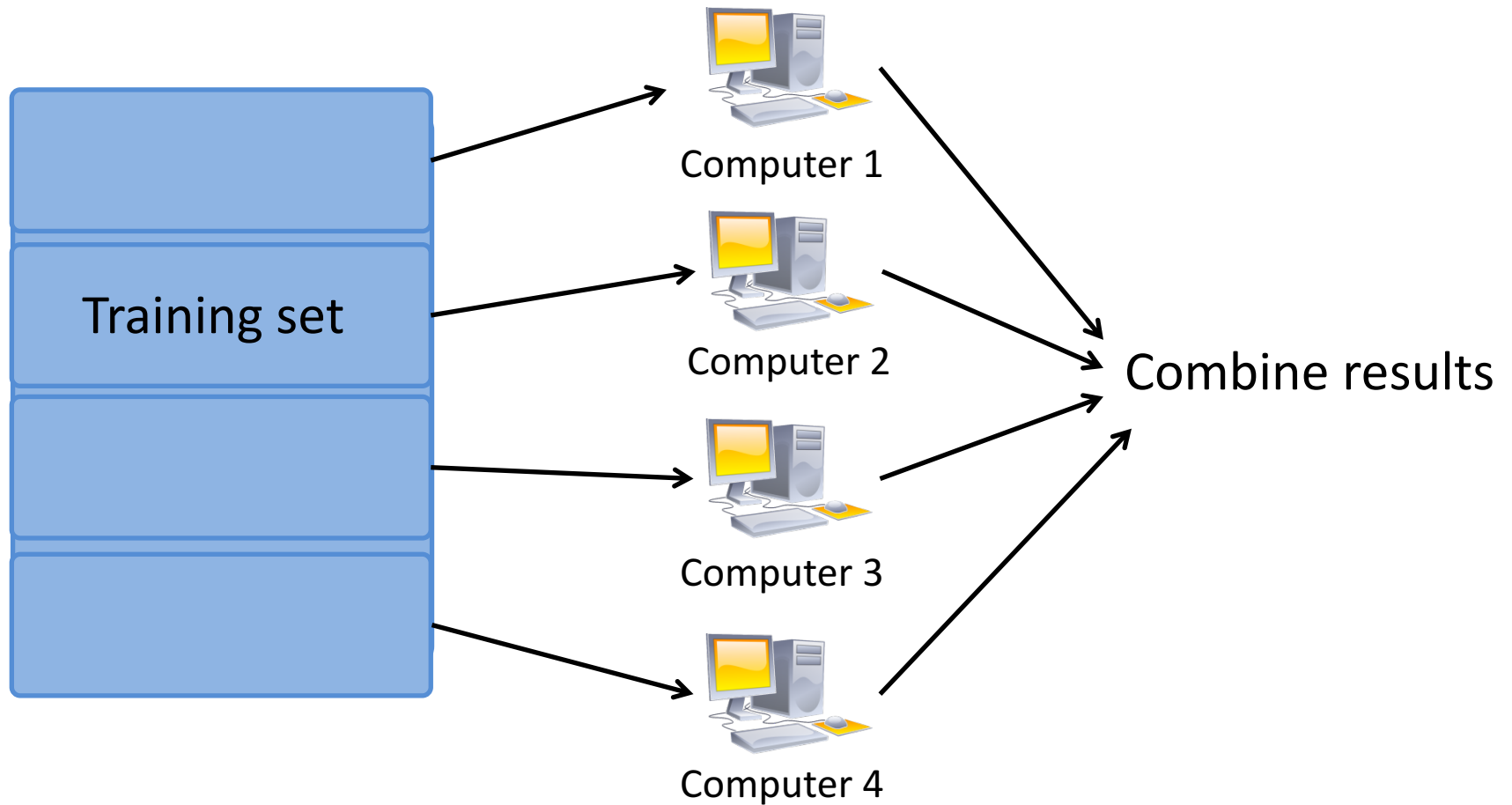


- Learning rate α is typically held constant
- Can slowly decrease α over time to force θ to converge:

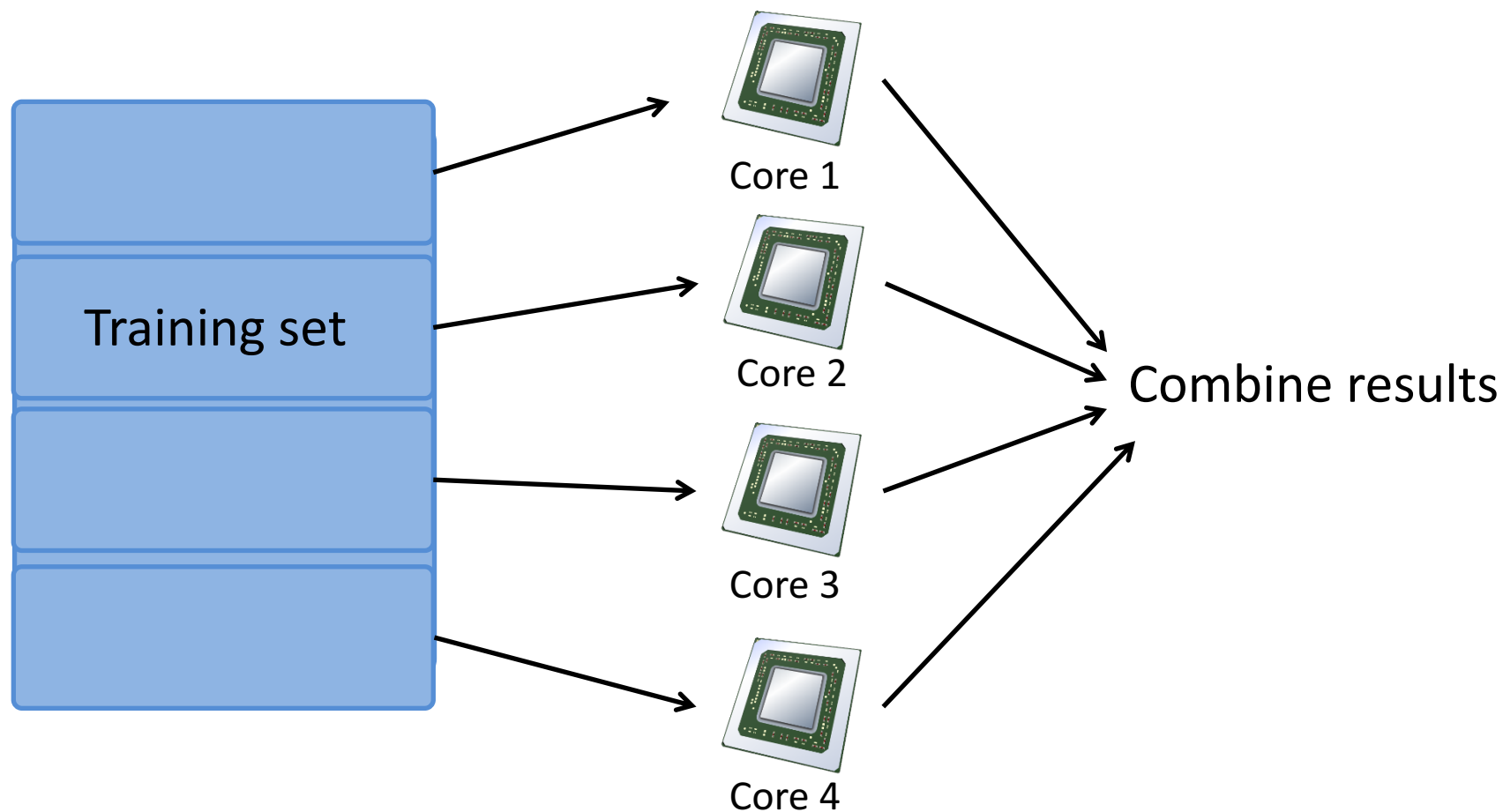
$$\text{e.g., } \alpha = \frac{\text{constant1}}{\text{iterationNumber} + \text{constant2}}$$

Graph- and Data-Parallelism

Map-Reduce



Multi-Core Machines



Map-Reduce for Batch GD

Split dataset up into chunks (e.g., with $n = 400$) to

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



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



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



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



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

$(\mathbf{x}_1, y_1) \dots (\mathbf{x}_{100}, y_{100})$

$(\mathbf{x}_{101}, y_{101}) \dots (\mathbf{x}_{200}, y_{200})$

$(\mathbf{x}_{201}, y_{201}) \dots (\mathbf{x}_{300}, y_{300})$

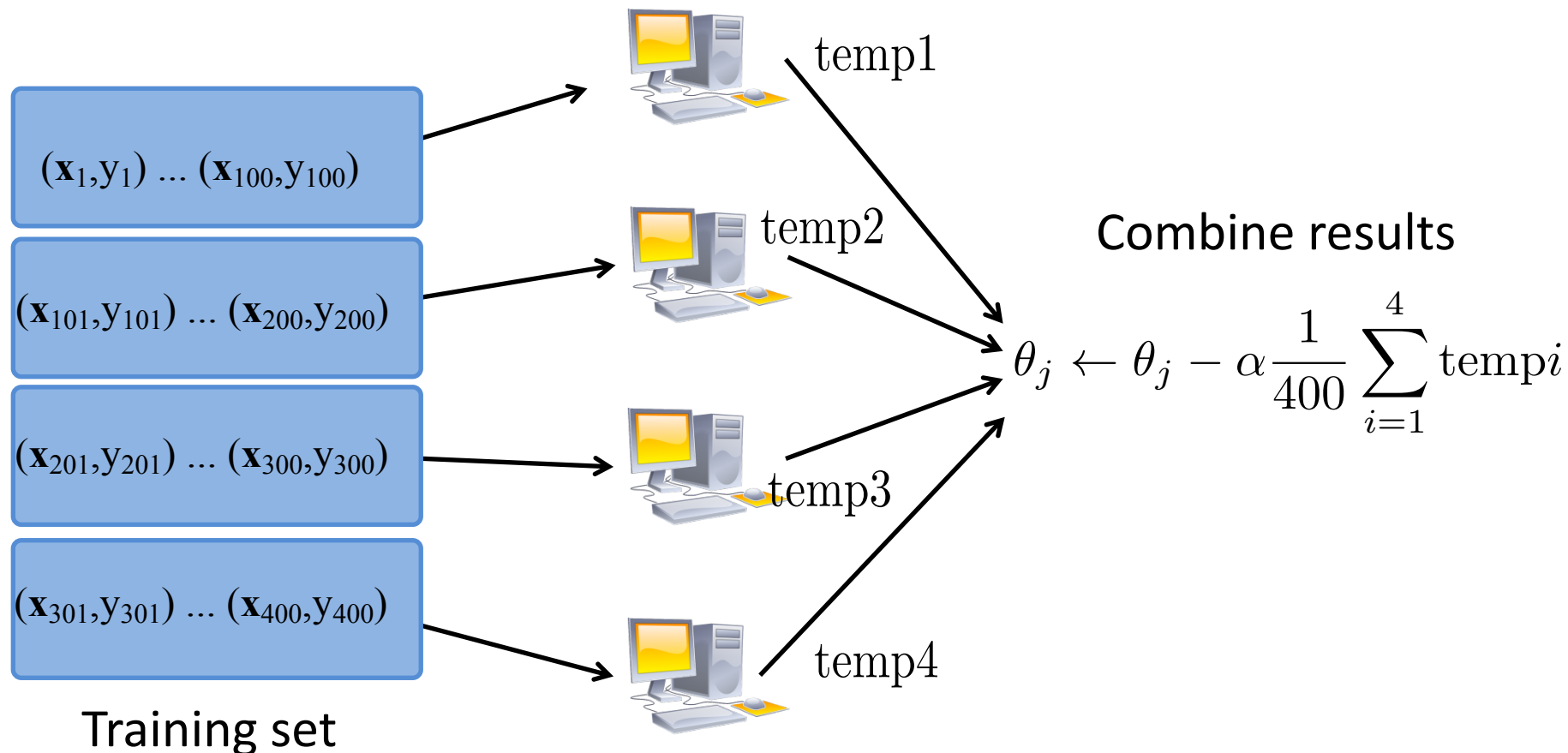
$(\mathbf{x}_{301}, y_{301}) \dots (\mathbf{x}_{400}, y_{400})$

Training set

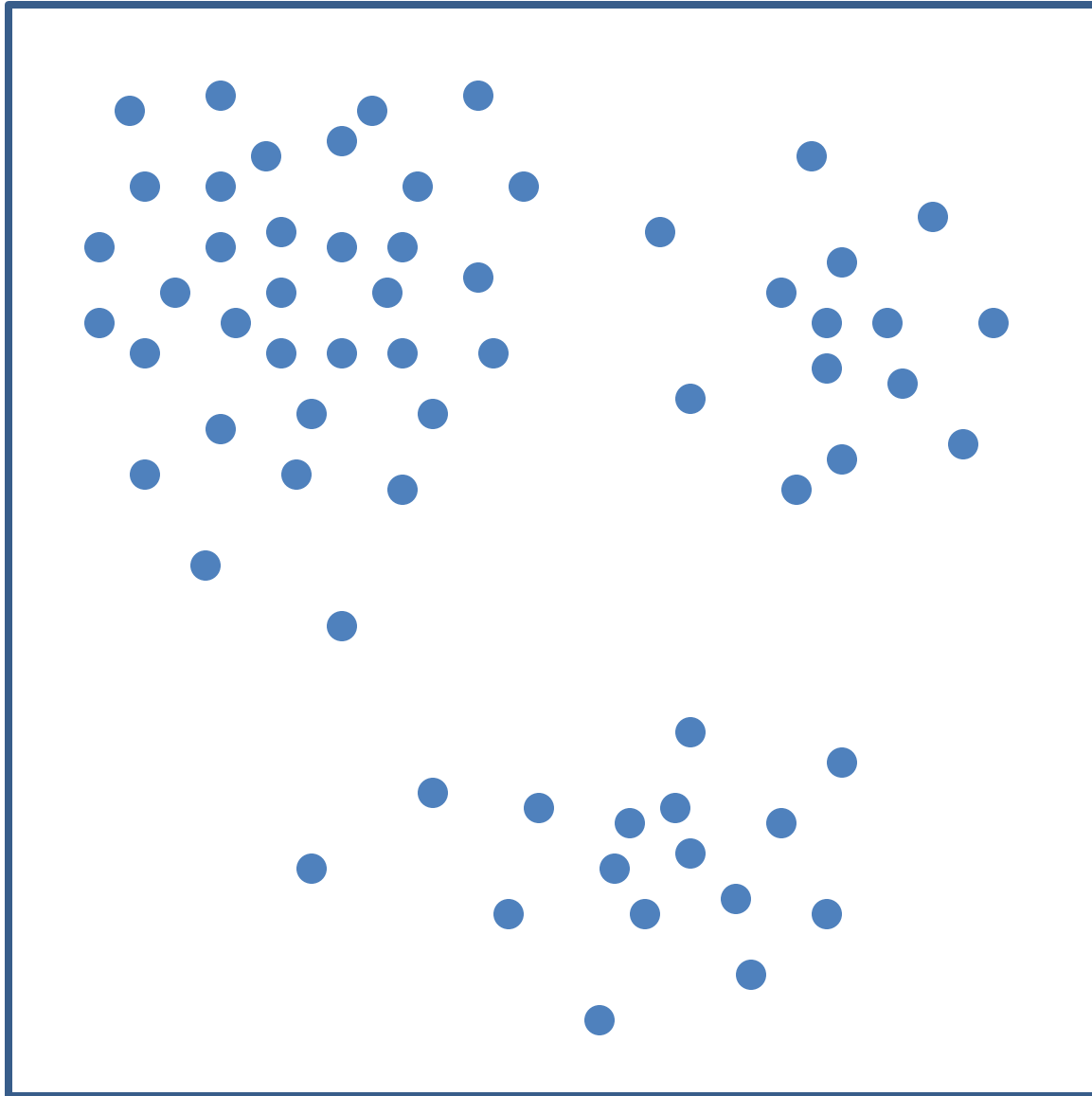
Map-Reduce for Batch GD

Split dataset up into chunks (e.g., with $n = 400$) to

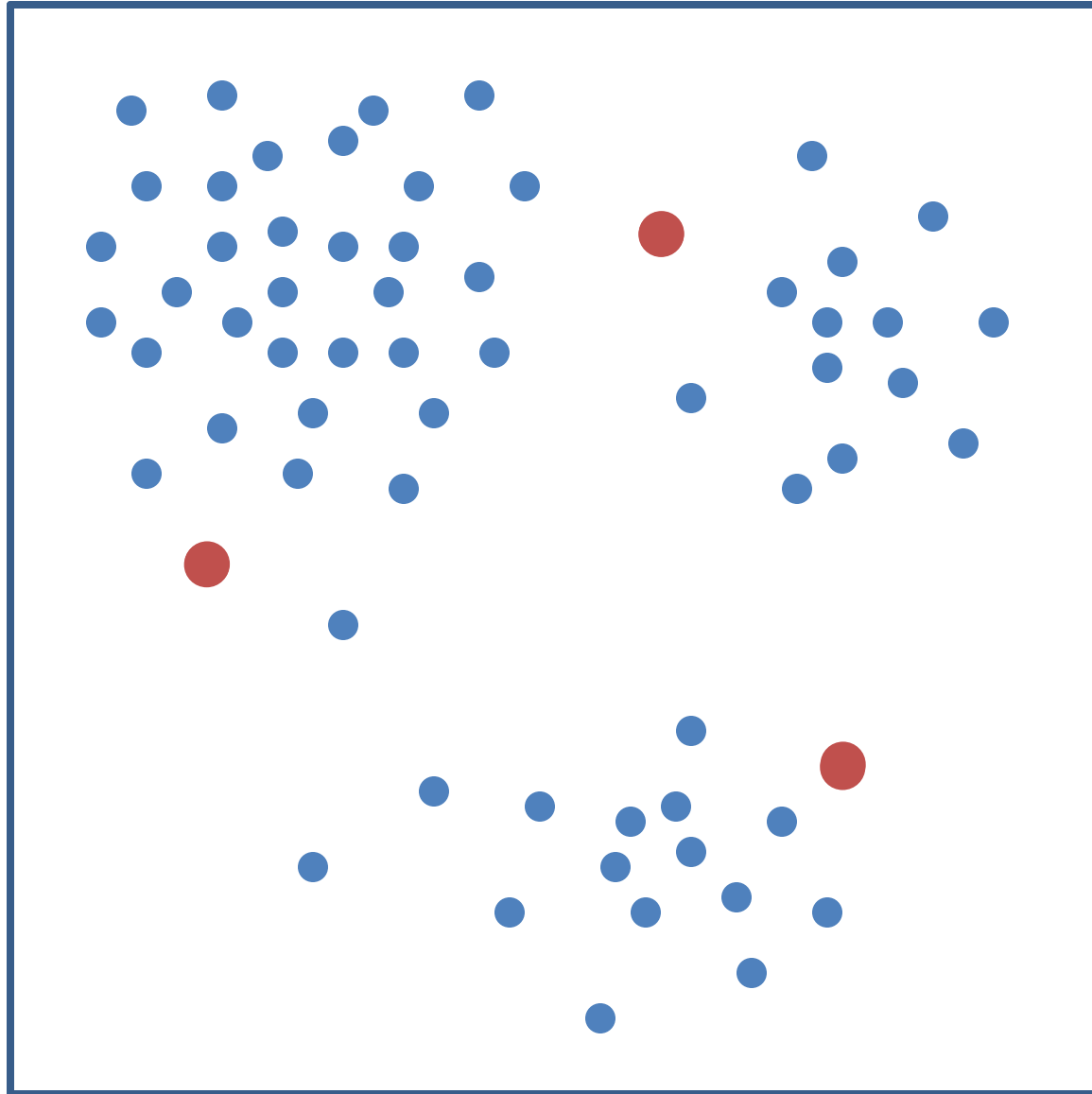
compute
$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n (h_{\theta}(\mathbf{x}_i) - y_i) x_{ij}$$



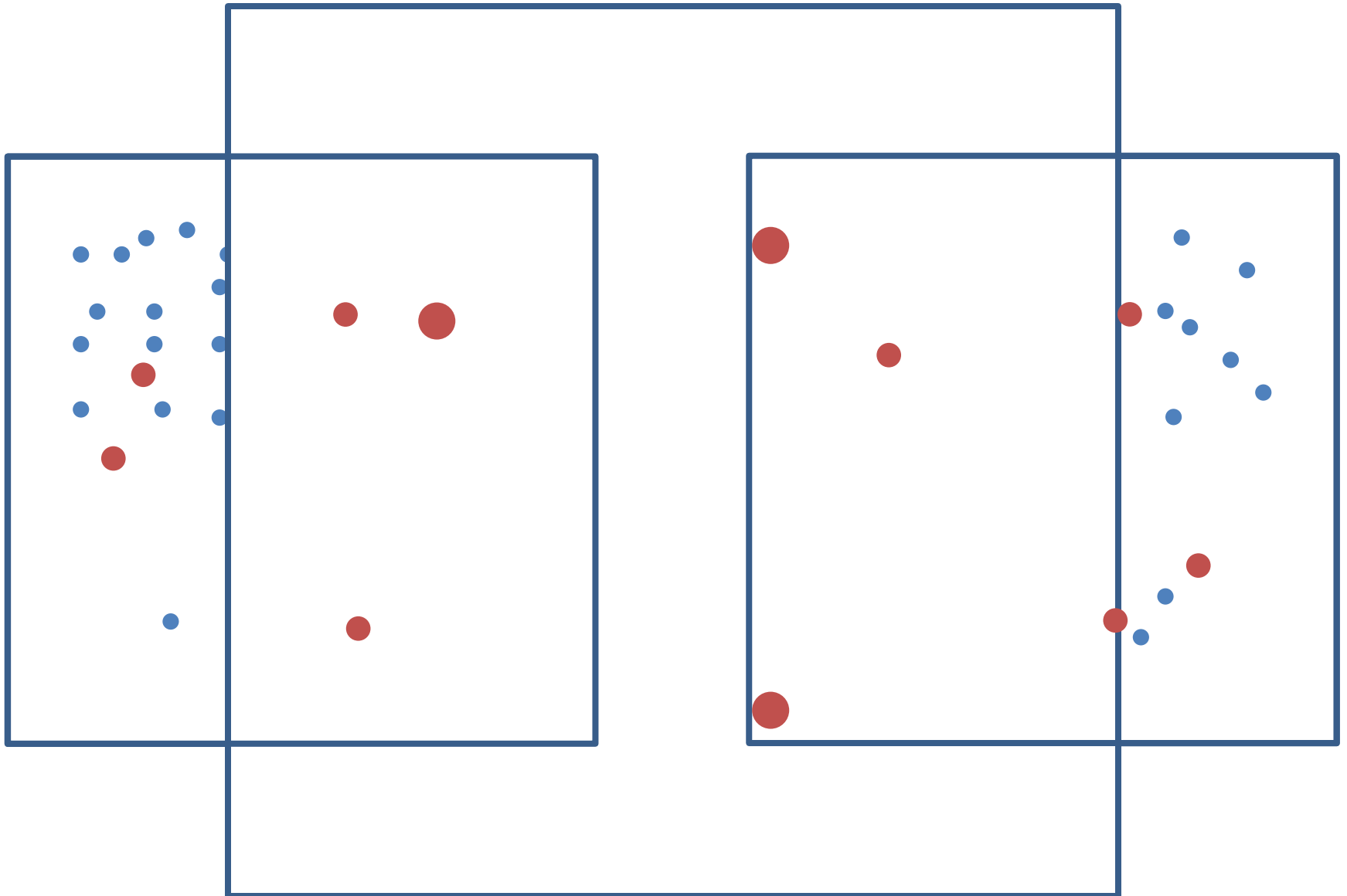
Parallelizing k -means



Parallelizing k -means

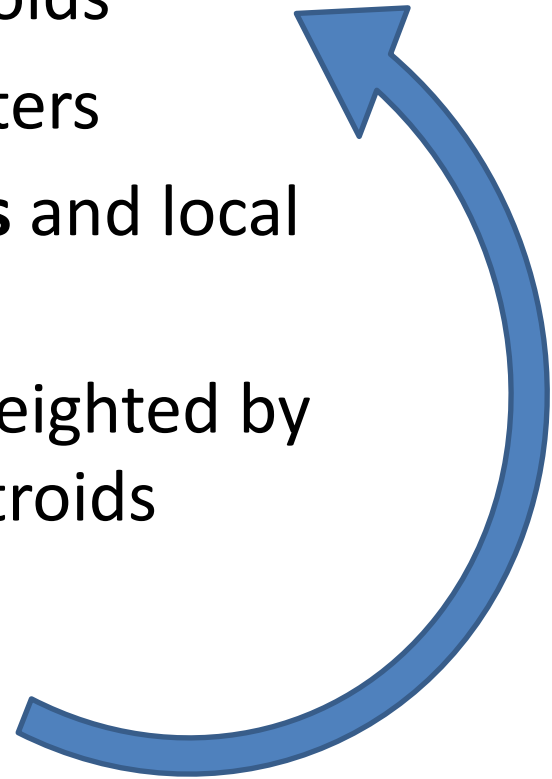


Parallelizing k -means



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**



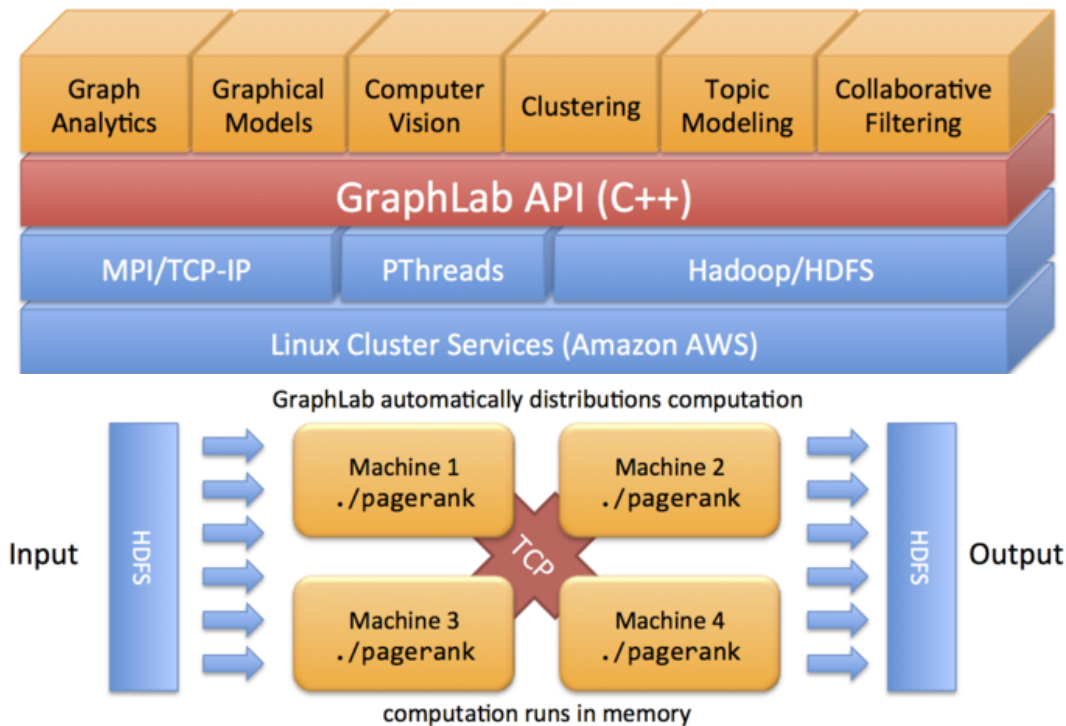
Discussion on MapReduce

- 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

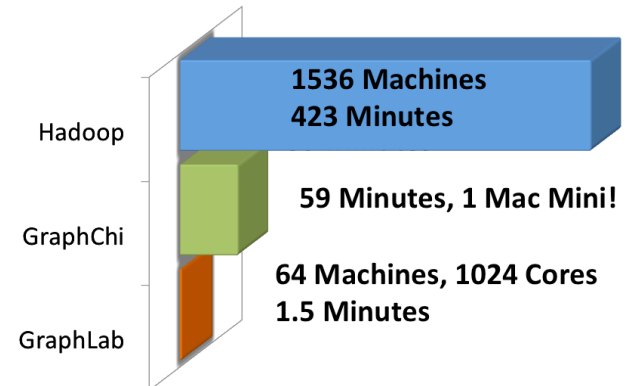
GraphLab



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



sense
learn
act **Triangle Counting in Twitter Graph**
40M Users
1.2B Edges
Total: 34.8 Billion Triangles

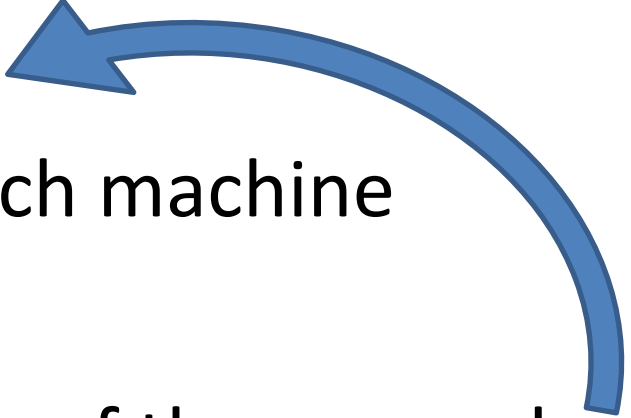


Peer-to-peer (P2P) systems

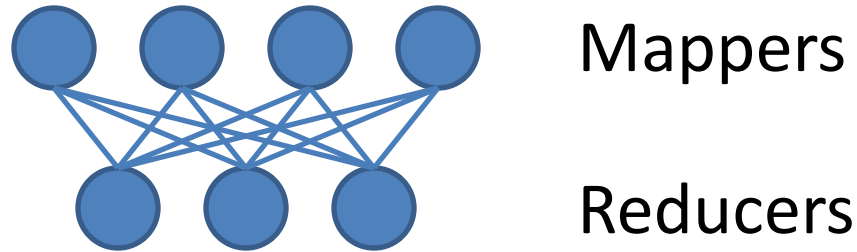
- Millions of machines connected in a network
 - Each machine can only contact its neighbors
- Each machine storing millions of data instances
 - Practically unlimited scale 😊
- Communication is the bottleneck
 - Aggregation is costly, broadcast is cheaper
- Messages are sent over a spanning tree
 - With an arbitrary node being the root

k -means in P2P

Datta et al, TKDE 2009

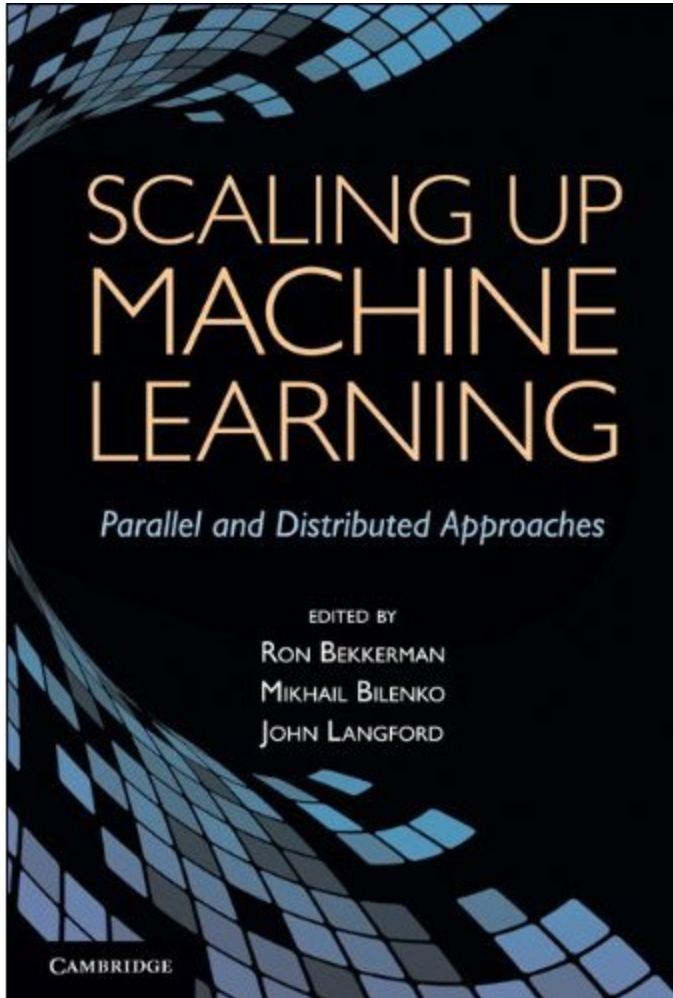
- Uniformly sample k centroids over P2P
 - Using a random walk method
 - Broadcast the centroids
 - Run local k -means on each machine
 - Sample n nodes
 - Aggregate local centroids of those n nodes
- 

MapReduce



- Process in parallel → shuffle → process in parallel
- Mappers output (key, value) records
 - Records with the same key are sent to the same reducer

For more information...

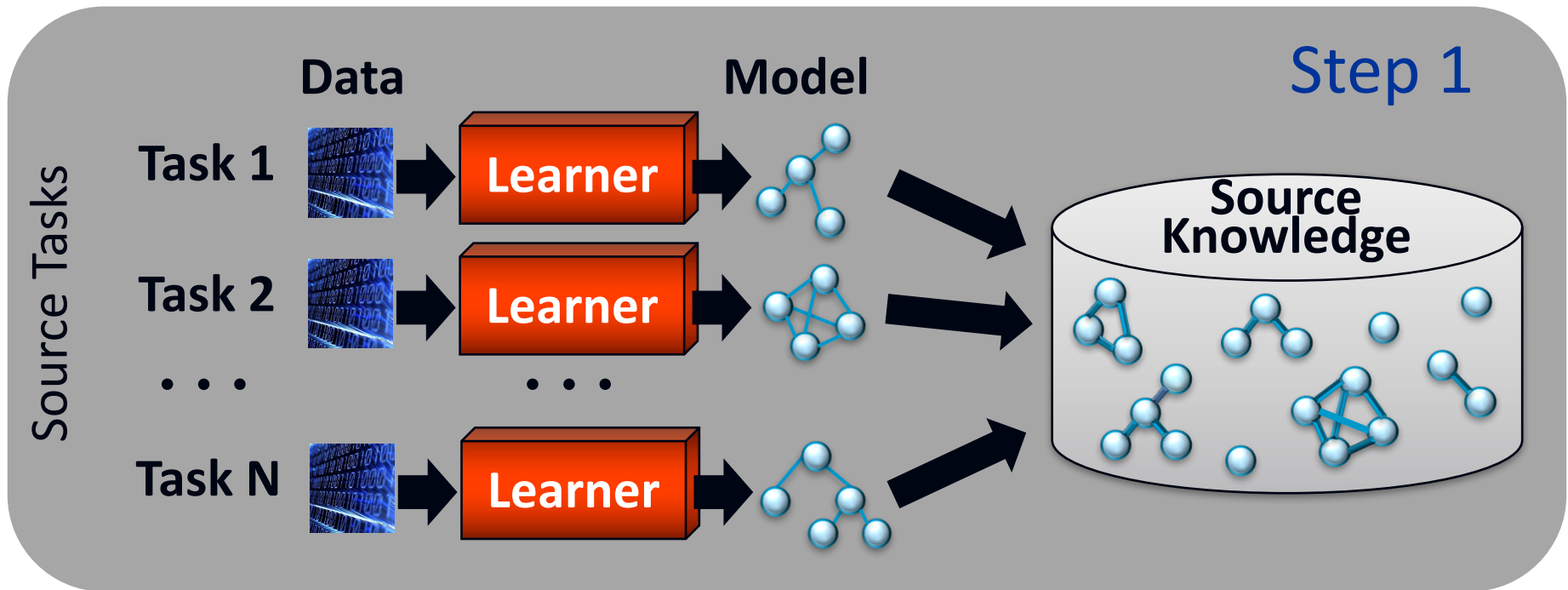


- Cambridge Univ. Press
- Released in 2011
- 21 chapters
- Covering
 - Platforms
 - Algorithms
 - Learning setups
 - Applications

Learning Multiple Tasks via Knowledge Transfer

Transfer Learning

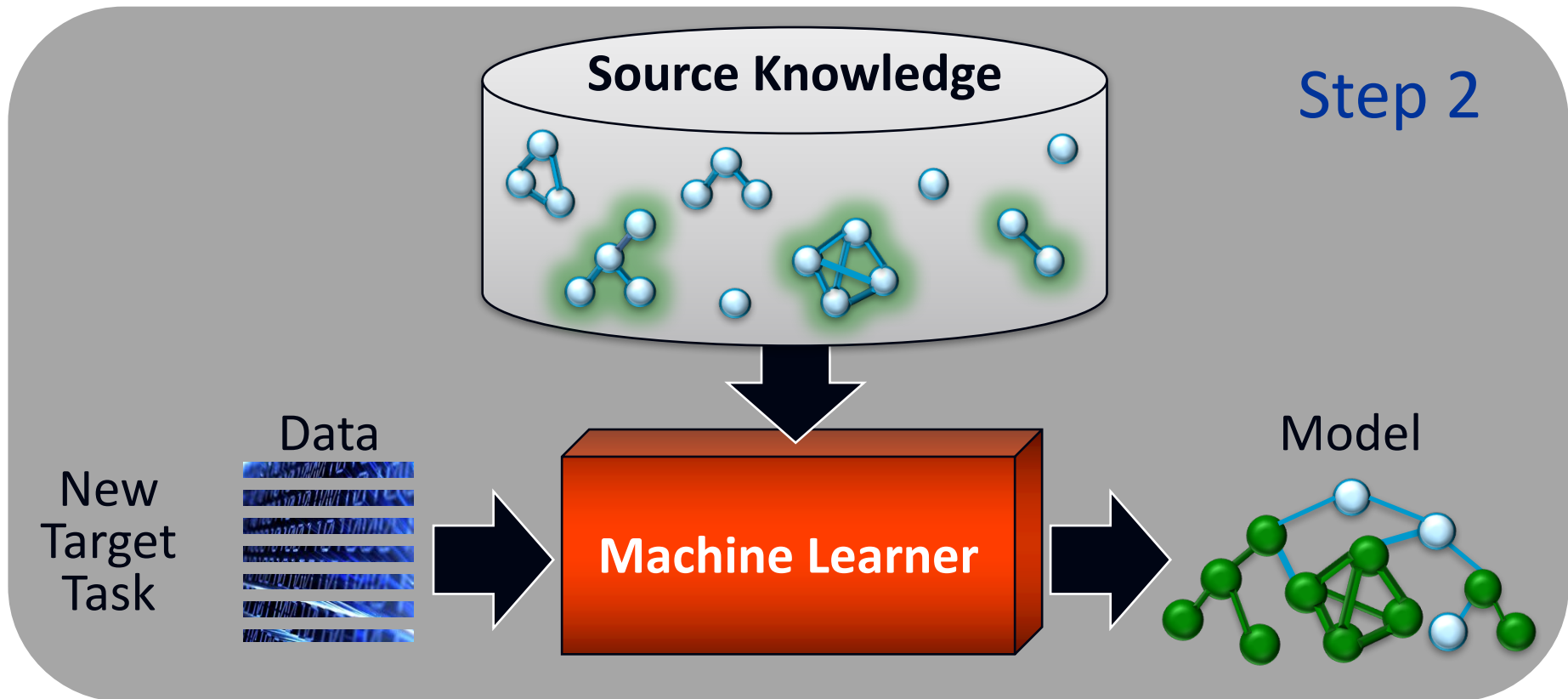
Idea: Transfer information from one or more *source tasks* to improve learning on a *target task*



- 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*

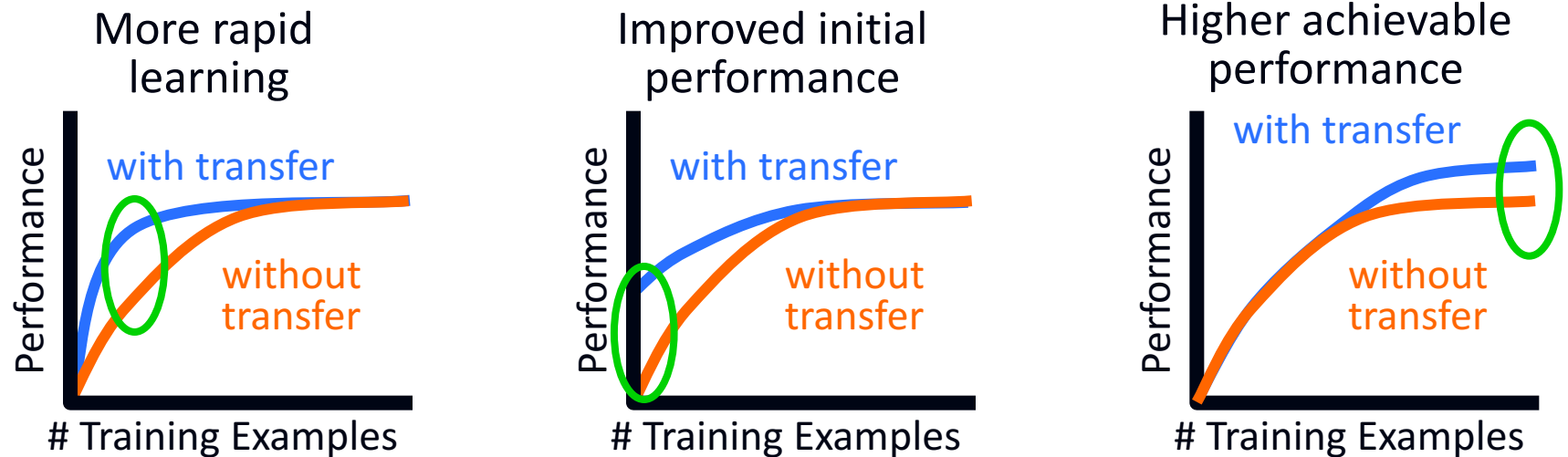


- Insufficient training data on the target task

Benefits of Transfer in Learning

- **Primary goal:** learning the target task T_{new} “better” after first learning related source tasks T_1, \dots, 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)

