

# Data-Intensive Computing with MapReduce

## Session 7: Clustering and Classification

Jimmy Lin  
University of Maryland  
Thursday, March 7, 2013



This work is licensed under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States  
See <http://creativecommons.org/licenses/by-nc-sa/3.0/us/> for details

# Today's Agenda

- Personalized PageRank
- Clustering
- Classification

# Clustering



# Problem Setup

- Arrange items into clusters
  - High similarity between objects in the same cluster
  - Low similarity between objects in different clusters
- Cluster labeling is a separate problem

# Applications

- Exploratory analysis of large collections of objects
- Image segmentation
- Recommender systems
- Cluster hypothesis in information retrieval
- Computational biology and bioinformatics
- Pre-processing for many other algorithms

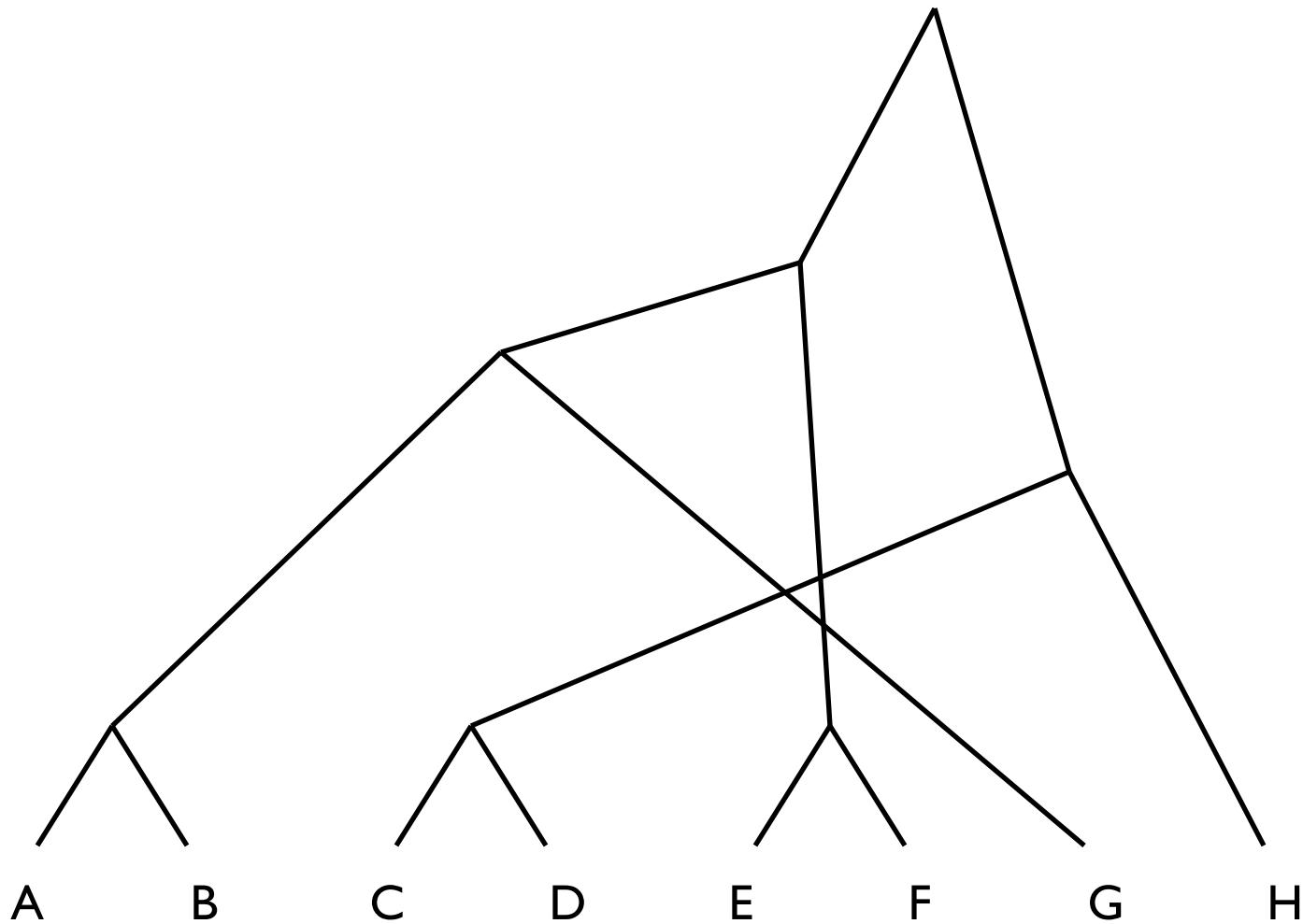
# Three Approaches

- Hierarchical
- K-Means
- Gaussian Mixture Models

# Hierarchical Agglomerative Clustering

- Start with each document in its own cluster
- Until there is only one cluster:
  - Find the two clusters  $c_i$  and  $c_j$ , that are most similar
  - Replace  $c_i$  and  $c_j$  with a single cluster  $c_i \cup c_j$
- The history of merges forms the hierarchy

# HAC in Action



# Cluster Merging

- Which two clusters do we merge?
- What's the similarity between two clusters?
  - Single Link: similarity of two most similar members
  - Complete Link: similarity of two least similar members
  - Group Average: average similarity between members

# Link Functions

- Single link:

- Uses maximum similarity of pairs:

$$\text{sim}(c_i, c_j) = \max_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

- Can result in “straggly” (long and thin) clusters due to *chaining effect*

- Complete link:

- Use minimum similarity of pairs:

$$\text{sim}(c_i, c_j) = \min_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

- Makes more “tight” spherical clusters

# MapReduce Implementation

- What's the inherent challenge?
- One possible approach:
  - Iteratively use LSH to group together similar items
  - When dataset is small enough, run HAC in memory on a single machine
  - Observation: structure at the leaves is not very important

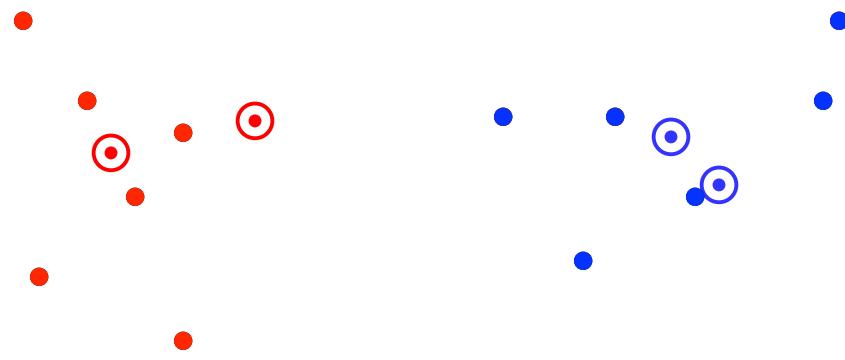
# K-Means Algorithm

- Let  $d$  be the distance between documents
- Define the centroid of a cluster to be:

$$\mu(c) = \frac{1}{|c|} \sum_{x \in c} x$$

- Select  $k$  random instances  $\{s_1, s_2, \dots, s_k\}$  as seeds.
- Until clusters converge:
  - Assign each instance  $x_i$  to the cluster  $c_j$  such that  $d(x_i, s_j)$  is minimal
  - Update the seeds to the centroid of each cluster
  - For each cluster  $c_j$ ,  $s_j = \mu(c_j)$

# K-Means Clustering Example



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

# Basic MapReduce Implementation

```
1: class MAPPER
2:   method CONFIGURE()
3:     c ← LOADCLUSTERS()
4:   method MAP(id i, point p)
5:     n ← NEARESTCLUSTERID(clusters c, point p)
6:     p ← EXTENDPOINT(point p) ← (Just a clever way to keep
7:     EMIT(clusterid n, point p) track of denominator)
1: class REDUCER
2:   method REDUCE(clusterid n, points [p1, p2, ...])
3:     s ← INITPOINTSUM()
4:     for all point p ∈ points do
5:       s ← s + p
6:     m ← COMPUTECENTROID(point s)
7:     EMIT(clusterid n, centroid m)
```

# MapReduce Implementation w/ IMC

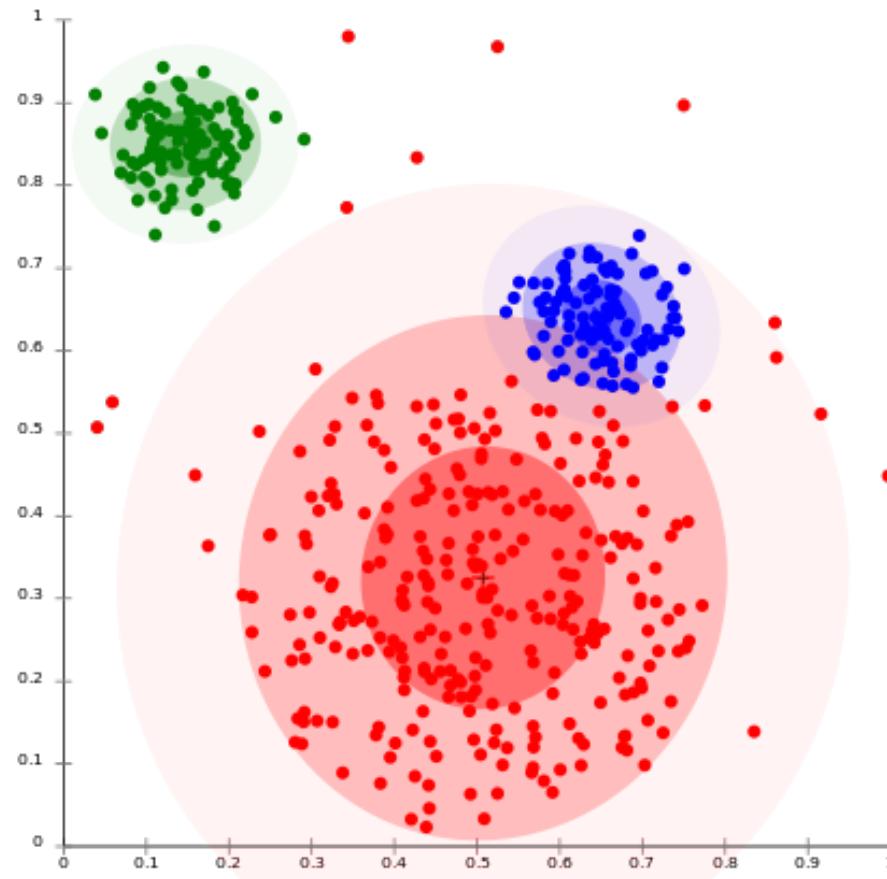
```
1: class MAPPER
2:   method CONFIGURE()
3:      $c \leftarrow \text{LOADCLUSTERS}()$ 
4:      $H \leftarrow \text{INITASSOCIATIVEARRAY}()$ 
5:   method MAP(id  $i$ , point  $p$ )
6:      $n \leftarrow \text{NEARESTCLUSTERID}(\text{clusters } c, \text{ point } p)$ 
7:      $p \leftarrow \text{EXTENDPOINT}(\text{point } p)$ 
8:      $H\{n\} \leftarrow H\{n\} + p$ 
9:   method CLOSE()
10:    for all clusterid  $n \in H$  do
11:      EMIT(clusterid  $n$ , point  $H\{n\}$ )
1: class REDUCER
2:   method REDUCE(clusterid  $n$ , points  $[p_1, p_2, \dots]$ )
3:      $s \leftarrow \text{INITPOINTSUM}()$ 
4:     for all point  $p \in \text{points}$  do
5:        $s \leftarrow s + p$ 
6:      $m \leftarrow \text{COMPUTECENTROID}(\text{point } s)$ 
7:     EMIT(clusterid  $n$ , centroid  $m$ )
```

# Implementation Notes

- Standard setup of iterative MapReduce algorithms
  - Driver program sets up MapReduce job
  - Waits for completion
  - Checks for convergence
  - Repeats if necessary
- Must be able keep cluster centroids in memory
  - With large  $k$ , large feature spaces, potentially an issue
  - Memory requirements of centroids grow over time!
- Variant:  $k$ -medoids

# Clustering w/ Gaussian Mixture Models

- Model data as a mixture of Gaussians
- Given data, recover model parameters



# Gaussian Distributions

- Univariate Gaussian (i.e., Normal):

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x - \mu)^2\right)$$

- A random variable with such a distribution we write as:

$$x \sim \mathcal{N}(\mu, \sigma^2)$$

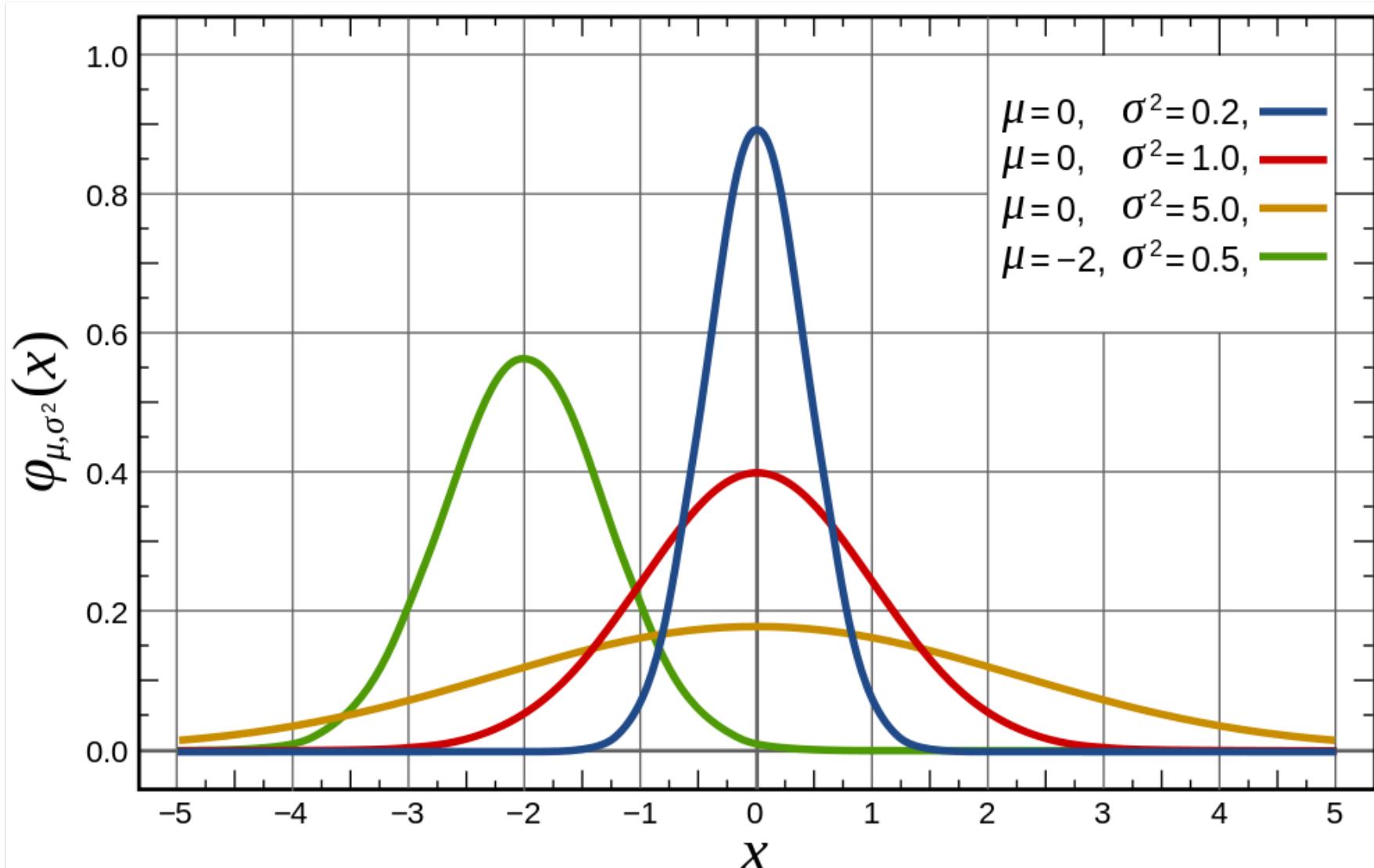
- Multivariate Gaussian:

$$p(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right)$$

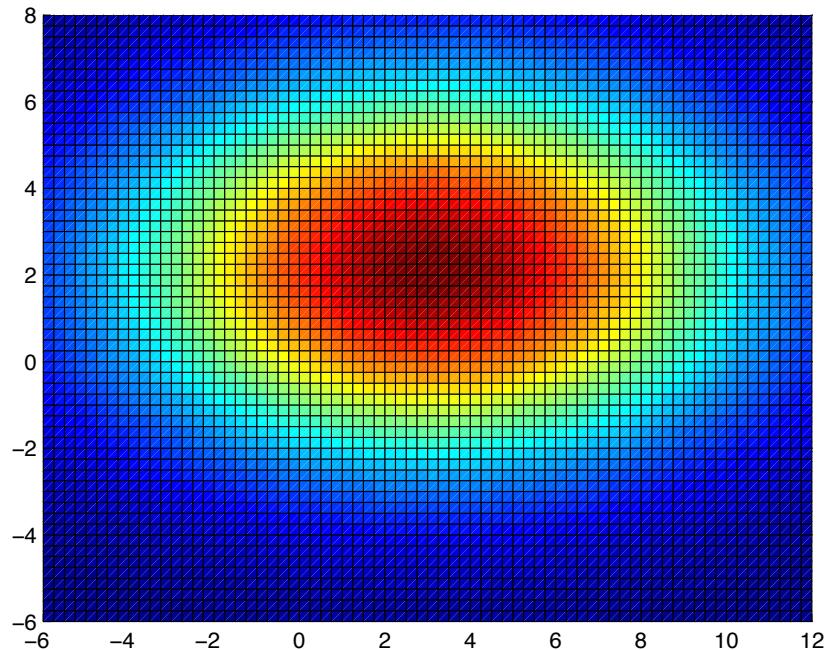
- A vector-value random variable with such a distribution we write as:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

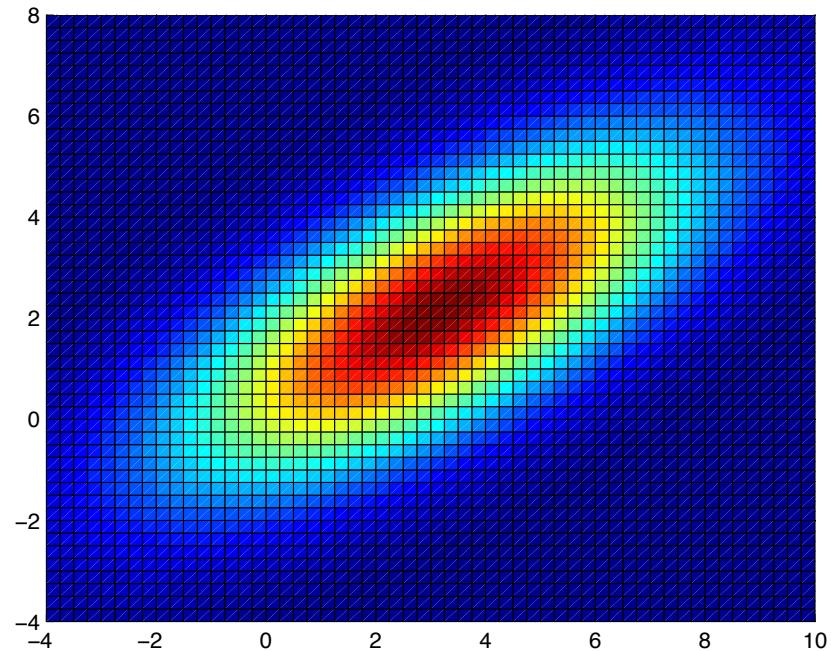
# Univariate Gaussian



# Multivariate Gaussians



$$\mu = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 25 & 0 \\ 0 & 9 \end{bmatrix}$$



$$\mu = \begin{bmatrix} 3 \\ 2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}$$

# Gaussian Mixture Models

- Model parameters
  - Number of components:  $K$
  - “Mixing” weight vector:  $\pi$
  - For each Gaussian, mean and covariance matrix:  $\mu_{1:K}$   $\Sigma_{1:K}$
- Varying constraints on co-variance matrices
  - Spherical vs. diagonal vs. full
  - Tied vs. untied

# Learning for Simple Univariate Case

- Problem setup:
  - Given number of components:  $K$
  - Given points:  $x_{1:N}$
  - Learn parameters:  $\pi, \mu_{1:K}, \sigma_{1:K}^2$
- Model selection criterion: maximize likelihood of data
  - Introduce indicator variables:

$$z_{n,k} = \begin{cases} 1 & \text{if } x_n \text{ is in cluster } k \\ 0 & \text{otherwise} \end{cases}$$

- Likelihood of the data:
$$p(x_{1:N}, z_{1:N,1:K} | \mu_{1:K}, \sigma_{1:K}^2, \pi)$$

# EM to the Rescue!

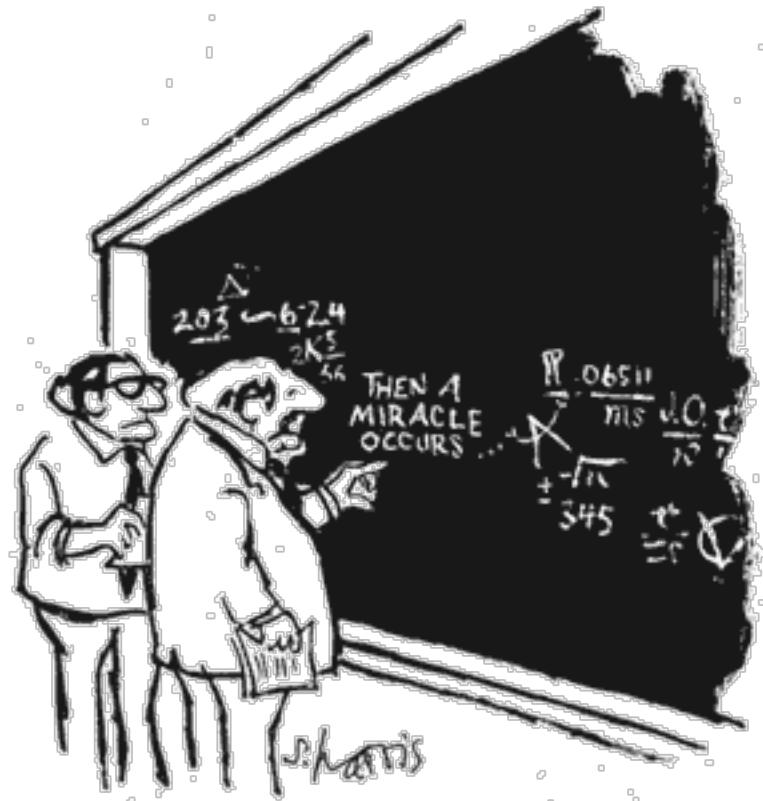
- We're faced with this:

$$p(x_{1:N}, z_{1:N,1:K} | \mu_{1:K}, \sigma_{1:K}^2, \pi)$$

- It'd be a lot easier if we knew the z's!

- Expectation Maximization

- Guess the model parameters
- E-step: Compute posterior distribution over latent (hidden) variables given the model parameters
- M-step: Update model parameters using posterior distribution computed in the E-step
- Iterate until convergence



"I THINK YOU SHOULD BE MORE  
EXPLICIT HERE IN STEP TWO."

# EM for Univariate GMMs

- Initialize:  $\pi, \mu_{1:K}, \sigma_{1:K}^2$
- Iterate:
  - E-step: compute expectation of z variables

$$\mathbb{E}[z_{n,k}] = \frac{\mathcal{N}(x_n | \mu_k, \sigma_k^2) \cdot \pi_k}{\sum_{k'} \mathcal{N}(x_n | \mu_{k'}, \sigma_{k'}^2) \cdot \pi_{k'}}$$

- M-step: compute new model parameters

$$\pi_k = \frac{1}{N} \sum_n z_{n,k}$$

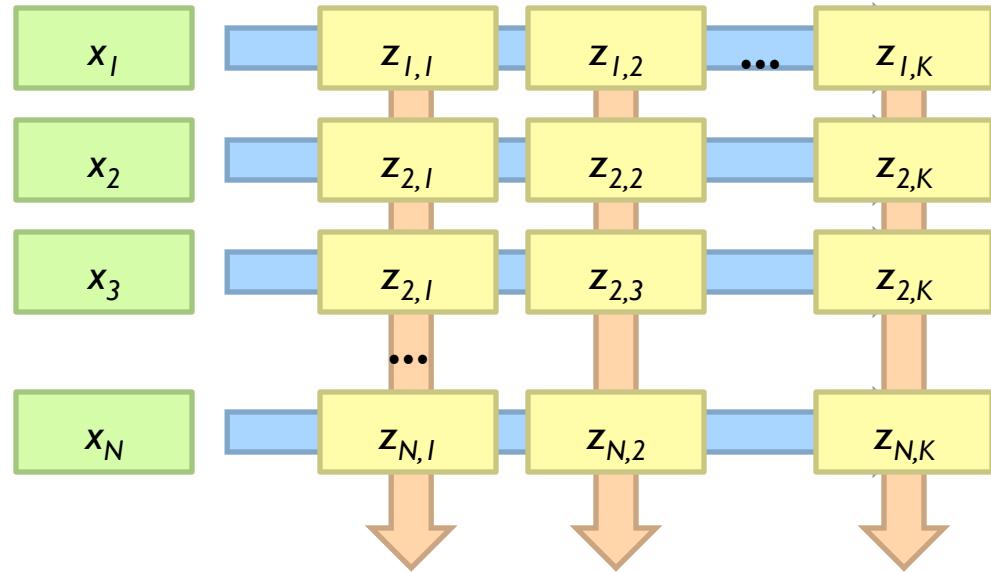
$$\mu_k = \frac{1}{\sum_n z_{n,k}} \sum_n z_{n,k} \cdot x_n$$

$$\sigma_k^2 = \frac{1}{\sum_n z_{n,k}} \sum_n z_{n,k} \|x_n - \mu_k\|^2$$

# MapReduce Implementation

Map

$$\mathbb{E}[z_{n,k}] = \frac{\mathcal{N}(x_n | \mu_k, \sigma_k^2) \cdot \pi_k}{\sum_{k'} \mathcal{N}(x_n | \mu_{k'}, \sigma_{k'}^2) \cdot \pi_{k'}}$$



Reduce

$$\pi_k = \frac{1}{N} \sum_n z_{n,k}$$

$$\mu_k = \frac{1}{\sum_n z_{n,k}} \sum_n z_{n,k} \cdot x_n$$

$$\sigma_k^2 = \frac{1}{\sum_n z_{n,k}} \sum_n z_{n,k} \|x_n - \mu_k\|^2$$

# K-Means vs. GMMs

Map

K-Means

Reduce

Compute distance of  
points to centroids

Recompute new centroids

GMM

E-step: compute expectation  
of z indicator variables

M-step: update values of  
model parameters

# Summary

- Hierarchical clustering
  - Difficult to implement in MapReduce
- K-Means
  - Straightforward implementation in MapReduce
- Gaussian Mixture Models
  - Implementation conceptually similar to  $k$ -means, more “bookkeeping”

# Classification



# Supervised Machine Learning

- The generic problem of function induction given sample instances of input and output
  - Classification: output draws from finite discrete labels
  - Regression: output is a continuous value
- Focus here on supervised classification
  - Suffices to illustrate large-scale machine learning

This is not meant to be an exhaustive treatment of machine learning!

# Applications

- Spam detection
- Content (e.g., movie) classification
- POS tagging
- Friendship recommendation
- Document ranking
- Many, many more!

# Supervised Binary Classification

- Restrict output label to be *binary*
  - Yes/No
  - 1/0
- Binary classifiers form a primitive building block for multi-class problems
  - One vs. rest classifier ensembles
  - Classifier cascades

# Limits of Supervised Classification?

- Why is this a big data problem?
  - Isn't gathering labels a serious bottleneck?
- Solution: user behavior logs
  - Learning to rank
  - Computational advertising
  - Link recommendation
- The virtuous cycle of data-driven products

# The Task

- Given  $D = \{(x_i, y_i)\}_i^n$   


↓ label  
↑ (sparse) feature vector

$$x_i = [x_1, x_2, x_3, \dots, x_d]$$

$$y \in \{0, 1\}$$

- Induce  $f : X \rightarrow Y$ 
  - Such that loss is minimized
- $$\frac{1}{n} \sum_{i=0}^n \ell(f(x_i), y_i)$$


↑ loss function
- Typically, consider functions of a parametric form:

$$\arg \min_{\theta} \frac{1}{n} \sum_{i=0}^n \ell(f(x_i; \theta), y_i)$$


↑ model parameters

**Key insight: machine learning as an optimization problem!**  
**(closed form solutions generally not possible)**

# Gradient Descent: Preliminaries

- Rewrite:

$$\arg \min_{\theta} \frac{1}{n} \sum_{i=0}^n \ell(f(\mathbf{x}_i; \theta), y_i) \quad \longrightarrow \quad \arg \min_{\theta} L(\theta)$$

- Compute gradient:

- “Points” to fastest increasing “direction”

$$\nabla L(\theta) = \left[ \frac{\partial L(\theta)}{\partial w_0}, \frac{\partial L(\theta)}{\partial w_1}, \dots, \frac{\partial L(\theta)}{\partial w_d} \right]$$

- So, at any point: \*

$$\mathbf{b} = \mathbf{a} - \gamma \nabla L(\mathbf{a})$$

$$L(\mathbf{a}) \geq L(\mathbf{b})$$

\* caveats

# Gradient Descent: Iterative Update

- Start at an arbitrary point, iteratively update:

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla L(\theta^{(t)})$$

- We have:

$$L(\theta^{(0)}) \geq L(\theta^{(1)}) \geq L(\theta^{(2)}) \dots$$

- Lots of details:

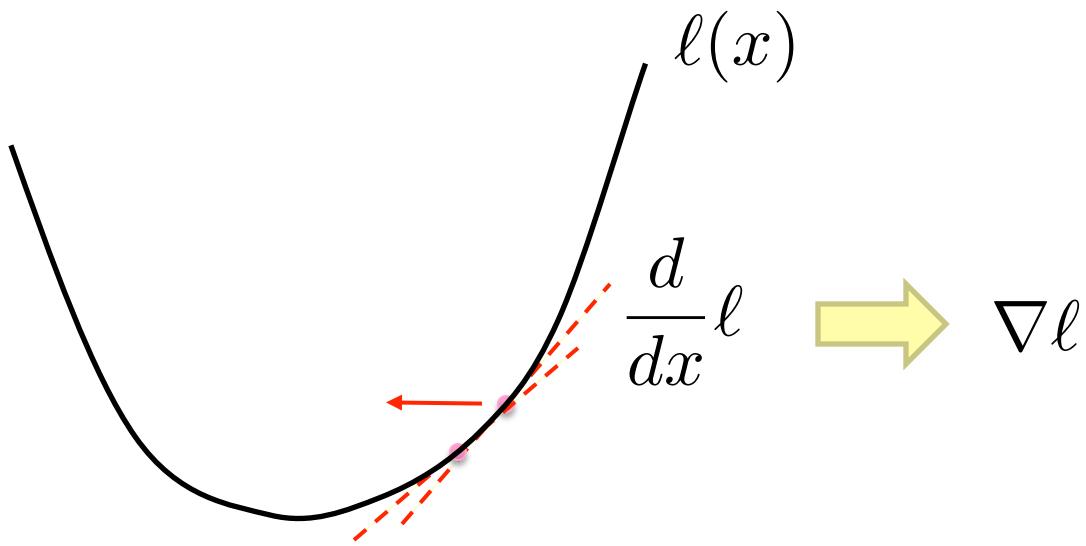
- Figuring out the step size
- Getting stuck in local minima
- Convergence rate
- ...

# Gradient Descent

Repeat until convergence:

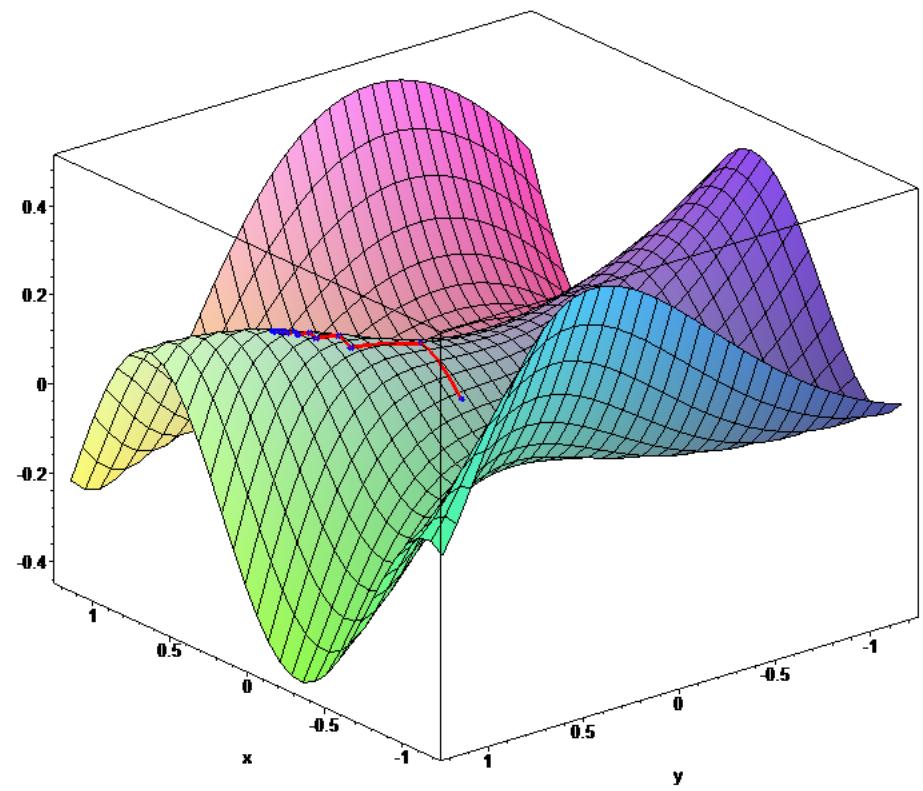
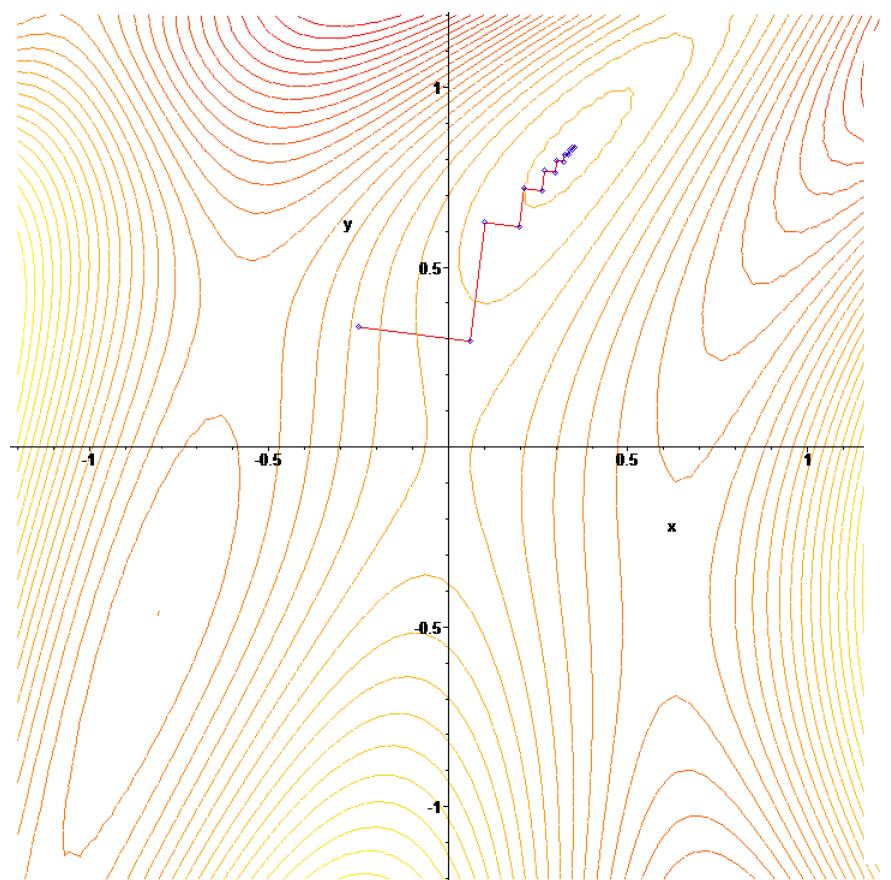
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^n \nabla \ell(f(\mathbf{x}_i; \theta^{(t)}), y_i)$$

# Intuition behind the math...



$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^n \nabla \ell(f(\mathbf{x}_i; \theta^{(t)}), y_i)$$

New weights    Old weights                      Update based on gradient



The background image shows a wide, open landscape with rolling green hills. The sky above is a vibrant blue, filled with large, white, fluffy clouds. The foreground is a mix of green grass and some brown, possibly dry, areas. In the distance, more hills and mountains are visible under the same cloudy sky.

# Gradient Descent

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^n \nabla \ell(f(\mathbf{x}_i; \theta^{(t)}), y_i)$$

# Lots More Details...

- Gradient descent is a “first order” optimization technique
  - Often, slow convergence
  - Conjugate techniques accelerate convergence
- Newton and quasi-Newton methods:
  - Intuition: Taylor expansion

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$$

- Requires the Hessian (square matrix of second order partial derivatives): impractical to fully compute

# Logistic Regression



# Logistic Regression: Preliminaries

- Given  $D = \{(x_i, y_i)\}_i^n$

$$x_i = [x_1, x_2, x_3, \dots, x_d]$$
$$y \in \{0, 1\}$$

- Let's define:

$$f(x; w) : \mathbb{R}^d \rightarrow \{0, 1\}$$

$$f(x; w) = \begin{cases} +1 & \text{if } w \cdot x \geq t \\ -1 & \text{if } w \cdot x < t \end{cases}$$

- Interpretation:

$$\ln \left[ \frac{\Pr(y=1|x)}{\Pr(y=0|x)} \right] = w \cdot x$$

$$\ln \left[ \frac{\Pr(y=1|x)}{1 - \Pr(y=1|x)} \right] = w \cdot x$$

# Relation to the Logistic Function

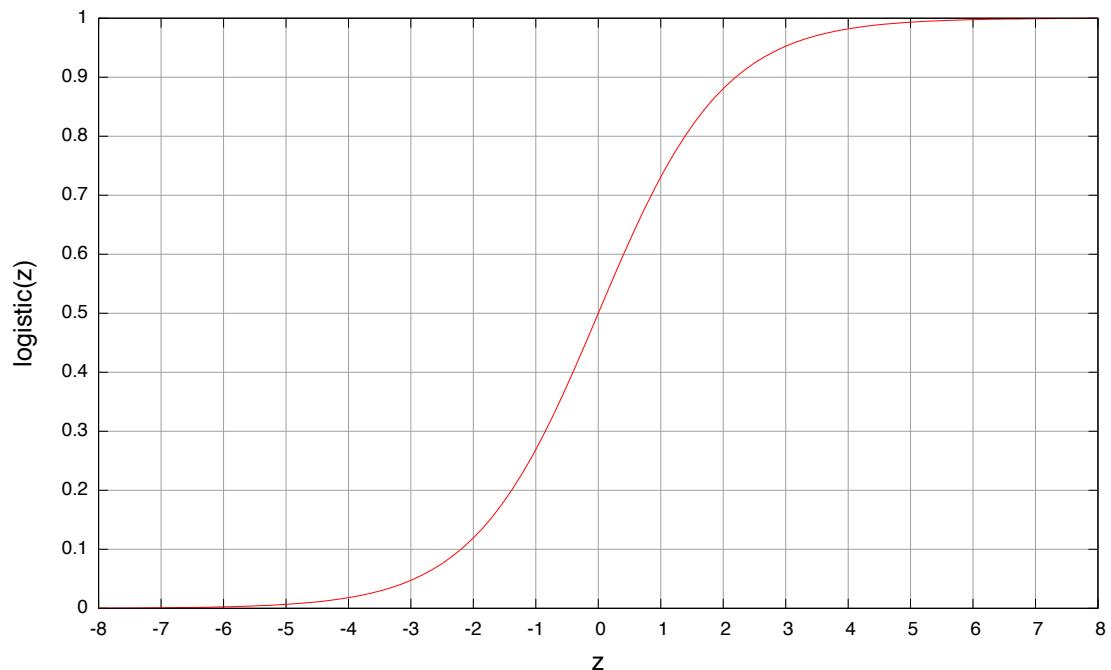
- After some algebra:

$$\Pr(y = 1|x) = \frac{e^{w \cdot x}}{1 + e^{w \cdot x}}$$

$$\Pr(y = 0|x) = \frac{1}{1 + e^{w \cdot x}}$$

- The logistic function:

$$f(z) = \frac{e^z}{e^z + 1}$$



# Training an LR Classifier

- Maximize the conditional likelihood:

$$\arg \max_w \prod_{i=1}^n \Pr(y_i | \mathbf{x}_i, w)$$

- Define the objective in terms of conditional log likelihood:

$$L(w) = \sum_{i=1}^n \ln \Pr(y_i | \mathbf{x}_i, w)$$

- We know  $y \in \{0, 1\}$  so:

$$\Pr(y | \mathbf{x}, w) = \Pr(y = 1 | \mathbf{x}, w)^y [1 - \Pr(y = 0 | \mathbf{x}, w)]^{(1-y)}$$

- Substituting:

$$L(w) = \sum_{i=1}^n \left( y_i \ln \Pr(y_i = 1 | \mathbf{x}_i, w) + (1 - y_i) \ln \Pr(y_i = 0 | \mathbf{x}_i, w) \right)$$

# LR Classifier Update Rule

- Take the derivative:

$$L(\mathbf{w}) = \sum_{i=1}^n \left( y_i \ln \Pr(y_i = 1 | \mathbf{x}_i, \mathbf{w}) + (1 - y_i) \ln \Pr(y_i = 0 | \mathbf{x}_i, \mathbf{w}) \right)$$

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}) = \sum_{i=0}^n \mathbf{x}_i \left( y_i - \Pr(y_i = 1 | \mathbf{x}_i, \mathbf{w}) \right)$$

- General form for update rule:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \gamma^{(t)} \nabla_{\mathbf{w}} L(\mathbf{w}^{(t)})$$

$$\nabla L(\mathbf{w}) = \left[ \frac{\partial L(\mathbf{w})}{\partial w_0}, \frac{\partial L(\mathbf{w})}{\partial w_1}, \dots, \frac{\partial L(\mathbf{w})}{\partial w_d} \right]$$

- Final update rule:

$$\mathbf{w}_i^{(t+1)} \leftarrow \mathbf{w}_i^{(t)} + \gamma^{(t)} \sum_{j=0}^n x_{j,i} \left( y_j - \Pr(y_j = 1 | \mathbf{x}_j, \mathbf{w}^{(t)}) \right)$$

# Lots more details...

- Regularization
- Different loss functions
- ...

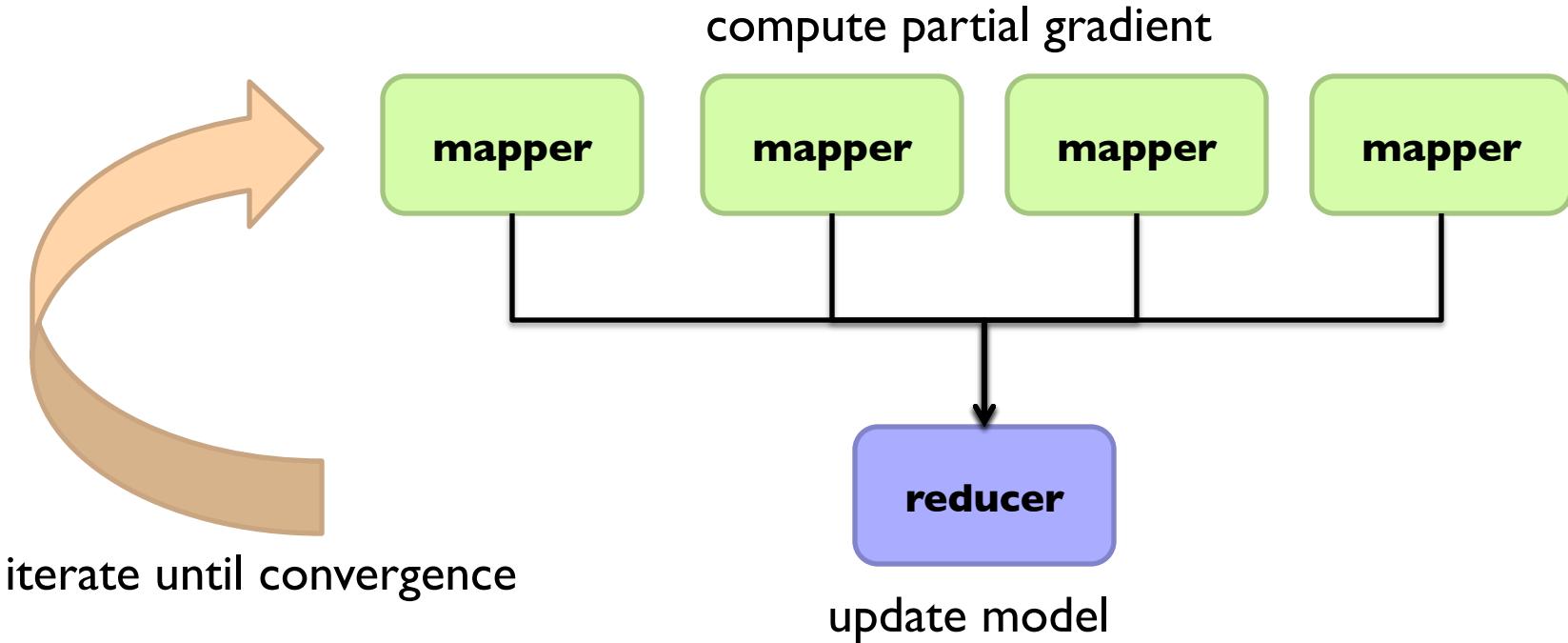
Want more details?  
Take a real machine-learning course!

# MapReduce Implementation

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^n \nabla \ell(f(\mathbf{x}_i; \theta^{(t)}), y_i)$$

mappers

single reducer



# Shortcomings

- Hadoop is bad at iterative algorithms
  - High job startup costs
  - Awkward to retain state across iterations
- High sensitivity to skew
  - Iteration speed bounded by slowest task
- Potentially poor cluster utilization
  - Must shuffle all data to a single reducer
- Some possible tradeoffs
  - Number of iterations vs. complexity of computation per iteration
  - E.g., L-BFGS: faster convergence, but more to compute

# Gradient Descent

A photograph of a vibrant water park featuring a complex network of multi-colored water slides (yellow, blue, green, orange, purple) winding through a steel frame structure. In the foreground, a large yellow splash pool is visible, with several people in swimwear playing in the water. The background shows a clear blue sky with scattered white clouds and some green trees.

# Stochastic Gradient Descent

# Batch vs. Online

## Gradient Descent

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^n \nabla \ell(f(\mathbf{x}_i; \theta^{(t)}), y_i)$$

“batch” learning: update model after considering all training instances

## Stochastic Gradient Descent (SGD)

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla \ell(f(\mathbf{x}; \theta^{(t)}), y)$$

“online” learning: update model after considering each (randomly-selected) training instance

In practice... just as good!

# Practical Notes

- Most common implementation:
  - Randomly shuffle training instances
  - Stream instances through learner
- Single vs. multi-pass approaches
- “Mini-batching” as a middle ground between batch and stochastic gradient descent

We've solved the iteration problem!

What about the single reducer problem?

# Ensembles



# Ensemble Learning

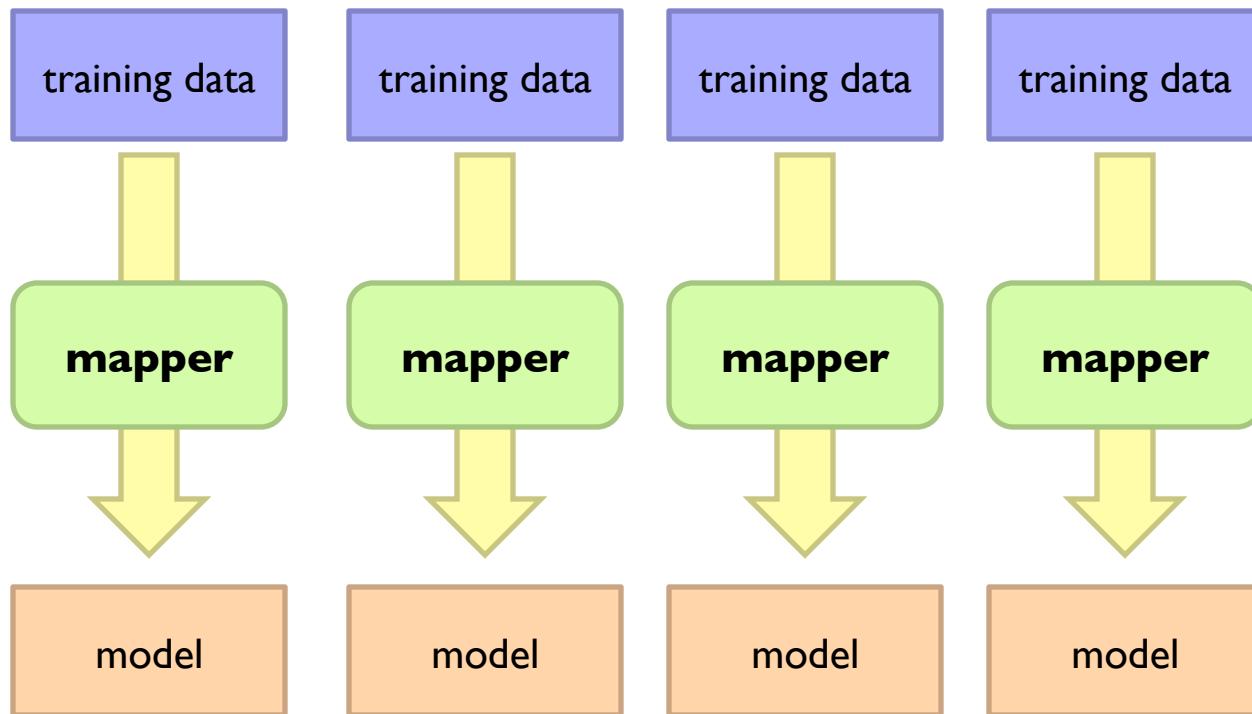
- Learn multiple models, combine results from different models to make prediction
- Why does it work?
  - If errors uncorrelated, multiple classifiers being wrong is less likely
  - Reduces the variance component of error
- A variety of different techniques:
  - Majority voting
  - Simple weighted voting:
$$y = \arg \max_{y \in Y} \sum_{k=1}^n \alpha_k p_k(y|x)$$
  - Model averaging
  - ...

# Practical Notes

- Common implementation:
  - Train classifiers on different input partitions of the data
  - Embarassingly parallel!
- Contrast with bagging
- Contrast with boosting

# MapReduce Implementation

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \nabla \ell(f(\mathbf{x}; \theta^{(t)}), y)$$



# MapReduce Implementation: Details

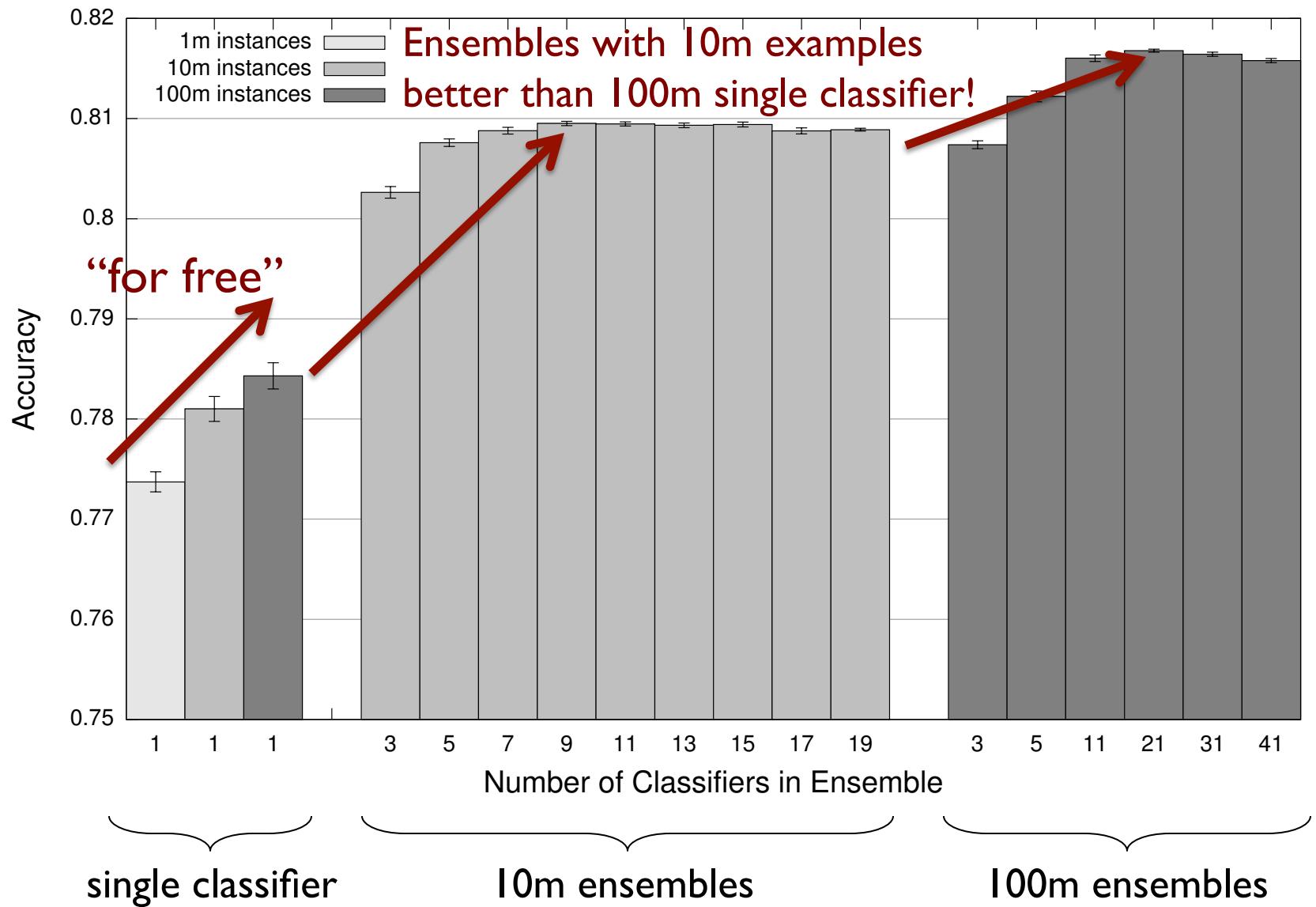
- Shuffling/resort training instances before learning
- Two possible implementations:
  - Mappers write model out as “side data”
  - Mappers emit model as intermediate output

# Sentiment Analysis Case Study

Lin and Kolcz, SIGMOD 2012

- Binary polarity classification: {positive, negative} sentiment
  - Independently interesting task
  - Illustrates end-to-end flow
  - Use the “emoticon trick” to gather data
- Data
  - Test: 500k positive/500k negative tweets from 9/1/2011
  - Training: {1m, 10m, 100m} instances from before (50/50 split)
- Features: Sliding window byte-4grams
- Models:
  - Logistic regression with SGD (L2 regularization)
  - Ensembles of various sizes (simple weighted voting)

Diminishing returns...



# Takeaway Lesson

- Big data “recipe” for problem solving
  - Simple technique
  - Simple features
  - Lots of data
- Usually works very well!

# Today's Agenda

- Personalized PageRank
- Clustering
- Classification

A photograph of a traditional Japanese rock garden. In the foreground, a gravel path is raked into fine, parallel lines. Several large, dark, irregular stones are scattered across the garden. A small, shallow pond is visible in the middle ground, surrounded by more stones and low-lying green plants. In the background, there are more trees and shrubs, and the wooden buildings of a residence are visible behind the garden wall.

Questions?