Search Engine Architecture

7. Classification

Noted slides adapted from: Lin et al.'s Big Data Infrastructure, UMD Spring 2015 with cosmetic changes.

Agenda

- Introduction to machine learning
- Gradient descent
- Logistic regression
- Stochastic gradient descent
- Scaling optimization

Machine Learning: High-Level

- Algorithms that can learn from data
- Supervised
  - Infer a function from labeled training data
- Unsupervised
  - No labels in training data
  - Find interesting patterns or structure

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

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

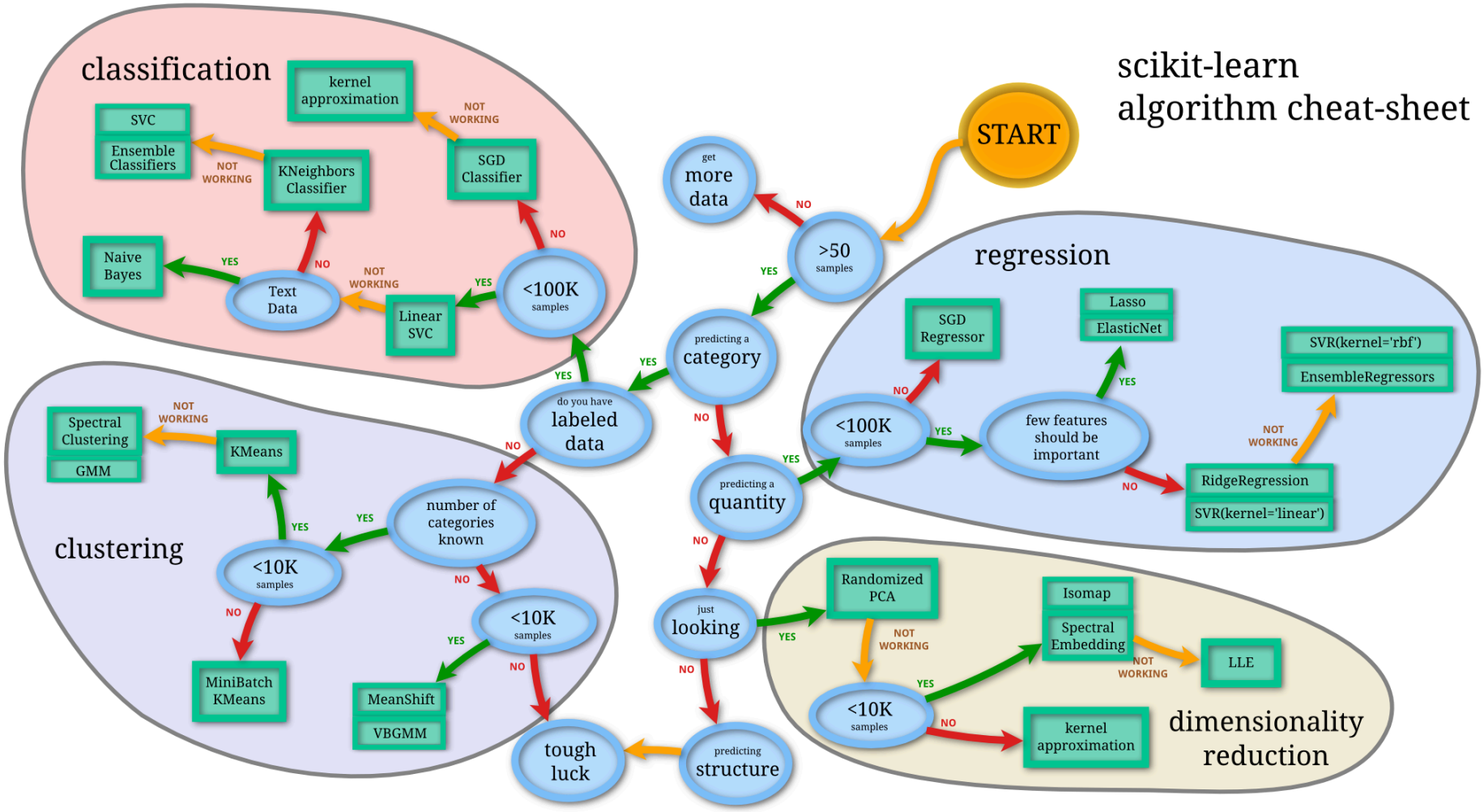
This is not meant to be an exhaustive treatment of machine learning!

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

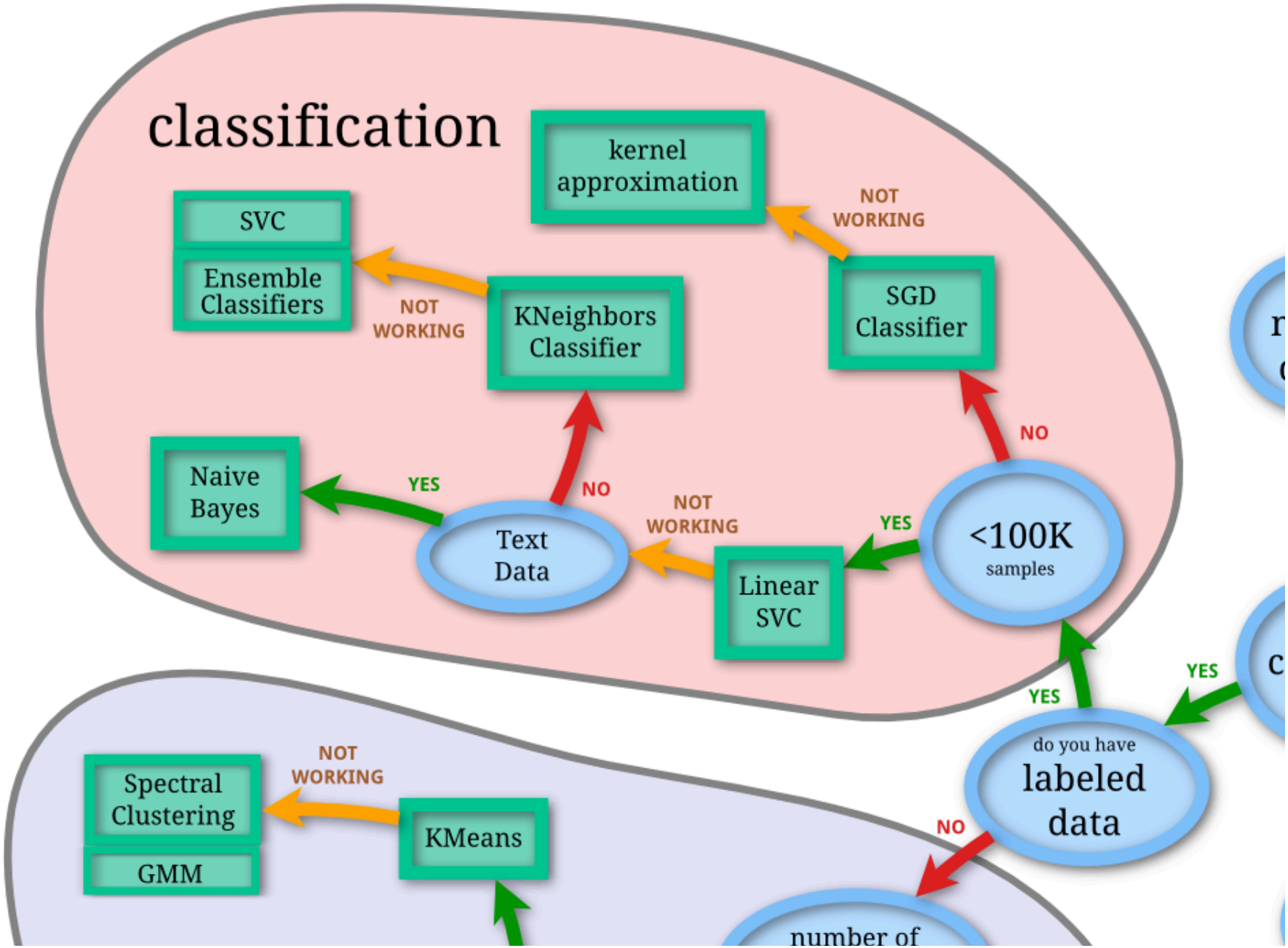
Applications

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.



Source: http://scikit-learn.org/stable/tutorial/machine_learning_map/


classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
NO
NOT WORKING
YES
Text Data
Linear SVC
<100K samples
YES
YES
do you have labeled data
NO
NOT WORKING
Spectral Clustering
GMM
KMeans
number of

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

The Task

Given $D = \{(\mathbf{x}_i, y_i)\}_i^n$

label

(sparse) feature vector

$$\mathbf{x}_i = [x_1, x_2, x_3, \ldots, x_d]$$
$$y \in \{0, 1\}$$

Induce $f : X \rightarrow Y$

Such that loss is minimized

$$\frac{1}{n} \sum_{i=0}^{n} \ell(f(\mathbf{x}_i), y_i)$$

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Gradient Descent: Preliminaries

- Rewrite:

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

- Compute gradient:
  - "Points" to fastest increasing "direction"

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

- So, at any point:

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

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

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:

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

- Lots of details:
  - Figuring out the step size
  - Getting stuck in local minima
  - Convergence rate

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

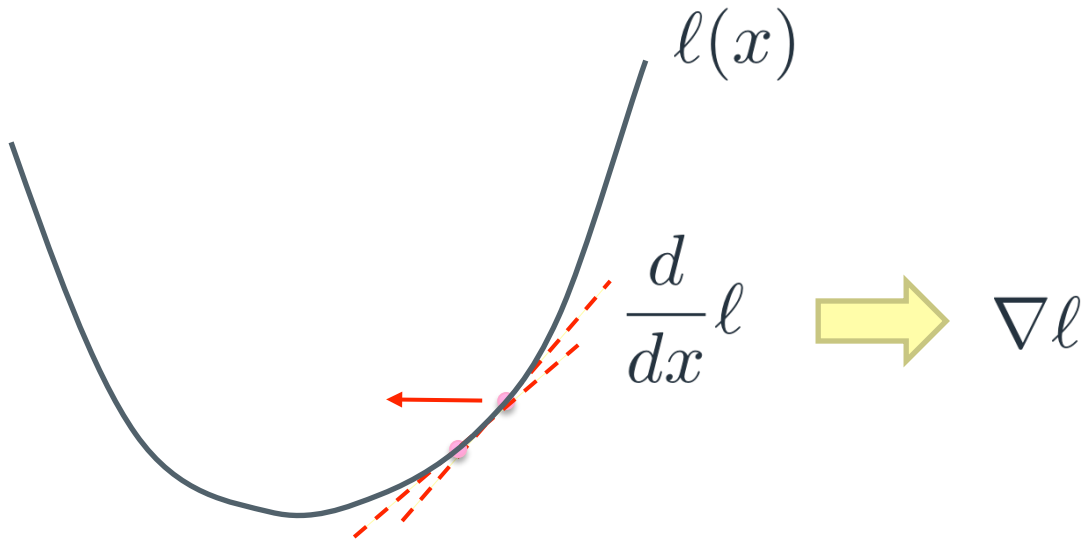
Gradient Descent

Repeat until convergence:

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Intuition behind the math…

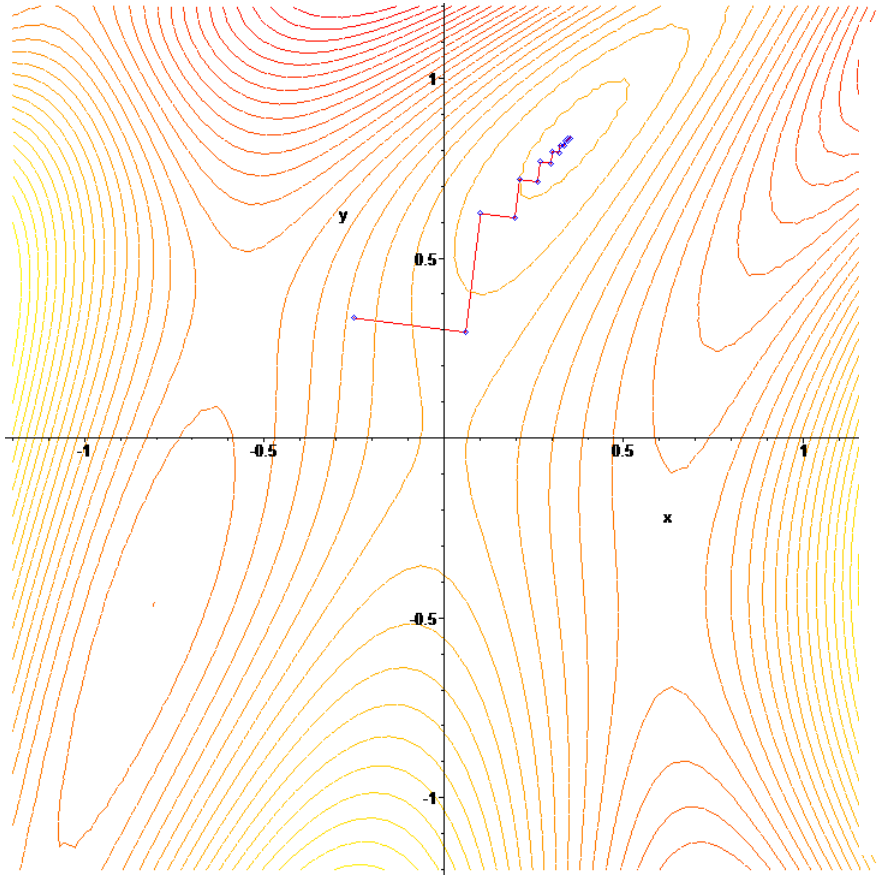


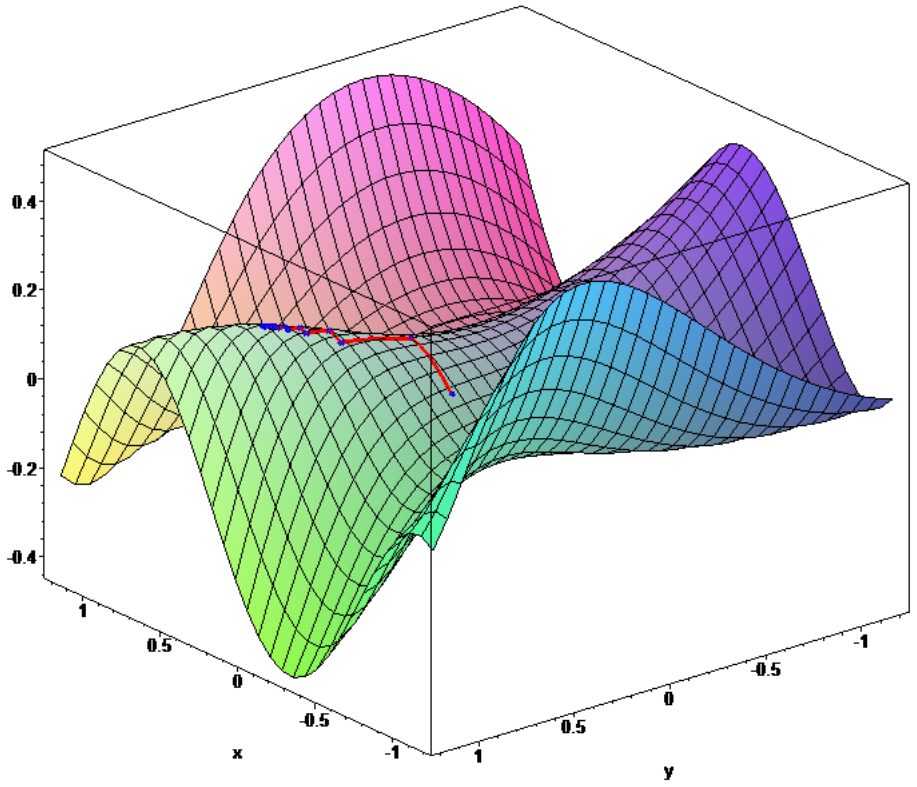
$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^{n} \nabla \ell(f(\mathrm{x}_i; \theta^{(t)}), y_i)$$

New weights Old weights

Update based on gradient

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.





Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.


Gradient Descent
$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \frac{1}{n} \sum_{i=0}^{n} \nabla \ell(f(\mathrm{x}_i; \theta^{(t)}), y_i)$
Source: Wikipedia (Hills)

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Logistic Regression

Logistic Regression: Preliminaries

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

  $\mathrm{x}_i = [x_1, x_2, x_3, \ldots, x_d]$

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

- Let's define:

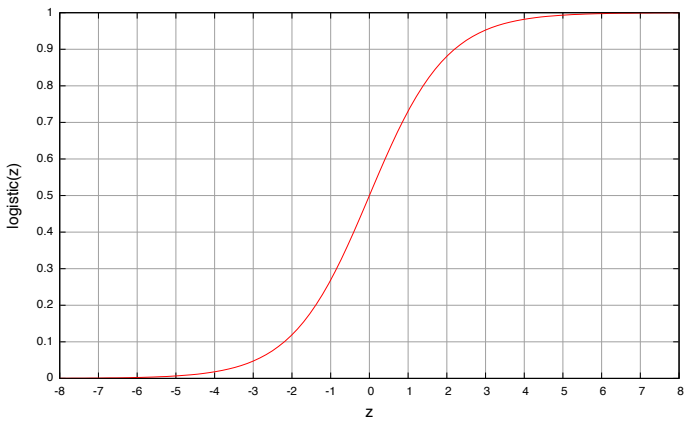
  $f(\mathrm{x}; \mathrm{w}) : \mathbb{R}^d \rightarrow \{0, 1\}$

  $$f(\mathrm{x}; \mathrm{w}) = \begin{cases} 1 \text{ if } \mathrm{w} \cdot \mathrm{x} \geq t \\ 0 \text{ if } \mathrm{w} \cdot \mathrm{x} < t \end{cases}$$

- Interpretation:

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

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



Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Why log odds?

- Dot product alone would be unbounded (not a probability)
  - Solve using odds
- Empirically, we see diminishing returns
  - Solve using log

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Relation to the Logistic Function

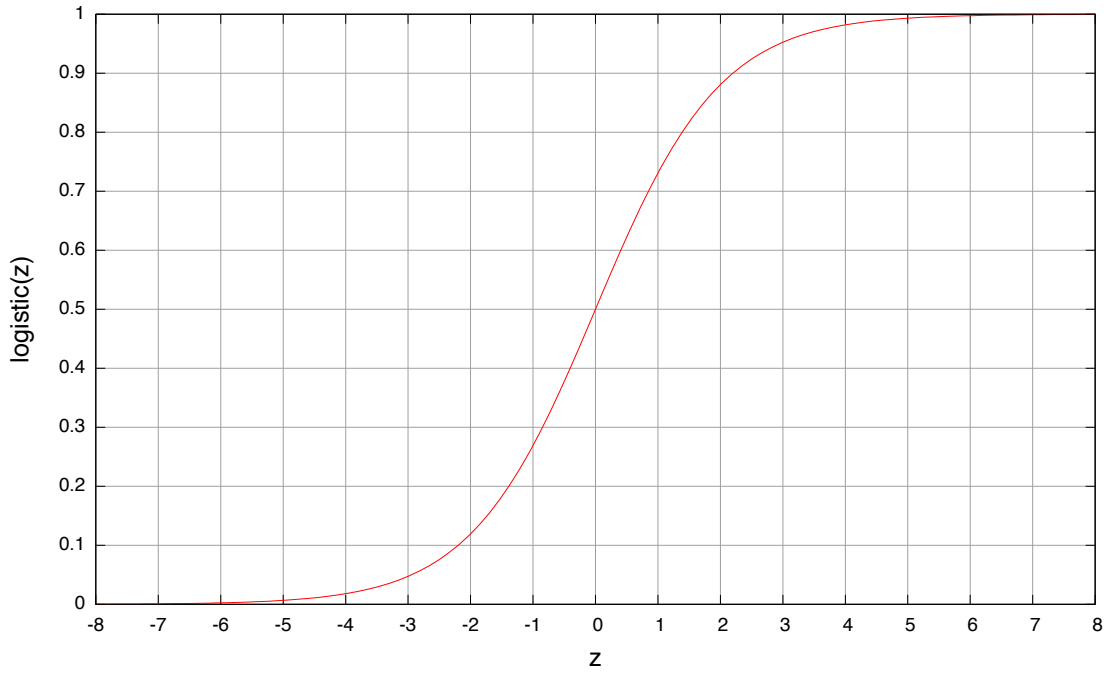
- After some algebra:

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

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

- The logistic function:

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



Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Training an LR Classifier

- Maximize the conditional likelihood:

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

- Define the objective in terms of conditional log likelihood:

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

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

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

  - Substituting:

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

LR Classifier Update Rule

- Take the derivative:

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

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

- General form for update rule:

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

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

- Final update rule:

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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

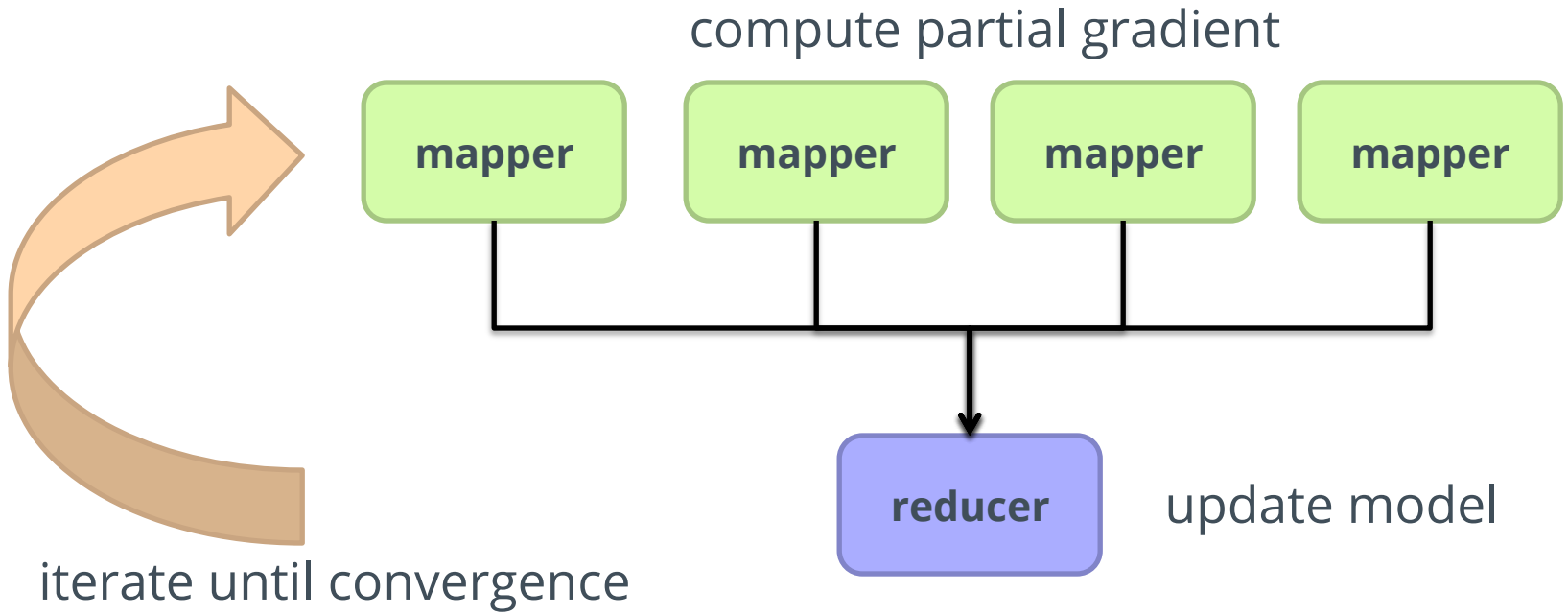
Lots more details…

- Overfitting
  - Model tuned too tightly to training data
- Regularization
  - E.g. add sum of square weights to loss function (L2)
- Different loss functions
  - E.g. hinge, epsilon-insensitive
- Evaluation
  - Accuracy, precision, recall, F1, …
    - Sound familiar?
- Cross-validation

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

MapReduce Implementation

$$\theta^{(t+1)} \leftarrow \theta^{(t)} - \gamma^{(t)} \underbrace{\underbrace{\frac{1}{n} \sum_{i=0}^{n} \nabla \ell(f(\mathrm{x}_i; \theta^{(t)}), y_i)}_{\text{mappers}}}_{\text{single reducer}}$$



Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Shortcomings

- MapReduce is bad at iterative algorithms
  - Hadoop has 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

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Scaling Optimization

Source: Wikipedia (Water Slide)

Batch vs. Online

Gradient Descent

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

"batch" learning: update model after considering all training instances

Stochastic Gradient Descent (SGD)

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

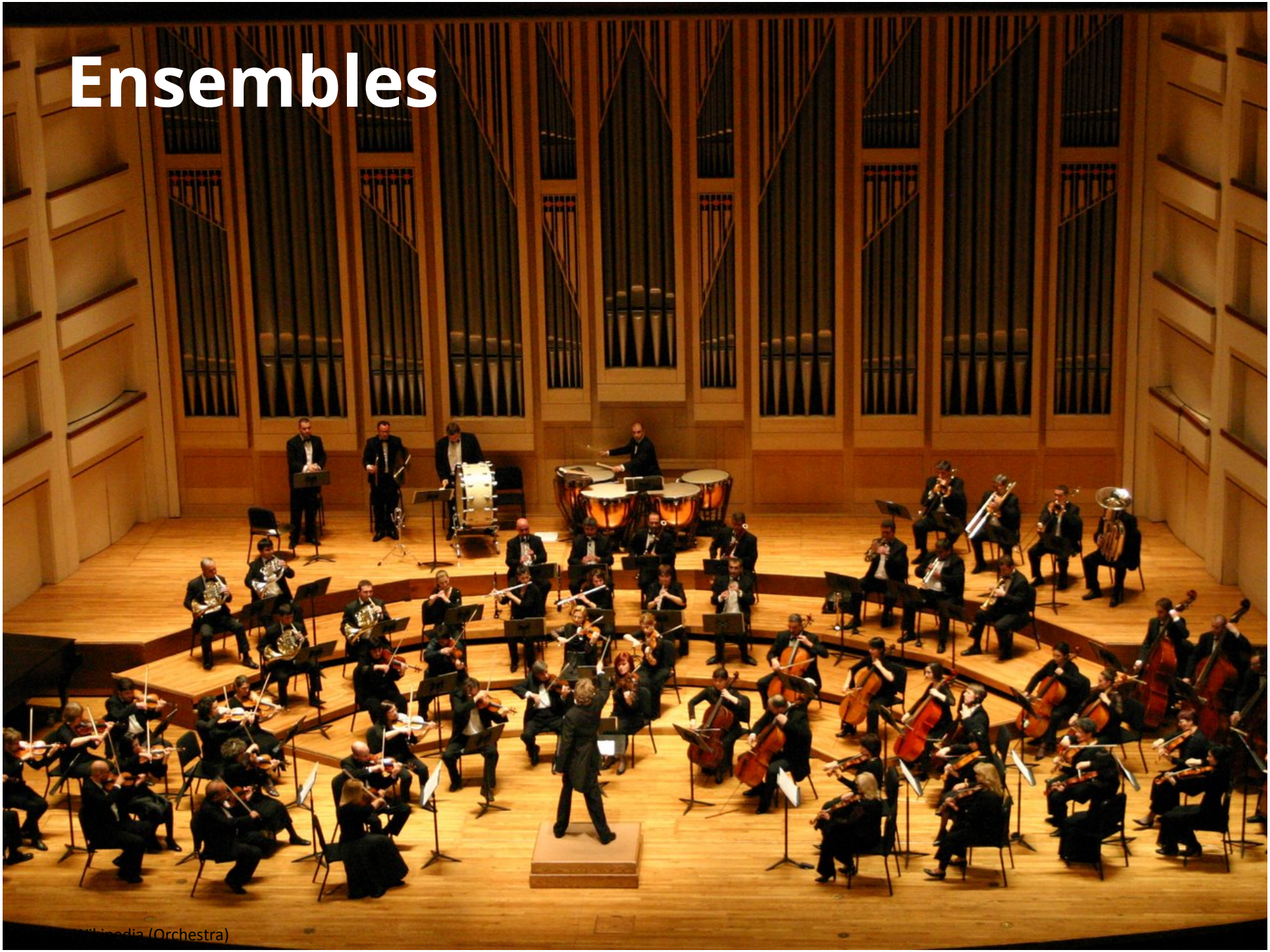
"online" learning: update model after considering *each* (randomly-selected) training instance

"mini-batch" learning: compromise between batch and fully online

In practice… just as good!

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Ensembles



Wikipedia (Orchestra)

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:

$$y = \arg\max_{y \in Y} \sum_{k=1}^{n} \alpha_k p_k(y|\mathrm{x})$$

  - Model averaging

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

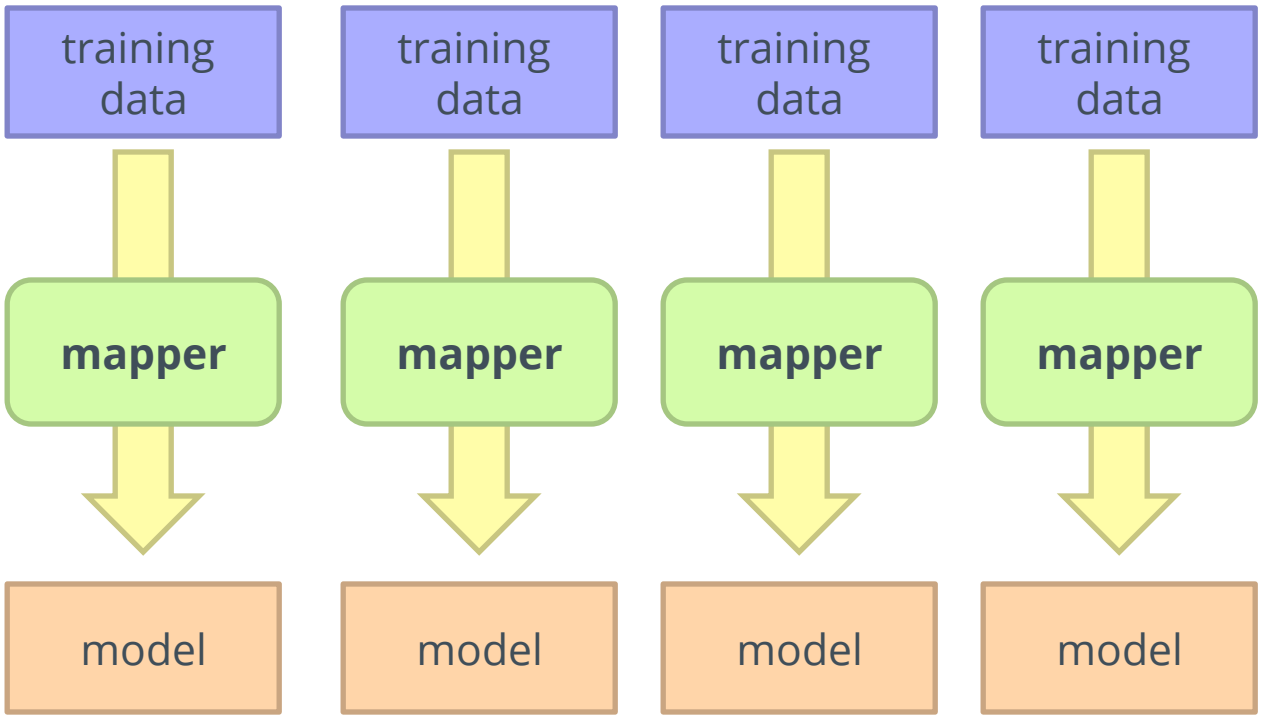
Practical Notes

- Common implementation:
  - Train classifiers on different input partitions of the data
  - Embarrassingly parallel!
- Contrast with bagging
  - Samples chosen randomly with replacement
- Contrast with boosting
  - Shift weight toward samples with errors

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

MapReduce Implementation

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



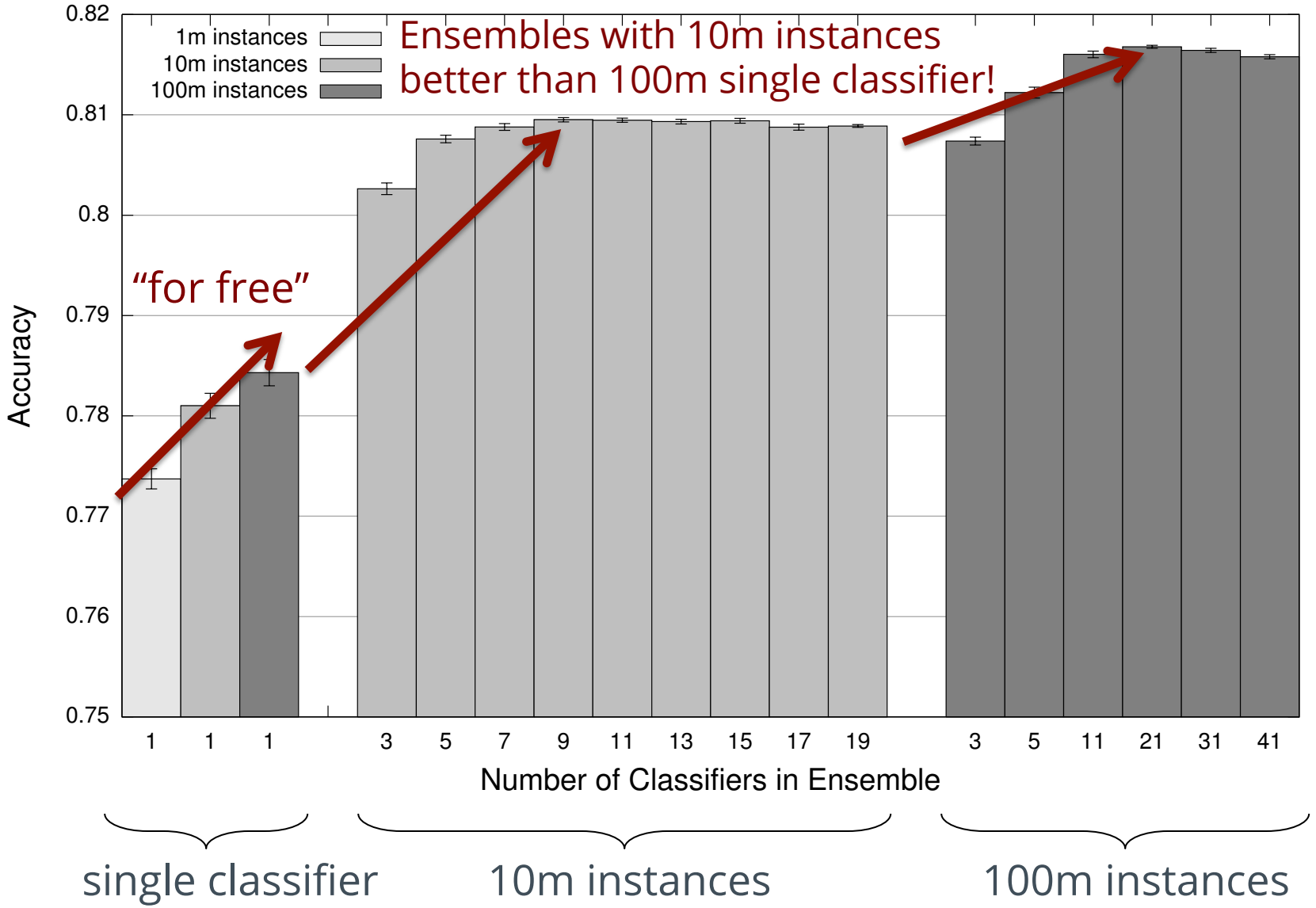
Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

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…



Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Rundown

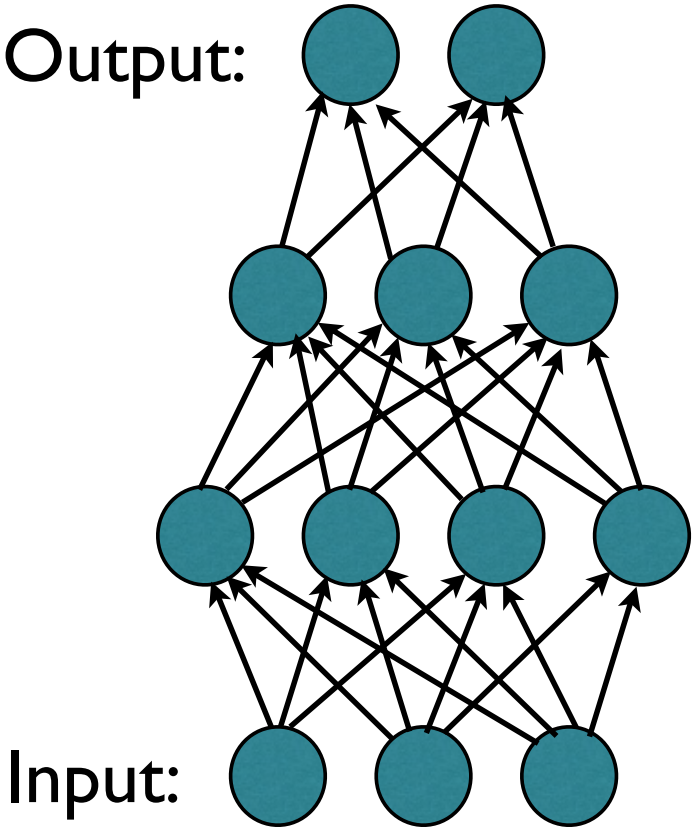
- Ridge regression
  - MSE loss with L2 regularization
- LASSO
  - MSE loss with L1 regularization
- Random forest
  - Bagging, subsampled features
- Gradient boosting
  - Train new models on residuals
- Support vector machine
  - Hinge loss, L2 regularization

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Scaling Stochastic Gradient Descent

(From Jeff Dean's RecSys2014 Keynote)

Neural Networks



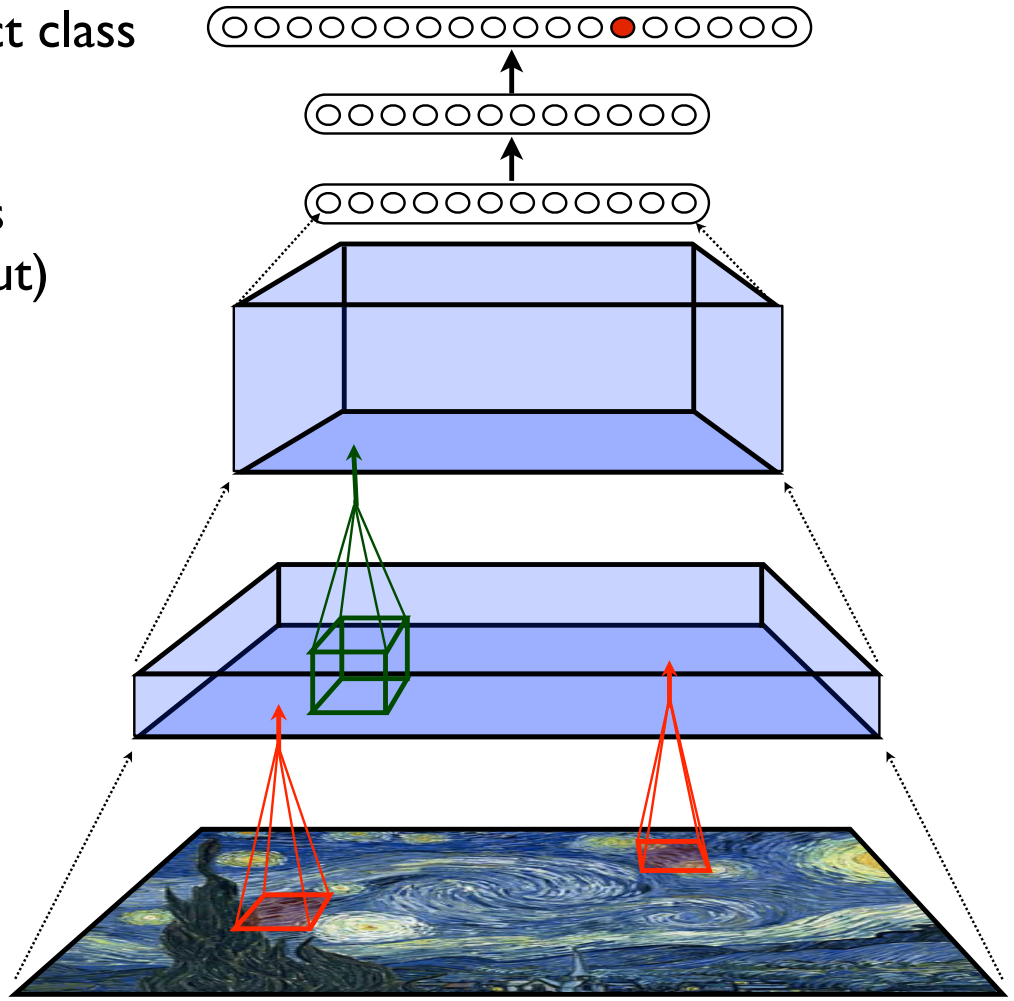
2012 Supervised Vision Model: "AlexNet"

Softmax to predict object class

Fully-connected layers
(and trained w/ DropOut)

Convolutional layers
(same weights used at all
spatial locations in layer)

~60M parameters



Basic architecture developed by Krizhevsky, Sutskever & Hinton

Won 2012 ImageNet challenge with **16.4%** top-5 error rate

Vision models, 2014 edition: GoogLeNet

Module with 6 separate convolutional layers

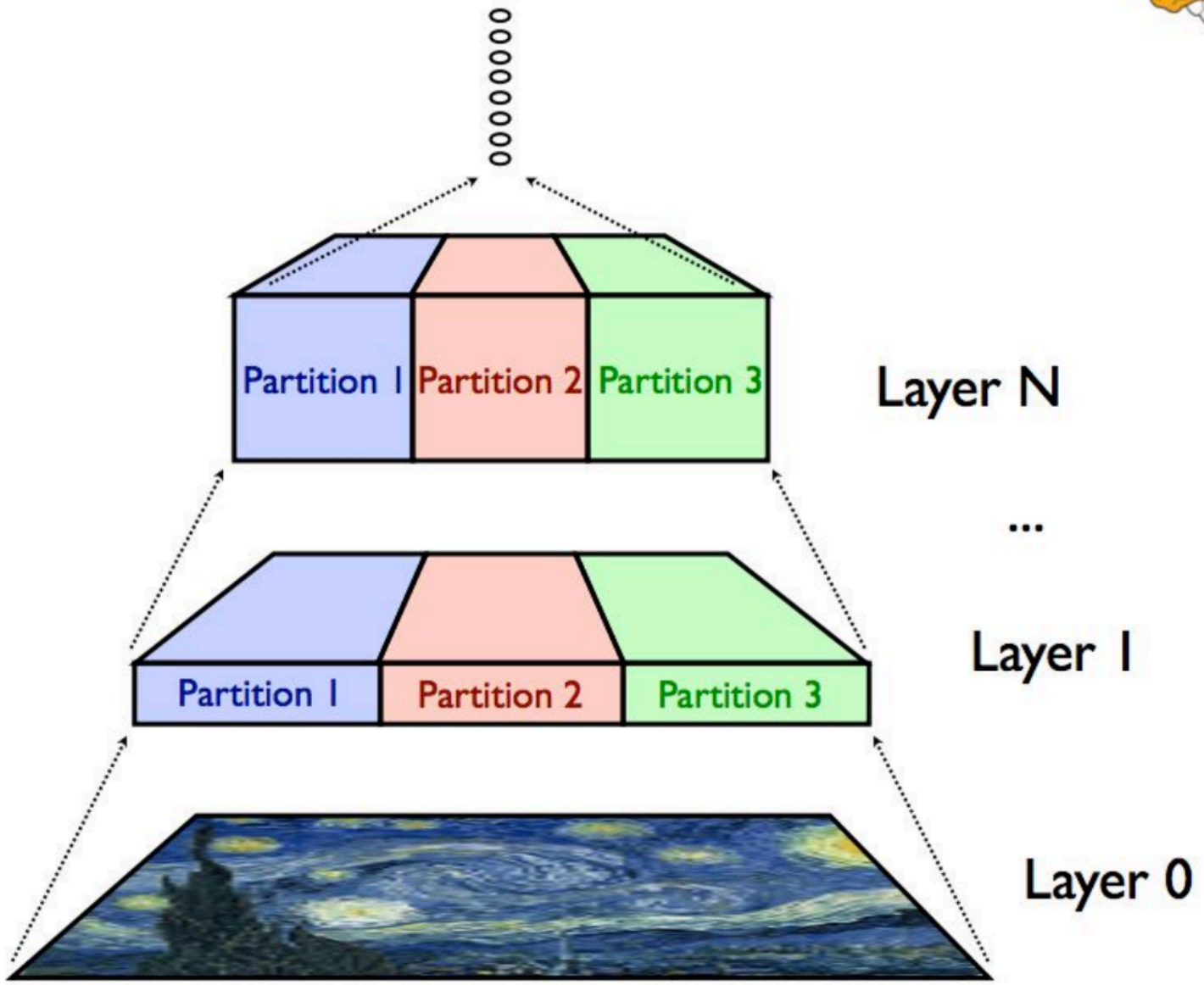
24 layers deep!

No fully-connected layer, so only ~6M parameters (but ~2B floating point operations per example)

Developed by team of Google Researchers:

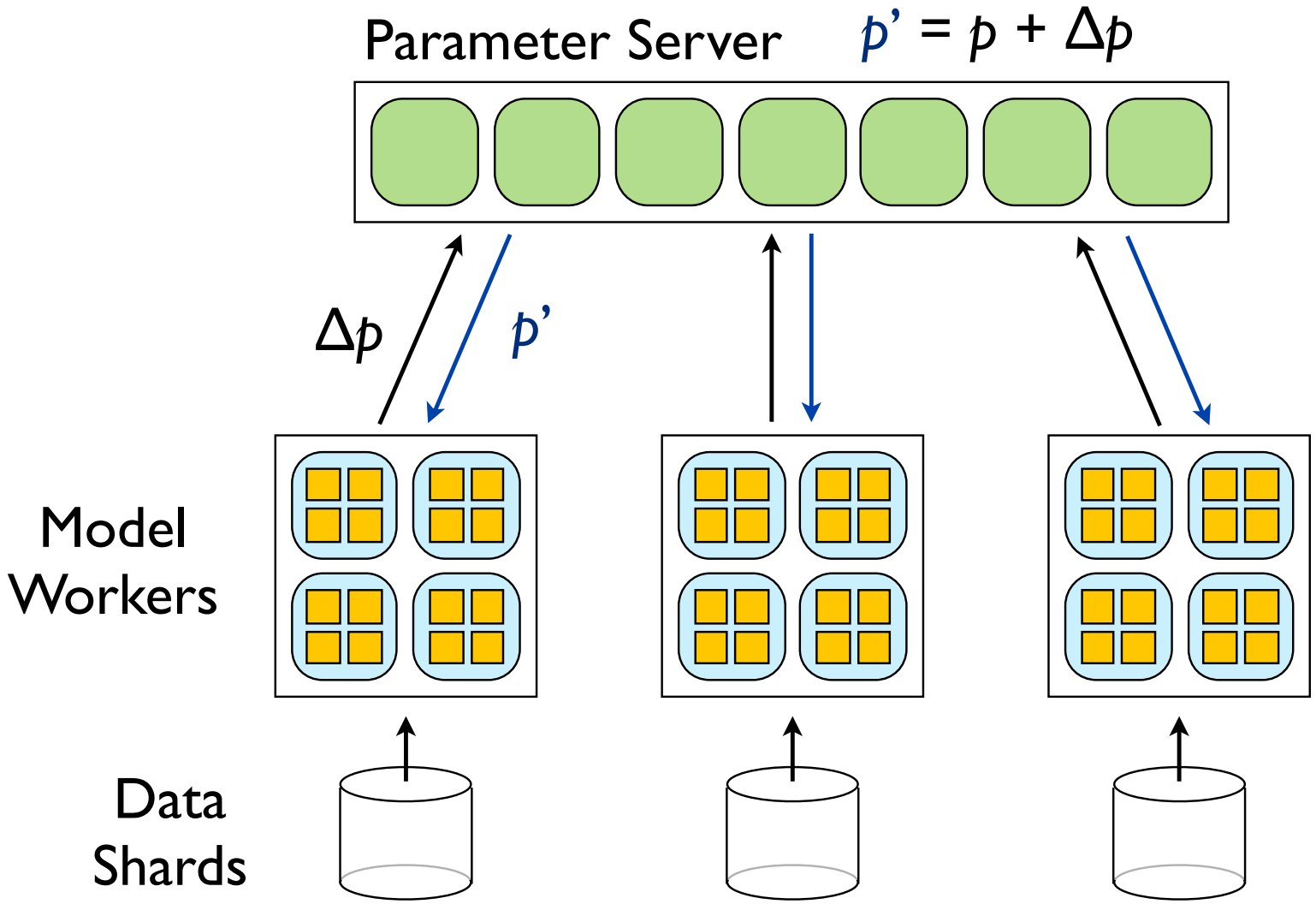
Won 2014 ImageNet challenge with **6.66%** top-5 error rate

Model Parallelism: Partition model across machines



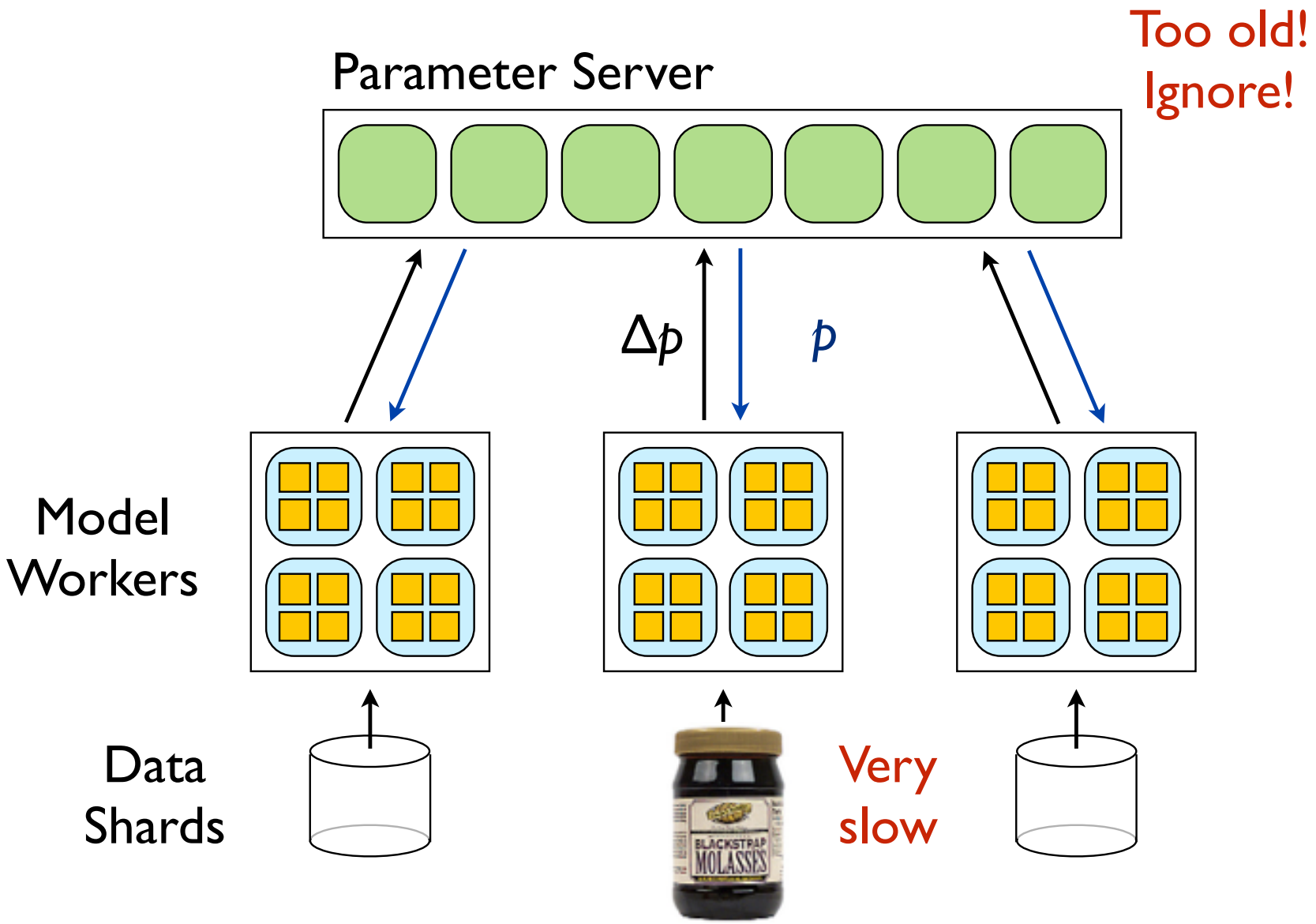
Data Parallelism:

Asynchronous Distributed Stochastic Gradient Descent



Google

Staleness



Google

Questions?