Big Data Analytics
Architectures, Algorithms
and Applications

Part #1: Scalable Big Data Algorithms

Edward Chang
HTC (prior: Google & U. California)

Simon Wu
HTC (Prior: Twitter & Microsoft)
Three Lectures

• Lecture #1: Scalable Big Data Algorithms
  – Scalability issues
  – Key algorithms with application examples

• Lecture #2: Intro to Deep Learning
  – Autoencoder & Sparse Coding
  – Graph models: CNN, MRF, & RBM

• Lecture #3: Analytics Platform [by Simon Wu]
  – Intro to LAMA platform
  – Code lab
Lecture #1 Outline

• Motivations – Why Big Data is not only desirable but also necessary?
• Applications
  – HTC XPRICE Tricorder
  – Context-aware Computing
• Key Parallel Algorithms
  – Frequent Itemset Mining [ACM RS 08]
  – Latent Dirichlet Allocation [TIST 10]
  – Support Vector Machines [MM 01] [MS 03][NIPS 07]
  – Spectral Clustering [PAMI 10]
  – Deep Learning [NIPS 12][ OSDI 14]
• Perspectives and Opportunities
Key References

Open Source Links
Downloaded > 12,000 times

• PSVM,
• PLDA+,
• Parallel Spectral Clustering, and
• Parallel Frequent Pattern Mining
Health Sensors
Machine Learning

• \( X: \) Data
  – \( U: \) Unlabeled data
  – \( L: \) Labeled data

• \( \Phi: \) Learning algorithm
  – Implied hypothesis

• \( f = \Phi (L + U) \)
  – Minimize some error function
  – Regularize parameters to prevent over-fitting

• \( \hat{y} = f (u \in U) \)
Scalability Issue

• $f = \Phi (L)$ — supervised learning
  – Training data can be voluminous
  – A few millions is already too many, though not enough!
  – Training data is scarce
Gene Classification

D = 4026 genes, L = 3, N = 59 cases
Scalability Issues

• $f = \Phi (L)$
  – Training data is too many
  – Training data is scarce

• $f = \Phi (L^* + U)$ semi-supervised learning
  – $L^*$ Collect most useful training data
  – $U$ Use unlabeled data
  – $L^* + U$ is voluminous!

• $f = \Phi (U)$ unsupervised learning
  – NN, CNN, RBM, Deep Learning
Challenges

• Volume, both too large and too small
  – Amount of data ↑
  – Amount of labeled data ↓
  – Dimensionality of data ↑

• Variety

• Velocity
  – Addressed in Lecture #3 with online learning
Why Big Data

• Simply too many data instances? Yes
• But also growing complexity, or dimensionality of data
Why Big Data

• Every learning model is a variant of the nearest neighbor model (distance computation, likelihood)
• An unseen instance needs to get the labels of its neighbors to predict its label
Why Big Data

- $f = .5$, $d = 2$, $\text{NN} = 25$
- When $d$ is large
  
  The volume of $\text{NN} \rightarrow 0$

  $f < 1$, $d > 100$, $f^d \rightarrow 0$

- Curse of dimensionality
More Data vs. Better Algorithms

Banko & Brill, 2001

Figure 2. Learning Curves for Confusable Disambiguation
Applications & Algorithms

• Applications
  – HTC XPRICE Tricorder
  – Context-aware Computing

• Key Algorithms
  – Frequent Itemset Mining [ACM RS 08]
  – Latent Dirichlet Allocation [WWW 09, TIST 10]
  – Support Vector Machines [MM 01, MS 03, NIPS 07, VLDB 14]
  – Spectral Clustering [ECML 08, PAMI 10]
  – Deep Learning [NIPS 12, OSDI 14]

• Perspectives and Opportunities
Fostering disruptive innovation to bring affordable health care to underprivileged

Portable device weight < 5 pounds

Exam 15+ diseases & monitor 5 vital signs

HTC was selected into ten finalists (from 255) on 8/27/2014

Final round: May, 2015
Based on membership so far, and memberships of others

Predict further membership

Diagnosis: Collaborative Filtering

Activities, Food, Symptoms, Diseases, Drugs
Collaborative Filtering

Based on *partially* observed matrix

Predict *unobserved* entries

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>?</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Activities, Food, Symptoms, Diseases, Drugs
FIM-based Prediction

To grow the base, we need association rules

- An association rule: $a, b, c \rightarrow d$
- A Bayesian interpretation: $P(d | a, b, c) = \frac{N(a,b,c,d)}{N(a,b,c)}$
- The key is to count the occurrences (support) of itemsets $N(\ldots)$
FIM-based Prediction

To grow the base, we need association rules

- An association rule: $a, b, c \rightarrow d$
- A Bayesian interpretation: $P(d | a, b, c) = \frac{N(a, b, c, d)}{N(a, b, c)}$
- The key is to count the occurrences (support) of itemsets $N(\ldots)$
FIM-based Prediction
FIM Preliminaries

• Observation 1: If an item A is not frequent, any pattern contains A won’t be frequent [R. Agrawal]
  ➔ use a threshold to eliminate infrequent items
  \{A\} → \{A,B\}

• Observation 2: Patterns containing A are subsets of (or found from) transactions containing A [J. Han]
  ➔ divide-and-conquer: select transactions containing A to form a conditional database (CDB), and find patterns containing A from that conditional database
  \{A, B\}, \{A, C\}, \{A\} → CDB A
  \{A, B\}, \{B, C\} → CDB B

• Observation 3: *Duplicates*!
Preprocessing

• According to Observation 1, we count the support of each item by scanning the database, and eliminate those infrequent items from the transactions.

• According to Observation 3, we sort items in each transaction by the order of descending support value.
Parallel Projection

- According to Observation 2, we construct CDB of item A; then from this CDB, we find those patterns containing A

- How to construct the CDB of A?
  - If a transaction contains A, this transaction should appear in the CDB of A
  - Given a transaction \( \{B, A, C\} \), it should appear in the CDB of A, the CDB of B, and the CDB of C

- Dedup solution: using the order of items:
  - sort \( \{B, A, C\} \) by the order of items \( \rightarrow \langle A, B, C \rangle \)
  - Put \( <> \) into the CDB of A
  - Put \( <A> \) into the CDB of B
  - Put \( <A, B> \) into the CDB of C
Example of Projection of a database into CDBs.
Left: sorted transactions in order of \( f, c, a, b, m, p \)
Right: conditional databases of frequent items
Example of Projection of a database into CDBs.
Left: sorted transactions;
Right: conditional databases of frequent items
Example of Projection

Example of Projection of a database into CDBs.
Left: sorted transactions;
Right: conditional databases of frequent items
Recursive Projections [H. Li, et al. ACM RS 08]

- Recursive projection form a search tree
- Each node is a CDB
- Using the order of items to prevent duplicated CDBs.
- Each level of breath-first search of the tree can be done by a MapReduce iteration.
- Once a CDB is small enough to fit in memory, we can invoke FP-growth to mine this CDB, and no more growth of the subtree.
Projection using MapReduce

<table>
<thead>
<tr>
<th>Map inputs (transactions) key=&quot;&quot;: value</th>
<th>Sorted transactions (with infrequent items eliminated)</th>
<th>Map outputs (conditional transactions) key: value</th>
<th>Reduce inputs (conditional databases) key: value</th>
<th>Reduce outputs (patterns and supports) key: value</th>
</tr>
</thead>
<tbody>
<tr>
<td>facdgimp</td>
<td>fcamp</td>
<td>p: fcamp</td>
<td>m: fcab</td>
<td>m: {fc/ fc/ fc ab}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>m: fcab</td>
<td>b: fcab</td>
<td>mf: 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: fc</td>
<td>a: fc</td>
<td>mc: 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: f</td>
<td>c: f</td>
<td>ma: 3</td>
</tr>
<tr>
<td>abcflmo</td>
<td>fcabm</td>
<td>m: fcab</td>
<td>b: fcab</td>
<td>m f c: 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: fc</td>
<td>a: fc</td>
<td>m f a: 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: f</td>
<td>c: f</td>
<td>m c a: 3</td>
</tr>
<tr>
<td>bfhjo</td>
<td>fb</td>
<td>b: f</td>
<td></td>
<td>m f c a: 3</td>
</tr>
<tr>
<td>bcksp</td>
<td>cb</td>
<td>p: cb</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: fc</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: fc/fc/ fc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>afceelpmn</td>
<td>fcamp</td>
<td>b: c</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>p: fcamp</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>m: fcab</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>a: fc</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: f</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: {f/f/f}</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: f</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

p:{fcam/fcam/cb} p:3, pc:3
Collaborative Filtering

[Confucius or Google QA, VLDB 2010]

Based on *membership* so far, and *memberships* of others

Predict further *membership*

```
<table>
<thead>
<tr>
<th>Users/Labels/Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>
```

Ed Chang @ BigDat 2015
Latent Semantic Analysis

• Search
  – Construct a latent layer for better for semantic matching

• Example:
  – iPhone crack
  – Apple pie

Users/Labels/Documents

• Collaborative Filtering Apps
  – Recommend Users → Docs
  – Recommend Labels → Docs
  – Recommend Photos → Docs

• Predict the ? In the gray cells
The Problem

• Two problems that arise using the vector space model:
  – Synonymy: many ways to refer to the same object, e.g. car and automobile
    • leads to poor recall
  – Polysemy: most words have more than one distinct meaning, e.g. model, python, chip
    • leads to poor precision
The Setting

• Corpus, a set of N documents
  – $D=\{d_1, \ldots ,d_N\}$

• Vocabulary, a set of M words
  – $W=\{w_1, \ldots ,w_M\}$

• A matrix of size $N \times M$ to represent the occurrence of words in documents
  – Called the term-document matrix
Documents, Topics, Words

• A document consists of a number of topics
  – A document is a probabilistic mixture of topics

• Each topic generates a number of words
  – A topic is a distribution over words
  – The probability of the $i^{th}$ word in a document

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_i = j)P(z_i = j)$$
Latent Dirichlet Allocation [M. Jordan 04]

- $\alpha$: uniform Dirichlet $\phi$ prior for per document $d$ topic distribution (corpus level parameter)
- $\beta$: uniform Dirichlet $\phi$ prior for per topic $z$ word distribution (corpus level parameter)
- $\theta_d$ is the topic distribution of doc $d$ (document level)
- $z_{dj}$ the topic if the $j^{th}$ word in $d$, $w_{dj}$ the specific word (word level)

M documents
Each $Nm$ words
$K$ topics
Example

Mixture topics

Mixture weights

Bayesian approach: use priors

Mixture weights $\sim$ Dirichlet($\alpha$)

Mixture topics $\sim$ Dirichlet($\beta$)
Inverting ("fitting") the model

DOCUMENT 1: money, bank, bank, loan, river, stream, bank, money, river, bank, money, bank, loan, money, stream, bank, money, bank, loan, bank, money, stream

DOCUMENT 2: river, stream, bank, stream, bank, money, loan, river, stream, loan, bank, river, bank, bank, stream, river, loan, bank, stream, bank, money, river, stream, loan, bank, river, bank, money, bank, stream, river, bank, stream, bank, money
LDA Gibbs Sampling: Inputs And Outputs

Inputs:

1. **Training data**: documents as bags of words
2. **Parameter**: the number of topics

Outputs:

1. A co-occurrence matrix of topics and documents
2. A co-occurrence matrix of topics and words
Example Application
corpus data

- TASA corpus: text from first grade to college
  - representative sample of text

- 26,000+ word types (stop words removed)
- 37,000+ documents
- 6,000,000+ word tokens
Example Topics

- 37K docs, 26K words
- 1700 topics, e.g.:
Polysemy
Three documents with the word “play” (numbers & colors → topic assignments)

A Play is written to be performed on a stage before a live audience or before motion picture or television cameras (for later viewing by large audiences). A Play is written because playwrights have something.

He was listening to music coming from a passing riverboat. The music had already captured his heart as well as his ear. It was jazz. Bix beiderbecke had already had music lessons. He wanted to play the cornet. And he wanted to play jazz.

Jim plays the game. Jim likes the game for one. The game book helps jim. Don comes into the house. Don and jim read the game book. The boys see a game for two. The two boys play the game.
LDA Gibbs Sampling: Inputs And Outputs

**Inputs:**

1. **Training data**: documents as bags of words
2. **Parameter**: the number of topics

**Outputs:**

1. A co-occurrence matrix of topics and documents
2. A co-occurrence matrix of topics and words
LDA Gibbs Sampling: Inputs And Outputs

Inputs:

1. **Training data**: documents as bags of words
2. **Parameter**: the number of topics

Outputs:

1. A co-occurrence matrix of topics and documents
2. A co-occurrence matrix of topics and words
PLDA+ --- enhanced parallel LDA
[ACM TIST 2010]

- PLDA is restricted by memory: Topic-word matrix has to fit into memory
- WK matrix must be globally synchronized
- Restricted by Amdahl’s Law: communication costs too high, e.g., 1/10 cost spent in IOs caps speedup to
Work Order Example

• Words a, b, c, a, c, d, e, f, a, c, b
• Words a, a, a, b, b, c, c, c, d, e, f
• Word sorting per node to improve locality
• Word bundles to balance workload and increase CPU computation unit to mask IO time
PLDA+ --- enhanced parallel LDA

• Take advantage of bag of words modeling: each Pw machine processes vocabulary in a word order
• Pipelining: fetching the updated topic distribution matrix while doing Gibbs sampling
• Ensure $t_f + t_u < t_s$ (4(A) is good, 4(B) suboptimal)

Fig. 4: Pipeline-based Gibbs Sampling in PLDA*. (A): $t_s \geq t_f + t_u$. (B): $t_s < t_f + t_u$. 

1/26/2015 Ed Chang @ BigDat 2015
## MapReduce VS. MPI?

<table>
<thead>
<tr>
<th></th>
<th>MapReduce</th>
<th>MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFS/IO and task rescheduling overhead between iterations</td>
<td>Yes</td>
<td>No +1</td>
</tr>
<tr>
<td>Flexibility of computation model</td>
<td>AllReduce only +0.5</td>
<td>Flexible +1</td>
</tr>
<tr>
<td>Efficient AllReduce</td>
<td>Yes +1</td>
<td>Yes +1</td>
</tr>
<tr>
<td>Recover from faults between iterations</td>
<td>Yes +1</td>
<td>Apps +0.5</td>
</tr>
<tr>
<td>Recover from faults within each iteration</td>
<td>Yes +1</td>
<td>Apps +0.5</td>
</tr>
<tr>
<td>Final Score for scalable machine learning</td>
<td>3.5</td>
<td>5</td>
</tr>
</tbody>
</table>
Speedup

1,500x using 2,000 machines

Ed Chang @ BigDat 2015
Applications & Algorithms

• Applications
  – HTC XPRICE Tricorder
  – Context-aware Computing

• Key Algorithms
  – Frequent Itemset Mining [ACM RS 08]
  – Latent Dirichlet Allocation [WWW 09, TIST 10]
  – Support Vector Machines [MM 01, MS 03, NIPS 07, VLDB 14]
    – Spectral Clustering [ECML 08, PAMI 10]
    – Deep Learning [NIPS 12, OSDI 14]

• Perspectives and Opportunities
Melanoma vs. Nevus
Key Technical Challenges

• Acquire labeled data (most data are unlabeled)
• Formulate distance function
• Train a classifier
• Classify unlabeled data
  – Fast
  – Low power consumption
Models

• Generative Models
  – Model distribution
  – One each class
  – Look for maximum likelihood
  – Need a lot of training data

• Discriminative Models
  – Model class boundaries
  – Ignore distribution
  – Support Vector Machines (SVMs)
IR $\rightarrow$ A Classification Problem

Use SVMActive to Acquire Training Data
IR $\rightarrow$ A Classification Problem

Most Data are Unlabeled
Step #1: Solicit Labels
Via Active Learning [MM 01]
Step #2: Compute Boundary
Step #3: Identify Useful Samples
Step #4: Solicit Feedback
Step #5: Refine Boundary
Step #6: Identify Samples
Step #7: User Feedback
Step #8: Refine Boundary
Step #9: Classify Data
Observations

• Identify good samples
• Collect diversified samples
• Provide useful results much earlier
• Eventually, if all data have been labeled, classification accuracy converges

• Next, how to quantify similarity?
  – One way is to hand-craft a kernel matrix
  – The other is to learn a good manifold
Similarity?
Distance Function Formulation

Ed Chang @ BigDat 2015
Group by Proximity
**Group by Proximity**

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1</td>
<td>.7</td>
<td>.4</td>
<td>.3</td>
<td>.7</td>
<td>.6</td>
<td>.2</td>
<td>.1</td>
</tr>
<tr>
<td>x2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Group by Shape

1  2  3  4

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>o</td>
<td>o</td>
<td></td>
</tr>
</tbody>
</table>

5  6  7  8
# Group by Shape

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1</td>
<td>.7</td>
<td>.7</td>
<td>.7</td>
<td>.2</td>
<td>.2</td>
<td>.2</td>
<td>.2</td>
</tr>
<tr>
<td>x2</td>
<td></td>
<td>1</td>
<td>.7</td>
<td>.7</td>
<td>.2</td>
<td>.2</td>
<td>.2</td>
<td>.2</td>
</tr>
<tr>
<td>x3</td>
<td></td>
<td></td>
<td>1</td>
<td>.7</td>
<td>.2</td>
<td>.2</td>
<td>.2</td>
<td>.2</td>
</tr>
<tr>
<td>x4</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>.2</td>
<td>.2</td>
<td>.2</td>
<td>.2</td>
</tr>
<tr>
<td>x5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>.7</td>
<td>.7</td>
<td>.7</td>
</tr>
<tr>
<td>x6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>.7</td>
<td>.7</td>
</tr>
<tr>
<td>x7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>.7</td>
</tr>
<tr>
<td>x8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
## Group by Color

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1</td>
<td>.7</td>
<td>.3</td>
<td>.3</td>
<td>.3</td>
<td>.2</td>
<td>.2</td>
<td>.7</td>
</tr>
<tr>
<td>x2</td>
<td>1</td>
<td>.3</td>
<td>.3</td>
<td>.3</td>
<td>.3</td>
<td>.3</td>
<td>.7</td>
<td>.7</td>
</tr>
<tr>
<td>x3</td>
<td>1</td>
<td>.7</td>
<td>.7</td>
<td>.7</td>
<td>.3</td>
<td>.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>1</td>
<td>.7</td>
<td>.7</td>
<td>.3</td>
<td>.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x5</td>
<td>1</td>
<td>.7</td>
<td>.3</td>
<td>.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x6</td>
<td>1</td>
<td>.3</td>
<td>.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x7</td>
<td>1</td>
<td>.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Similarity?
Distance Function Formulation

1 2 3 4

5 6 7 8

Ed Chang @ BigDat 2015
<table>
<thead>
<tr>
<th>NORMAL</th>
<th>CANCEROUS</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Normal Moles" /></td>
<td><img src="image2" alt="Cancerous Moles" /></td>
</tr>
</tbody>
</table>
| **“A” IS FOR ASYMMETRY**  
- If you draw a line through the middle of the mole, the halves of a melanoma won’t match in size. | ![Normal Moles](image1) | ![Cancerous Moles](image2) |
| **“B” IS FOR BORDER**  
- The edges of an early melanoma tend to be uneven, crusty or notched. | ![Normal Moles](image1) | ![Cancerous Moles](image2) |
| **“C” IS FOR COLOR**  
- Healthy moles are uniform in color. A variety of colors, especially white and/or blue, is bad. | ![Normal Moles](image1) | ![Cancerous Moles](image2) |
| **“D” IS FOR DIAMETER**  
- Melanomas are usually larger in diameter than a pencil eraser, although they can be smaller. | ![Normal Moles](image1) | ![Cancerous Moles](image2) |
| **“E” IS FOR EVOLVING**  
- When a mole changes in size, shape or color, or begins to bleed or scab, this points to danger. | ![Normal Moles](image1) | ![Cancerous Moles](image2) |
## Group by Labels

Update the Kernel Matrix

<table>
<thead>
<tr>
<th></th>
<th>x1</th>
<th>x2</th>
<th>x3</th>
<th>x4</th>
<th>x5</th>
<th>x6</th>
<th>x7</th>
<th>x8</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>1</td>
<td>.7</td>
<td>.3</td>
<td>.3</td>
<td>.3</td>
<td>.2</td>
<td>.2</td>
<td>.7</td>
</tr>
<tr>
<td>x2</td>
<td>1</td>
<td>.7</td>
<td>.3</td>
<td>.3</td>
<td>.3</td>
<td>.2</td>
<td>.7</td>
<td></td>
</tr>
<tr>
<td>x3</td>
<td>1</td>
<td>.7</td>
<td>.7</td>
<td>.7</td>
<td>.3</td>
<td>.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x4</td>
<td>1</td>
<td>.7</td>
<td>.7</td>
<td>.3</td>
<td>.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x5</td>
<td>1</td>
<td>.7</td>
<td>.3</td>
<td>.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x6</td>
<td>1</td>
<td>.3</td>
<td>.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x7</td>
<td>1</td>
<td>.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Ed Chang @ BigDat 2015
Similarity Theories

• Objects are similar in all respects (Richardson 1928)
• Objects are similar in some respects (Tversky 1977)
• Similarity is a process of determining respects, rather than using predefined respects (Goldstone 94)
Traditional Similarity Theories

• Objects are similar in all or some respects

• Minkowski Function
  \[ D = \left( \sum_{i=1..M} (p_i - q_i)^n \right)^{1/n} \]

• Weighted Minkowski Function
  \[ D = \left( \sum_{i=1..M} w_i (p_i - q_i)^n \right)^{1/n} \]

• Same \( w \) is imposed to app pairs of objects \( p \) and \( q \)
DPF: Dynamic Partial Function
[B. Li, E. Chang, et al, MM Systems 2013]

• Similarity is a process of determining respects, rather than using predefined respects (Goldstone 94)

\[
\begin{align*}
a_1 &= [0 | | | 0 ... 0 ] \\
a_2 &= [ | | | 0 0 ... 0 ] \\
a_3 &= [ | 0 | | 0 ... 0 ] \\
\vdots \\
a_m &= [ 0 0 0 | | ... 0 ] \\
\end{align*}
\]
Lecture #2 Preview

• How can deep learning help learn features?
• Sparse coding confirms DPF on the right track

• For now, need to speed up the kernel method
  – Suppose we have a kernel matrix representing pairwise similarity of data instances
  – How to speed up SVM learning w/ kernel?
SVM Bottlenecks

Time consuming – 1M dataset, 8 days

Memory consuming – 1M dataset, 10G
Matrix Factorization Alternatives

<table>
<thead>
<tr>
<th>Factorization</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>QR</td>
<td>$O\left(\frac{4}{3}n^3\right)$</td>
</tr>
<tr>
<td>LU</td>
<td>$O\left(\frac{2}{3}n^3\right)$</td>
</tr>
<tr>
<td>Cholesky</td>
<td>$O\left(\frac{1}{3}n^3 + 2n^2\right)$</td>
</tr>
<tr>
<td>LDLT</td>
<td>$O\left(\frac{1}{3}n^3\right)$</td>
</tr>
<tr>
<td>Incomplete Cholesky</td>
<td>$O\left(p^2n\right)$</td>
</tr>
<tr>
<td>Kronecker</td>
<td>$O\left(2n^2\right)$</td>
</tr>
</tbody>
</table>
PSVM [E. Chang, et al, NIPS 07]

- Column-based Incomplete Cholesky Factorization (ICF)
  - Slower than row-based on single machine
  - Parallelizable on multiple machines

- Changing IPM computation order to achieve parallelization
  - \( D = (A \times B) \times C \)
  - \( D = A \times (B \times C) \)
Parallelized and Incremental SVM

Raw Data  Kernel Matrix  ICF  Matrix Multiplication

Incremental Data  Incremental Kernel Matrix  Incremental ICF  Incremental Matrix Multiplication

1/26/2015  Ed Chang @ BigDat 2015
Incomplete Cholesky Factorization (ICF)
Parallelized and Incremental SVM
Matrix Product

\[ \begin{array}{c}
p \times n \\
\times \\
n \times p \\
\end{array} \times 
\begin{array}{c}
p \times p \\
\end{array} = 
\begin{array}{c}
p \times p \\
\end{array} \]
# Speedup

<table>
<thead>
<tr>
<th>Machines</th>
<th>Image (200k)</th>
<th>CoverType (500k)</th>
<th>RCV (800k)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (s)</td>
<td>Speedup</td>
<td>Time (s)</td>
</tr>
<tr>
<td>10</td>
<td>1,958 (9)</td>
<td>10*</td>
<td>16,818 (442)</td>
</tr>
<tr>
<td>30</td>
<td>572 (8)</td>
<td>34.2</td>
<td>5,591 (10)</td>
</tr>
<tr>
<td>50</td>
<td>473 (14)</td>
<td>41.4</td>
<td>3,598 (60)</td>
</tr>
<tr>
<td>100</td>
<td>330 (47)</td>
<td>59.4</td>
<td>2,082 (29)</td>
</tr>
<tr>
<td>150</td>
<td>274 (40)</td>
<td>71.4</td>
<td>1,865 (93)</td>
</tr>
<tr>
<td>200</td>
<td>294 (41)</td>
<td>66.7</td>
<td>1,416 (24)</td>
</tr>
<tr>
<td>250</td>
<td>397 (78)</td>
<td>49.4</td>
<td>1,405 (115)</td>
</tr>
<tr>
<td>500</td>
<td>814 (123)</td>
<td>24.1</td>
<td>1,655 (34)</td>
</tr>
<tr>
<td>LIBSVM</td>
<td>4,334 NA</td>
<td>NA</td>
<td>28,149 NA</td>
</tr>
</tbody>
</table>
Overheads
Key Technical Challenges

• Acquire labeled data (most data are unlabeled)
• Formulate distance function
• Train a classifier
• Classify unlabeled data
  – Fast
  – Low power consumption
Context-Aware Computing

[Chang, et al. VLDB 2013, 2014]
Transportation-mode Detection

- what
- where
- when
Transportation Mode Detection
[Chang, et al., VLDB 2013, 2014]
Activity

7356 steps taken today
74% of goal of 10,000

13 floors climbed today
130% of goal of 10

3.42 miles traveled today
68% of goal of 5.00

2718 calories burned
124% of goal of 2,184

936 active score
94% of goal of 1,000

Top Daily Step Badge
5,000 steps

Top Daily Climb Badge
10 floors

Want to challenge yourself to be more active? Start a free week trial of the Fitbit trainer now!
Data Driven Classification

- Bag
- Chest
- Waist Pack
- Hand
- Pocket

<table>
<thead>
<tr>
<th>Activity</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Still</td>
<td>92%</td>
</tr>
<tr>
<td>Walk</td>
<td>89%</td>
</tr>
<tr>
<td>Run</td>
<td>97%</td>
</tr>
<tr>
<td>Bike</td>
<td>82%</td>
</tr>
<tr>
<td>Others</td>
<td>88%</td>
</tr>
</tbody>
</table>
Sensor Hub Saving

Accuracy

Estimated Hourly Power Consumption (mA)

- SVM (linear)
- SVM (d-2 polynomial)
- SVM (d-3 polynomial)
- SVM (d-4 polynomial)
- SVM (d-5 polynomial)
- SVM (RBF)

0.03 mA 0.15 mA 0.5 mA 1.38 mA 3.3 mA 91.53 mA

86.36 88.46 90.66 90.72 90.73 91.53

80 82 84 86 88 90 92

0 200 400 600 800 1000 1200

HTC Proprietary and Confidential

Ed Chang @ BigDat 2015
SVMs $\rightarrow$ Max Margin $M$

- Min $|w|^2/2$
  - subject to $y_i(x_iw+b) \geq 1$
  - $i = 1,...,N$

- $L_p = \min_{w,b} |w|^2/2 + \sum_{i=1..N} \alpha_i [y_i(x_iw+b)-1]$

- $w = \sum_{i=1..N} \alpha_i y_i x_i$

- $0 = \sum_{i=1..N} \alpha_i y_i$
Wolfe Dual

- \[ \text{Ld} = \sum_{i=1..N} \alpha - \frac{1}{2} \sum \sum_{i,j=1..N} \alpha_i \alpha_j y_i y_j x_i x_j \]
- Subject to
  - \[ \alpha_i \geq 0 \]
  - \[ \alpha_i [y_i(x_i w + b) - 1] = 0 \]
  - KKT conditions
    - \[ \alpha_i > 0, y_i(x_i w + b) = 1 \] (Support Vectors)
    - \[ \alpha_i = 0, y_i(x_i w + b) > 1 \]
Class Prediction

• $y_q = w \cdot x_q + b$

• $w = \sum_{i=1..N} \alpha_i y_i x_i$

• $y_q = \text{sign}(\sum_{i=1..N} \alpha_i y_i (x_i \cdot x_q) + b)$
Sensor Hub Saving 2/2

- Power Consumption by MCU/CPU
- Classifier: SVM (degree-3 polynomial)

88.5 mA

177X power reduction

0.5 mA

Phone

Sensor hub
Applications & Algorithms

• Applications
  – HTC XPRICE Tricorder
  – Context-aware Computing

• Key Algorithms
  – Frequent Itemset Mining [ACM RS 08]
  – Latent Dirichlet Allocation [WWW 09, TIST 10]
  – Support Vector Machines [MM 01, MS 03, NIPS 07, VLDB 14]
    Spectral Clustering [ECML 08, PAMI 10]
  – Deep Learning [NIPS 12, OSDI 14]

• Perspectives and Opportunities
Clustering
Most Widely Used Pattern Recognition Subroutine

• Microarray Data Analysis
• Ultrasound Image Segmentation
• Document Pattern Discovery
• High-dimensional Data Indexing
K Means
Spectral Clustering [A. Ng, M. Jordan]

• Exploit *pairwise similarity* of data instances

• Key steps
  – Construct pairwise similarity matrix
    • *e.g.*, using Geodisc distance
  – Compute the Laplacian matrix
  – Apply eigendecomposition
  – Perform $k$-means
Scalability Problem

- Quadratic computation of $n \times n$ matrix
- Approximation methods

### Methods
- Dense Matrix
  - Sparsification
    - $t$-NN
    - $\xi$-neighborhood
    - ... random
  - Nystrom
    - greedy
  - Others
  - ...
Sparsification vs. Sampling

- Construct the dense similarity matrix $S$
- Sparsify $S$
- Compute Laplacian matrix $L$
  \[ L = I - D^{-1/2}SD^{-1/2}, \quad D_{ii} = \sum_{j=1}^{n} S_{ij} \]
- Apply $ARPACLK$ on $L$
- Use $k$-means to cluster rows of $V$ into $k$ groups

- Randomly sample $l$ points, where $l \ll n$
- Construct dense similarity matrix $[A \ B]$ between $l$ and $n$ points
- Normalize $A$ and $B$ to be in Laplacian form
  \[ R = A + A^{-1/2}BB^TA^{-1/2}; \]
  \[ R = U\Sigma U^T \]
- $k$-means
Empirical Study [song, et al., ecml 08]

• Dataset: RCV1 (Reuters Corpus Volume I)
  – A filtered collection of 193,944 documents in 103 categories

• Photo set: PicasaWeb
  – 637,137 photos

• Experiments
  – Clustering quality vs. computational time
    • Measure the similarity between CAT and CLS
    • Normalized Mutual Information (NMI)
      \[
      NMI(CAT;CLS) = \frac{I(CAT;CLS)}{\sqrt{H(CAT)H(CLS)}}
      \]
    – Scalability
NMI Comparison (on RCV1)

Nystrom method

Sparse matrix approximation

1/26/2015

Ed Chang @ BigDat 2015
Speedup Test on 637,137 Photos

- $K = 1000$ clusters

<table>
<thead>
<tr>
<th>Machines</th>
<th>Eigenvalue Solver Time (sec.)</th>
<th>Speedup</th>
<th>$k$-means Time (sec.)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$8.074 \times 10^4$</td>
<td>2.00</td>
<td>$3.609 \times 10^4$</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>$4.427 \times 10^4$</td>
<td>3.65</td>
<td>$1.806 \times 10^4$</td>
<td>4.00</td>
</tr>
<tr>
<td>4</td>
<td>$2.184 \times 10^4$</td>
<td>7.39</td>
<td>$8.469 \times 10^3$</td>
<td>8.52</td>
</tr>
<tr>
<td>8</td>
<td>$9.867 \times 10^3$</td>
<td>16.37</td>
<td>$4.620 \times 10^3$</td>
<td>15.62</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>$4.886 \times 10^3$</td>
<td>33.05</td>
<td>$2.021 \times 10^3$</td>
<td>35.72</td>
</tr>
<tr>
<td>64</td>
<td>$4.067 \times 10^8$</td>
<td>39.71</td>
<td>$1.433 \times 10^3$</td>
<td>50.37</td>
</tr>
<tr>
<td>128</td>
<td>$3.471 \times 10^3$</td>
<td>46.52</td>
<td>$1.090 \times 10^3$</td>
<td>66.22</td>
</tr>
<tr>
<td>256</td>
<td>$4.021 \times 10^3$</td>
<td>40.16</td>
<td>$1.077 \times 10^3$</td>
<td>67.02</td>
</tr>
</tbody>
</table>

- Achiever linear speedup when using 32 machines, after that, sub-linear speedup because of increasing communication and sync time
## Sparsification vs. Sampling

<table>
<thead>
<tr>
<th></th>
<th>Sparsification</th>
<th>Nystrom, random sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>Full n x n similarity scores</td>
<td>None</td>
</tr>
<tr>
<td>Pre-processing</td>
<td>O($n^2$) worst case; easily parallizable</td>
<td>O(nl), l &lt;&lt; n</td>
</tr>
<tr>
<td>Complexity (bottleneck)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effectiveness</td>
<td>Good</td>
<td>Not bad (Jitendra M., PAMI)</td>
</tr>
</tbody>
</table>
Applications & Algorithms

• Applications
  – HTC XPRICE Tricorder
  – Context-aware Computing

• Key Algorithms
  – Frequent Itemset Mining [ACM RS 08]
  – Latent Dirichlet Allocation [WWW 09, TIST 10]
  – Support Vector Machines [MM 01, MS 03, NIPS 07, VLDB 14]
  – Spectral Clustering [ECML 08, PAMI 10]

  Deep Learning [NIPS 12, OSDI 14]

• Perspectives and Opportunities
Multiple-Layer Networks

Neuron Network (NN) Model

An elementary neuron with R inputs is shown below. Each input is weighted with an appropriate w. The sum of the weighted inputs and the bias forms the input to the transfer function f. Neurons can use any differentiable transfer function f to generate their output.

\[ a = f(Wp + b) \]

Where

\[ R = \text{number of elements in input vector} \]
NN Model
Transfer Functions (Activation Function)

Multilayer networks often use the log-sigmoid transfer function \( \text{logsig} \). The function \( \text{logsig} \) generates outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity.
NN Model
Feedforward Network

A single-layer network of S logsig neurons having R inputs is shown below in full detail on the left and with a layer diagram on the right.

where...

\[ a = \text{logsig}(Wp + b) \]

\[ R = \text{number of elements in input vector} \]

\[ S = \text{number of neurons in layer} \]
NN Model
Learning Algorithm

The following slides describe the learning process of a multi-layer neural network employing the backpropagation algorithm. To illustrate this process, a three-layer neural network with two inputs and one output, which is shown in the picture below, is used:

![Neural Network Diagram]
Learning Algorithm: Backpropagation

Each neuron is composed of two units. First unit adds products of weights coefficients and input signals. The second unit realizes a nonlinear function, called neuron transfer (activation) function. Signal $e$ is adder output signal, and $y = f(e)$ is output signal of nonlinear element. Signal $y$ is also output signal of...
Feed Forward

Pictures below illustrate how signal is forward-feeding through the network, Symbols $w_{(xm)n}$ represent weights of connections between network input $x_m$ and neuron $n$ in input layer. Symbols $y_n$ represents output signal of neuron $n$.

\[ y_1 = f_1(w_{(x1)1}x_1 + w_{(x2)1}x_2) \]
Feed Forward

\[
y_2 = f_2(w_{(x1)}x_1 + w_{(x2)}x_2)
\]
Feed Forward

$$y_3 = f_3(w_{(x1)3}x_1 + w_{(x2)3}x_2)$$
Feed Forward

Propagation of signals through the hidden layer. Symbols $w_{mn}$ represent weights of connections between output of neuron $m$ and input of neuron $n$ in the next layer.

\[ y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3) \]
Feed Forward

\[ y_5 = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3) \]
Learning Algorithm: Forward Pass

Propagation of signals through the output layer.

\[ y = f_6(w_{46}y_4 + w_{56}y_5) \]
Learning Algorithm: Backpropagation

To teach the neural network we need training data set. The training data set consists of input signals \((x_1 \text{ and } x_2)\) assigned with corresponding target (desired output) \(z\).

The network training is an iterative process. In each iteration weights coefficients of nodes are modified using new data from training data set. Modification is calculated using algorithm described below:

Each teaching step starts with forcing both input signals from training set. After this stage we can determine output signals values for each neuron in each network layer.
Learning Algorithm: Backpropagation

In the next algorithm step the output signal of the network $y$ is compared with the desired output value (the target $z$), which is found in training data set. The difference is called error signal $\delta$ of output layer neuron

$$\delta = z - y$$
Learning Algorithm: Backpropagation

The idea is to propagate error signal $\delta$ (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.
Learning Algorithm: Backpropagation

The idea is to propagate error signal $\delta$ (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.
Learning Algorithm: Backpropagation

The weights' coefficients $w_{mn}$ used to propagate errors back are equal to this used during computing output value. Only the direction of data flow is changed (signals are propagated from output to inputs one after the other). This technique is used for all network layers. If propagated errors came from few neurons they are added. The illustration is below:
Learning Algorithm: Backpropagation

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below $df(e)/de$ represents derivative of neuron activation function (which weights are modified).

\[
w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1
\]

\[
w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2
\]
Learning Algorithm: Backpropagation

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below $df(e)/de$ represents derivative of neuron activation function (which weights are modified).

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$
Learning Algorithm: Backpropagation

When the error signal for each neuron is computed, the weights coefficients of each neuron input node may be modified. In formulas below $df(e)/de$ represents derivative of neuron activation function (which weights are modified).

$$w'_{46} = w_{46} + \eta \delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta \delta \frac{df_6(e)}{de} y_5$$
Sigmoid function $f(e)$ and its derivative $f'(e)$

$$f(e) = \frac{1}{1 + e^{-\beta e}}, \quad \beta \text{ is the parameter for slope}$$

Hence

$$f'(e) = \frac{df(e)}{de} = \frac{d\left(\frac{1}{1 + e^{-\beta e}}\right)}{d(1 + e^{-\beta e})} \frac{df(e^{-\beta e})}{de}$$

$$f'(e) = \frac{-\beta}{(1 + e^{-\beta e})^2} e^{-\beta e} = \frac{-\beta}{(1 + e^{-\beta e})^2} e^{-e}$$

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

For simplicity, parameter for the slope $\beta = 1$

$$f'(e) = f(e)(1 - f(e))$$
Model Parallelism
[j. dean et al, nips 2012]
Scalable Deep Learning Platform

Microsoft Project ADAM

• Scalable training algorithm
  – Asynchronous SDG (stochastic gradient descent)

• Scalable model partitioning
  – Model parallelism

• Scalable model parameter store
  – Data parallelism

• Scalable data transformations
  – Data preprocessing and augmentation
Scalability of Backpropagation
[Project Adam, OSDI 2014]

• Based on the Multi-Spert system and exploits both model and data parallelism

\[ w' = w - \eta \Delta w \]

Model Training Optimizations (1/3)

- Multi-threaded training
  - Multiple threads are sharing the same model weights
  - NUMA-aware allocations to reduce cross-memory bus traffic

- Fast weight updates
  - Update the sharded model weights locally **WITHOUT** using locks
    - Weight updates are commutative and associative
    - Neural networks are resilient to the noise introduced
NUMA

• Non Uniform Memory Access
Model Training Optimizations (2/3)

• Reducing memory copies
  – Do not copy the parameters, pass a pointer instead

• Memory system optimizations
  – Fit the working sets in the L3 cache (e.g., 8M)

• Mitigating the impact of slow machines
  –Threads to process multiple images in parallel
  – Training epoch terminates when 75% of the model replicas are done ➔ 20% speed up
Model Training Optimizations (3/3)

• Reduce the communication to the parameter server
  – Can also offload some computation work to the parameter server
Concluding Remarks

• More data is helpful, and hence big data
• Computational time is reduced by using virtually infinitely amount of resources
• Once computation is fully parallelized, IO cost can be reduced via hardware solutions
• Both algorithmic approach and system approach are required to achieve good speedup
Key References

- **[MS 03]** Discovery of a Perceptual Distance Function for Measuring Image Similarity, B Li, E. Y. Chang, and Y Wu, Journal of Multimedia Systems, 2003
Foundations of Large-Scale Multimedia Information Management and Retrieval Mathematics of Perception covers knowledge representation and semantic analysis of multimedia data and scalability in signal extraction, data mining, and indexing. The book is divided into two parts: Part I - Knowledge Representation and Semantic Analysis focuses on the key components of mathematics of perception as it applies to data management and retrieval. These include feature selection/reduction, knowledge representation, semantic analysis, distance function formulation for measuring similarity, and multimodal fusion. Part II - Scalability Issues presents indexing and distributed methods for scaling up these components for high-dimensional data and Web-scale datasets. The book presents some real-world applications and remarks on future research and development directions.

The book is designed for researchers, graduate students, and practitioners in the fields of Computer Vision, Machine Learning, Large-scale Data Mining, Database, and Multimedia Information Retrieval.

Dr. Edward Y. Chang was a professor at the Department of Electrical & Computer Engineering, University of California at Santa Barbara, before he joined Google as a research director in 2006. Dr. Edward Y. Chang received his M.S. degree in Computer Science and Ph.D degree in Electrical Engineering, both from Stanford University.
Additional References

References (cont.)

References (cont.)