Big Data Analytics
Architectures, Algorithms and Applications
Part #1: Scalable Big Data Algorithms

Edward Chang 張智威
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

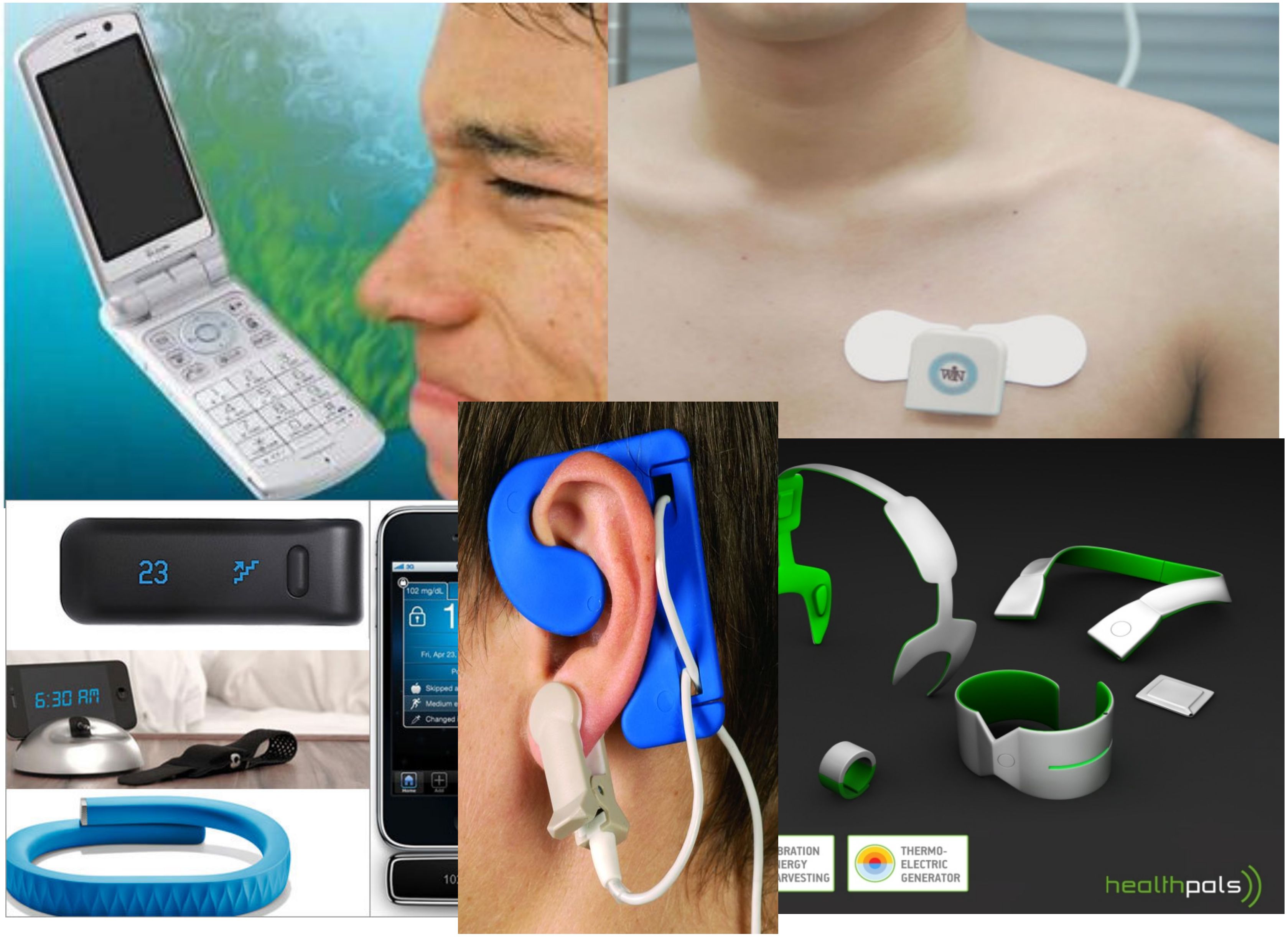
- [ACM RS 08] PFP: Parallel FP-Growth for Query Recommendation, H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang, ACM Recommendation Systems, Lausanne, October 2008
- [TIST 10] PLDA+: Parallel Latent Dirichlet Allocation with Data Placement and Pipeline Processing, ACM Transactions on Intelligent Systems and Technology, 2011.
- [MM 01] Support Vector Machine Active Learning for Image Retrieval, S. Tong and E. Chang, ACM MM, 2001
- [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
- [NIPS 07] Parallel Support Vector Machines, E. Y. Chang, et al., NIPS 2007.
- [PAMI 10] Parallel Spectral Clustering, W.-Y. Chen, Y. Song, H. Bai, Chih-Jen Lin, and E. Y. Chang, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2010.
- [NIPS 12] Large Scale Distributed Deep Networks. J. Dean, G.S. Corrado, R. Monga, K. Chen, M. Devin, Q.V. Le, M.Z. Mao, M.A. Ranzato, A. Senior, P. Tucker, K. Yang, A. Y. Ng, NIPS 2012.
- [OSDI 14] Project Adam: Building an Efficient and Scalable Deep Learning Training System Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman, OSDI 2014.
- [VLDB 14] Big Data, Small Footprint: The Design of a Low-Power Classifier for Detecting Transportation Modes (with Open Source dataset), M. Yu, T. Yu, C.-J. Lin, and E. Y. Chang, VLDB, August 2014.

Open Source Links

Downloaded > 12,000 times

- PSVM,
- PLDA+,
- Parallel Spectral Clustering, and
- Parallel Frequent Pattern Mining

Health Sensors



Machine Learning

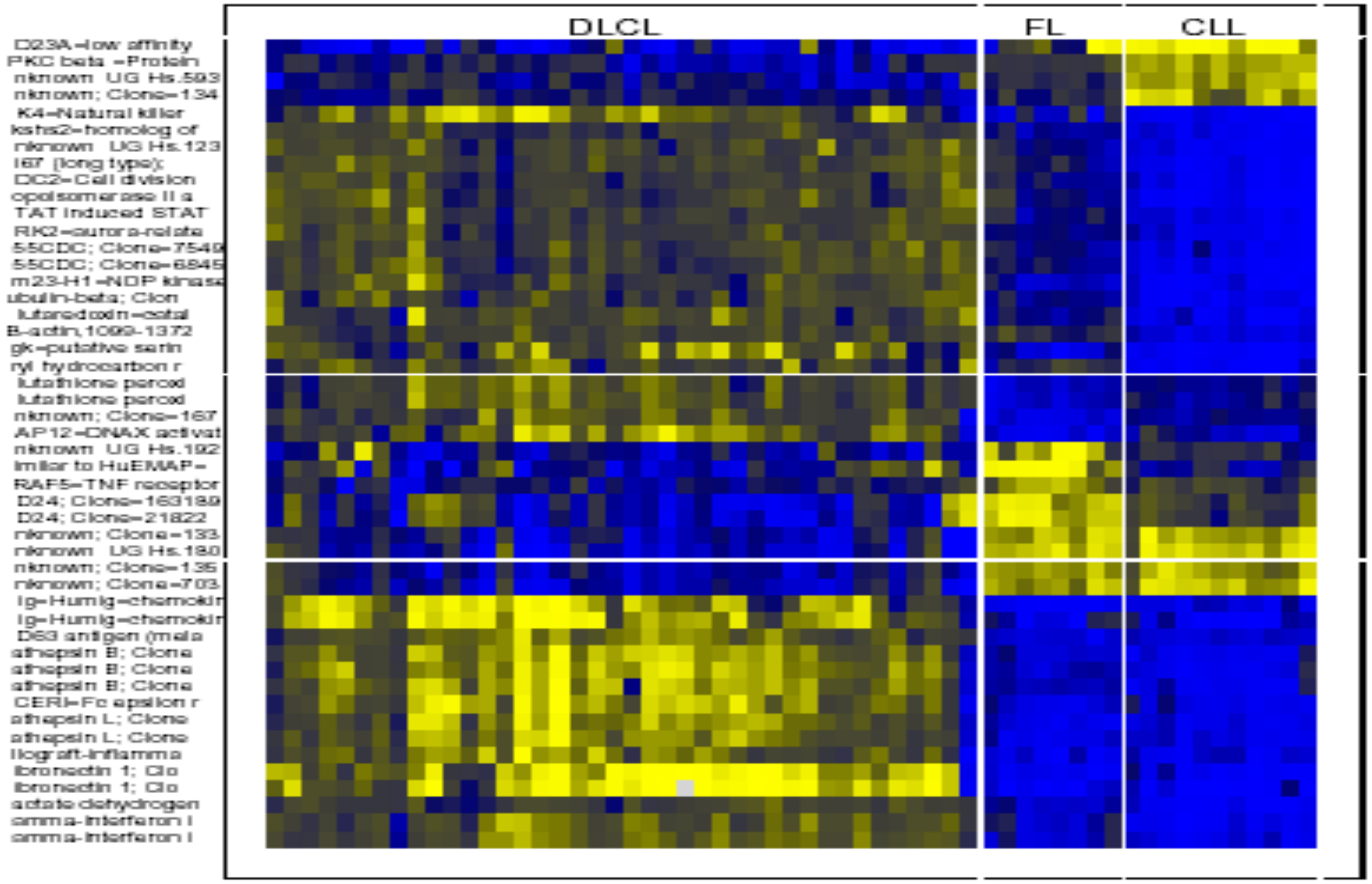
- X: Data
  - U: Unlabeled data
  - L: Labeled data
- Φ: Learning algorithm
  - Implied hypothesis
- $f = \Phi(L + U)$
  - Minimize some error function
  - Regularize parameters to prevent over-fitting
- $\hat{y} = f(\mathbf{u} \in U)$

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

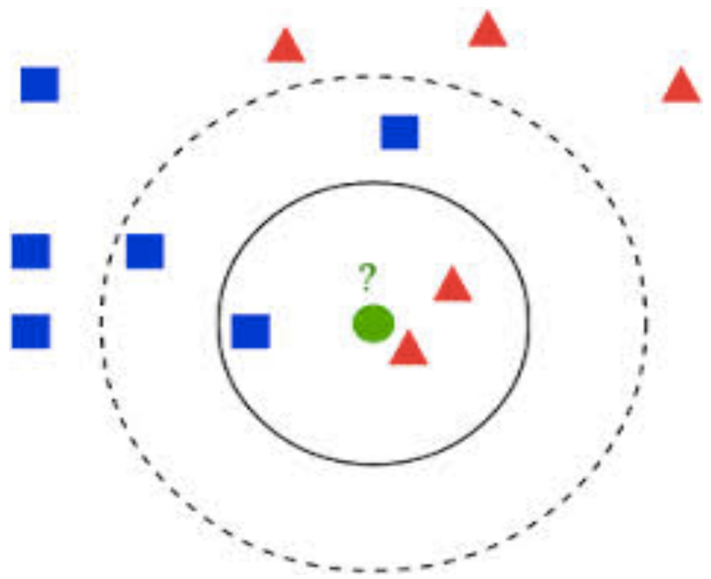
- Volume, both too large and too small
  - Amount of data ↑
  - Amount of labeled data ↓
  - dimensionality of data↑
- Variety
- Velocity
  - Addressed in Lecture #3 with online learning

Why Big Data

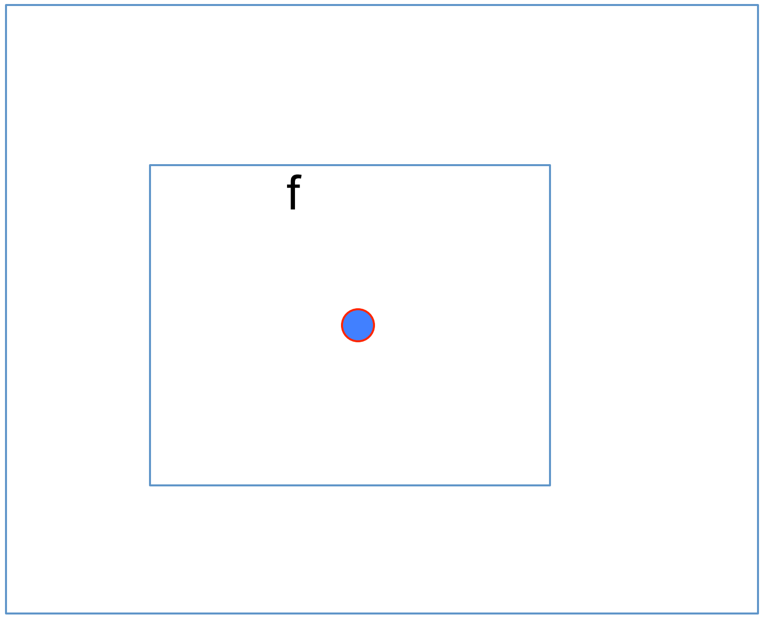
- Simply too many data instances? Yes
- But also growing complexity, or dimensionality of data

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



- $f = .5$, $d = 2$, NN = 25%
- When d is large

  The volume of NN → 0

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

- Curse of dimensionality

More Data vs. Better Algorithms

Banko & Brill, 2001

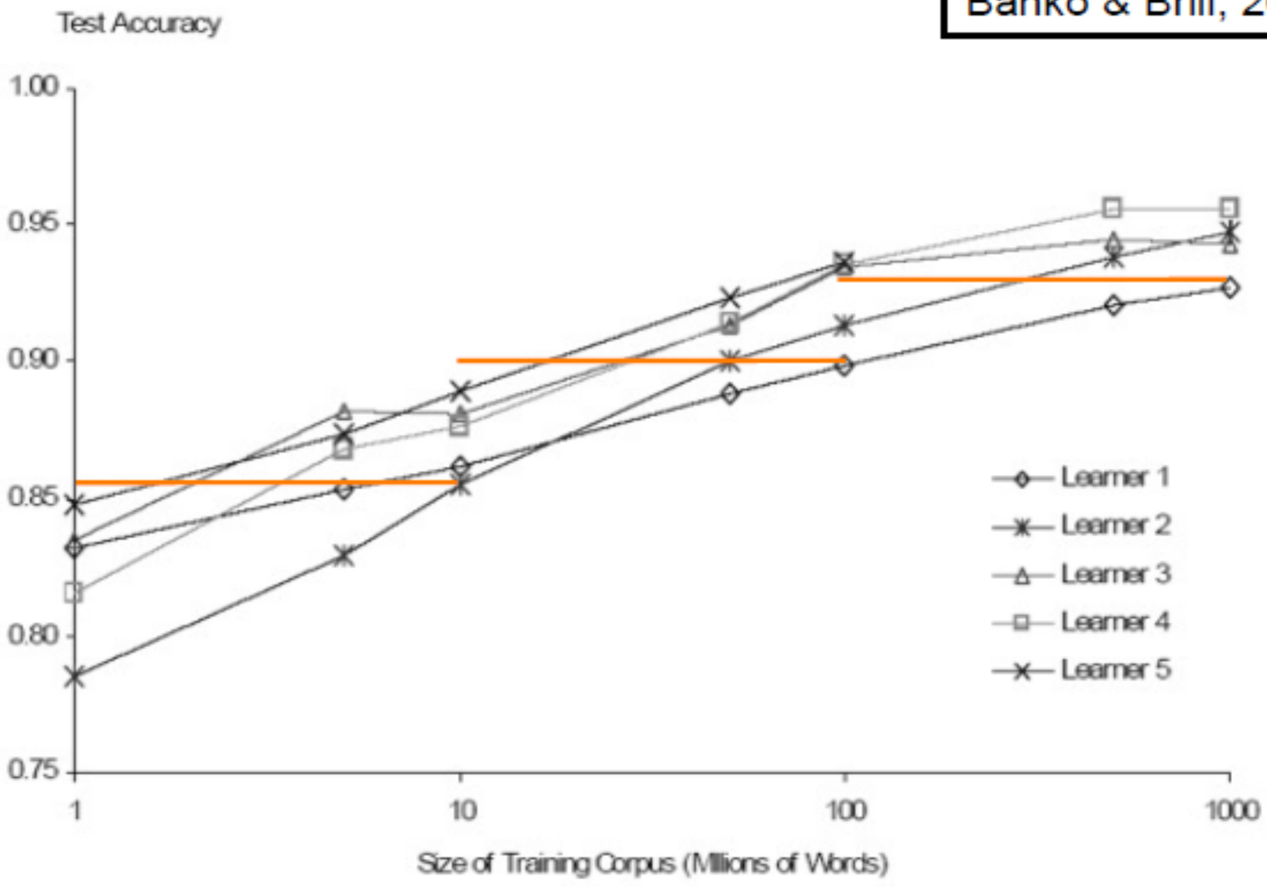


Figure 2. Learning Curves for Confusable Disambiguation

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

XPRIZE Tricorder

Fostering disruptive innovation to bring affordable health care to underprivileged

QUALCOMM TRICORDER XPRIZE

Dynamical Biomarkers Group

OFFICIAL TEAM

Portable device weight < 5 pounds

Exam 15+ diseases & monitor 5 vital signs

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

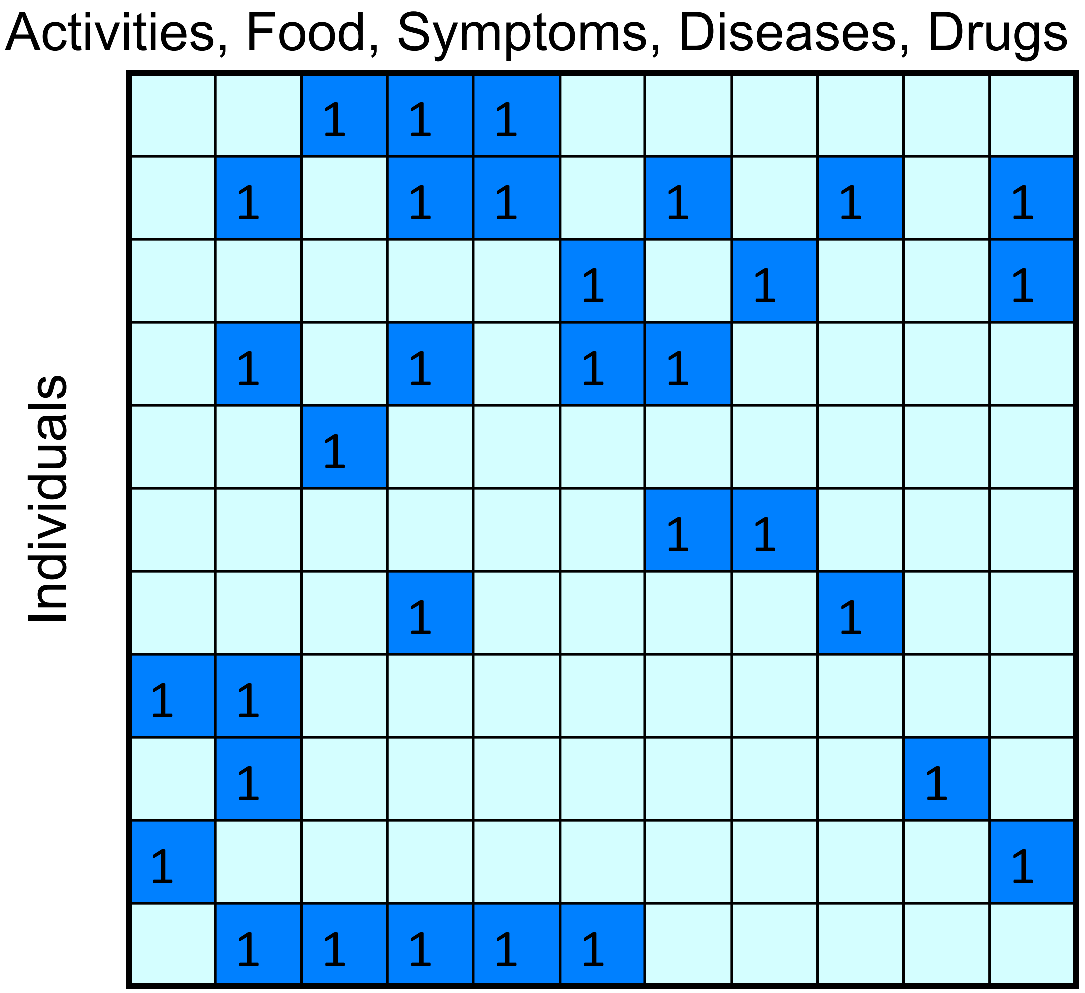
Final round : May, 2015

Diagnosis: Collaborative Filtering

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

↓

Predict further *membership*


Activities, Food, Symptoms, Diseases, Drugs

Individuals

		1	1	1						
	1		1	1		1		1		1
					1		1			1
	1		1		1	1				
		1								
						1	1			
			1					1		
1	1									
	1								1	
1										1
	1	1	1	1	1					

Collaborative Filtering

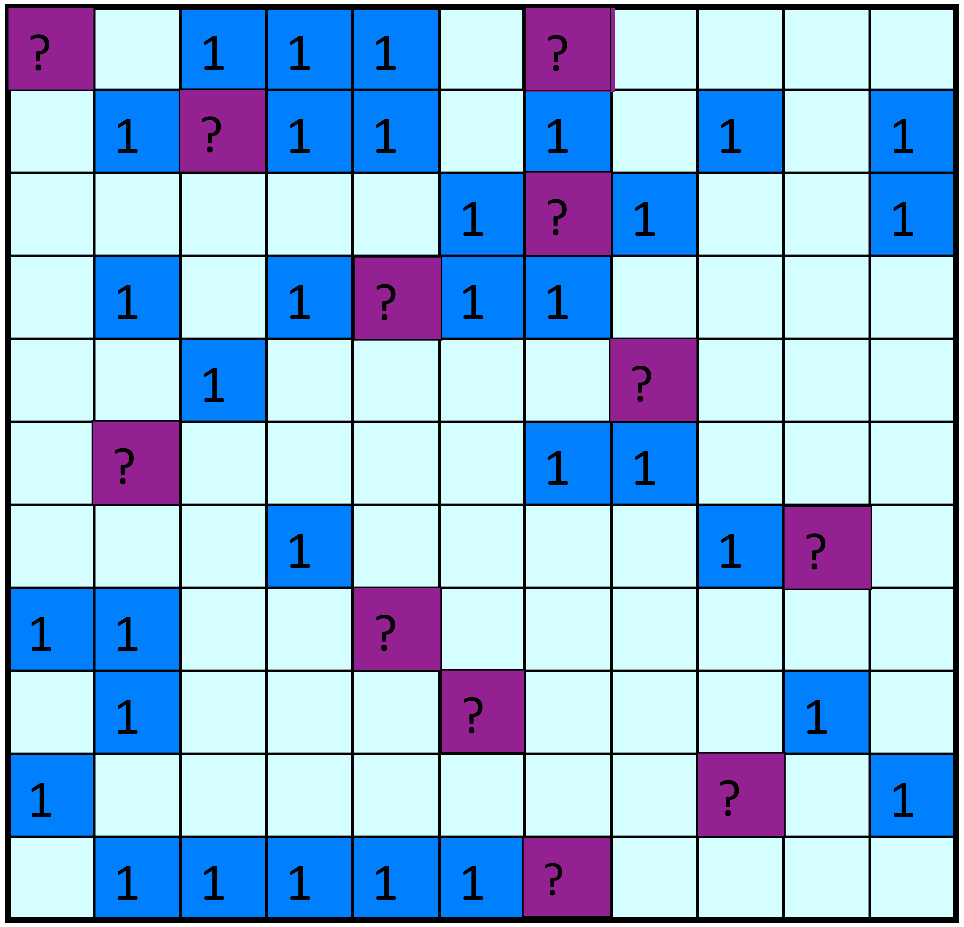
Based on *partially* observed matrix

↓

Predict *unobserved* entries

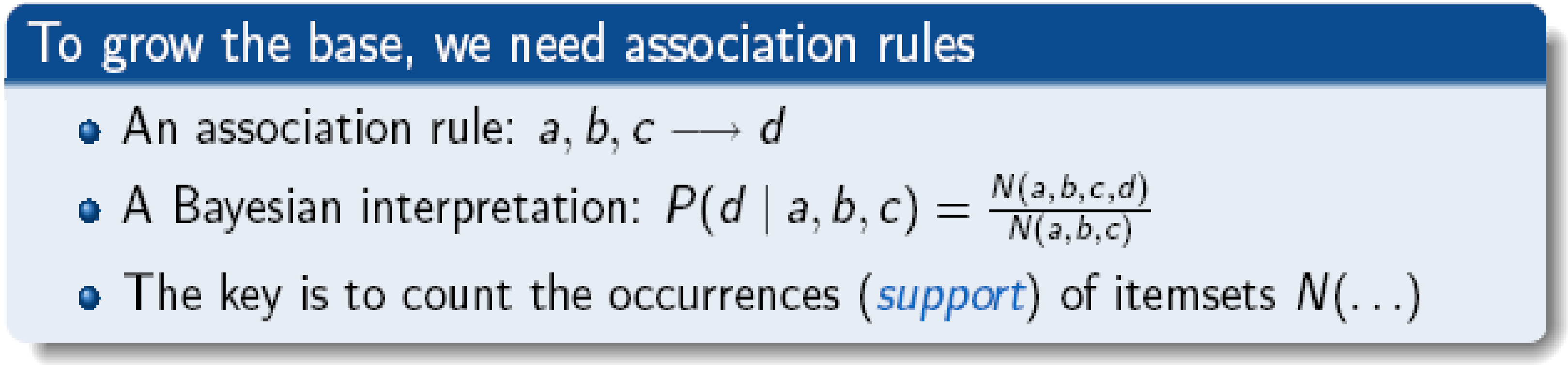
Activities, Food, Symptoms, Diseases, Drugs

Individuals



?		1	1	1		?				
	1	?	1	1		1		1		1
					1	?	1			1
	1		1	?	1	1				
		1					?			
	?					1	1			
			1					1	?	
1	1			?						
	1				?				1	
1							?			1
	1	1	1	1	1	?				

FIM-based Prediction



To grow the base, we need association rules

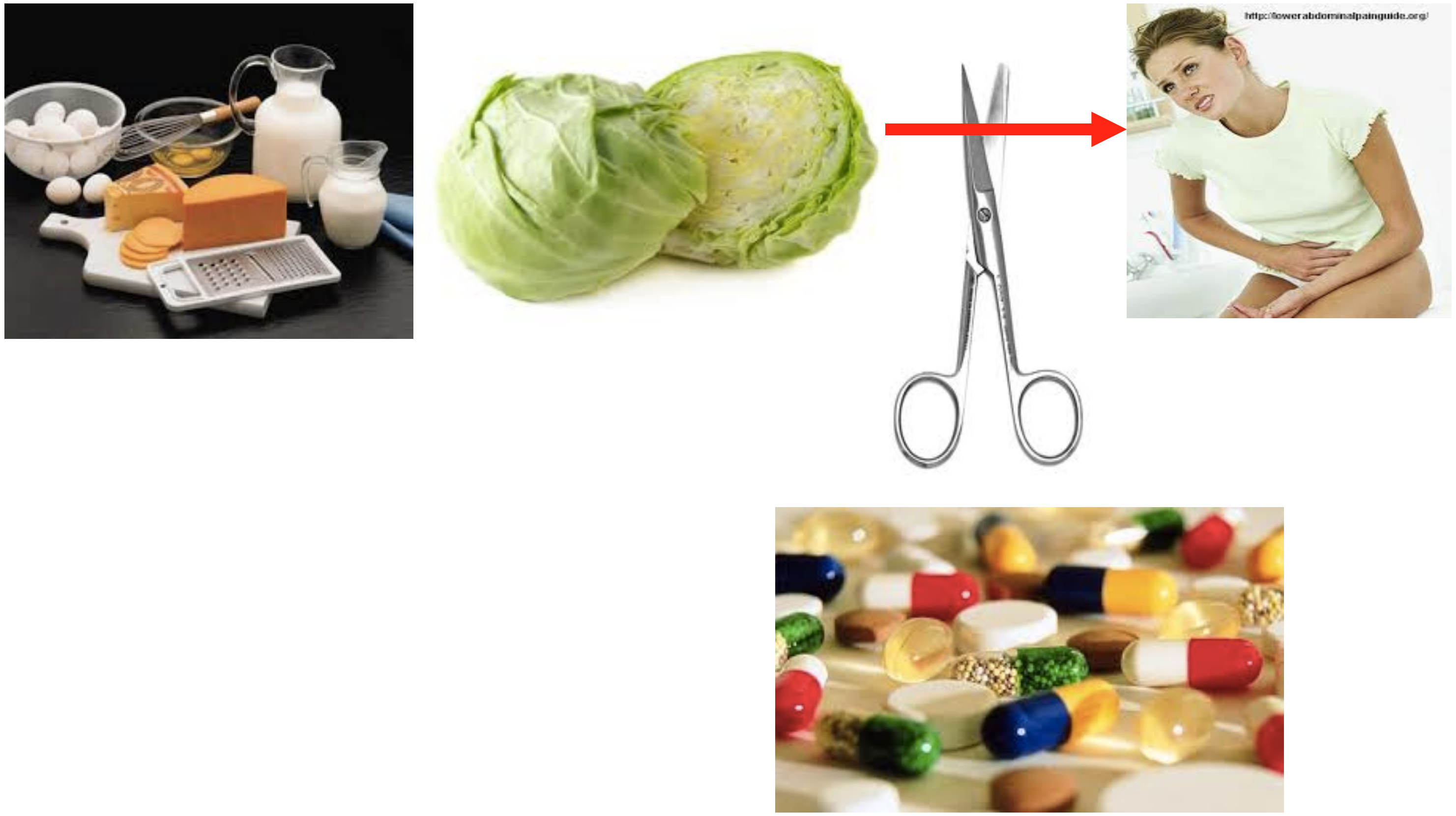
- An association rule: $a, b, c \longrightarrow d$
- A Bayesian interpretation: $P(d \mid a, b, c) = \frac{N(a,b,c,d)}{N(a,b,c)}$
- The key is to count the occurrences (*support*) of itemsets $N(\ldots)$

FIM-based Prediction

To grow the base, we need association rules

- An association rule: $a, b, c \longrightarrow d$
- A Bayesian interpretation: $P(d \mid a, b, c) = \frac{N(a,b,c,d)}{N(a,b,c)}$
- 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

f a c d g i m p	f c a m p
a b c f l m o	f c a b m
b f h j o	f b
b c k s p	c b p
a f c e l p m n	f c a m p

f: 4
c: 4
a: 3
b: 3
m: 3
p: 3

---

o: 2
d: 1
e: 1
g: 1
h: 1
i: 1
k: 1
l : 1
n: 1

- 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

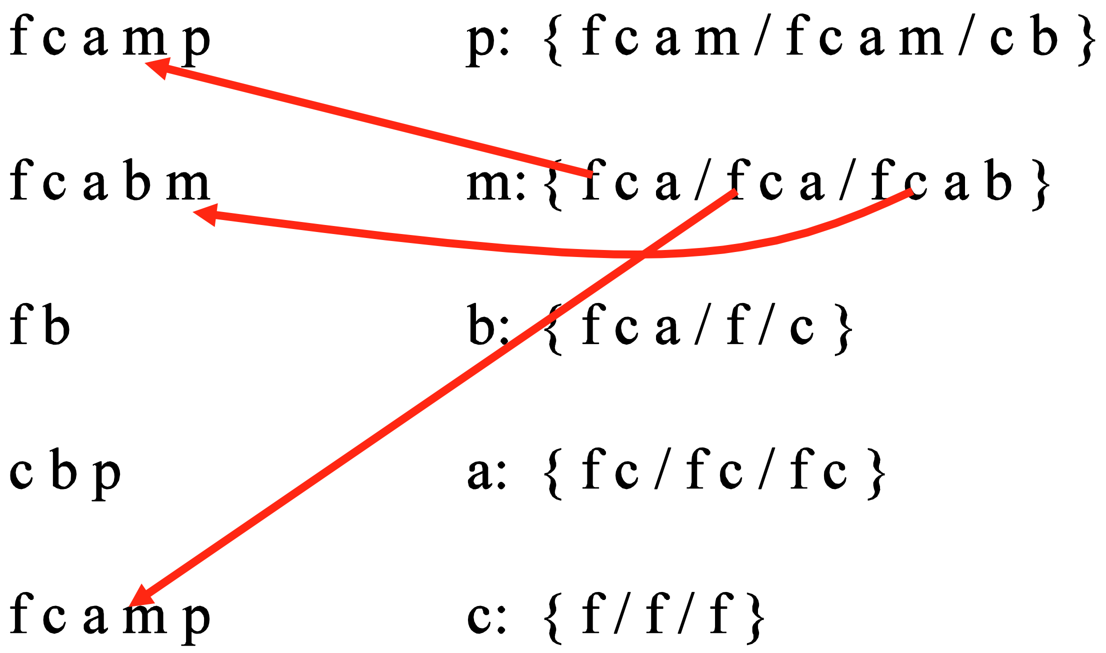
- According to Observation 2, we construct CDB of item A; then from this CDB, we find those patterns containing *A*
- How to construct the CDB of *A*?
  - If a transaction contains *A*, this transaction should appear in the CDB of *A*
  - Given a transaction {*B, A, C*}, it should appear in the CDB of *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 → <*A,B,C*>
  - Put <> into the CDB of *A*
  - Put <*A*> into the CDB of *B*
  - Put <*A,B*> into the CDB of *C*

Example of Projection

f c a m p	p: { f c a m / f c a m / c b }
f c a b m	m: { f c a / f c a / f c a b }
f b	b: { f c a / f / c }
c b p	a: { f c / f c / f c }
f c a m p	c: { f / f / f }

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



f c a m p	p: { f c a m / f c a m / c b }
f c a b m	m: { f c a / f c a / f c a b }
f b	b: { f c a / f / c }
c b p	a: { f c / f c / f c }
f c a m p	c: { f / f / f }

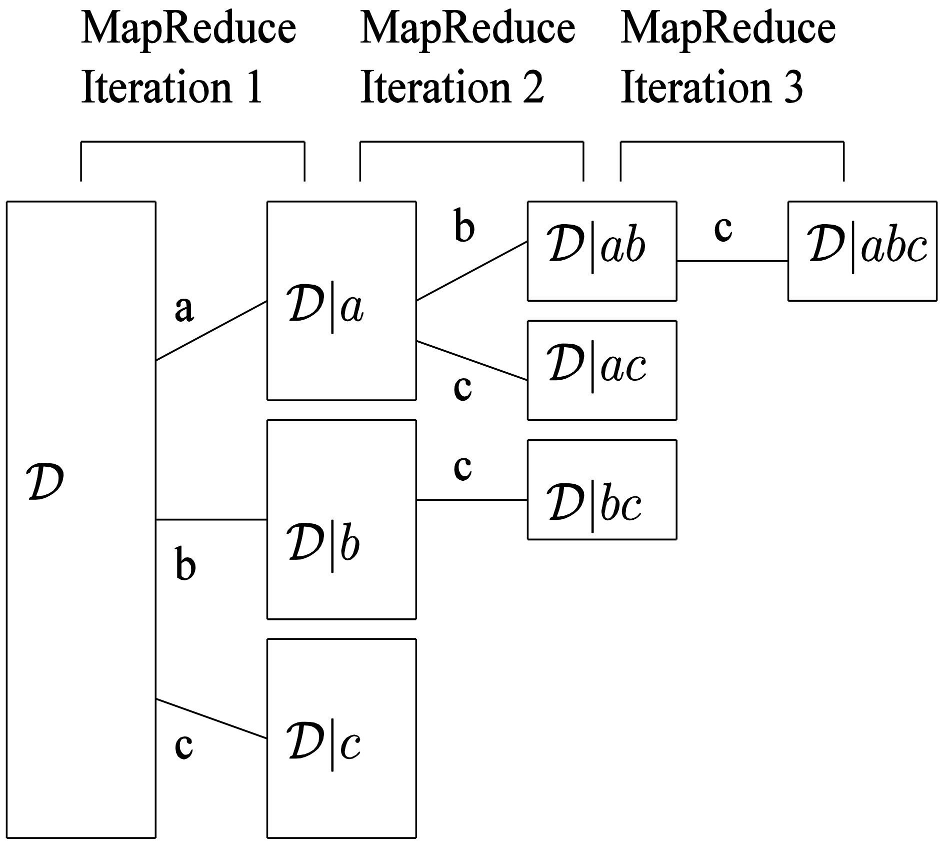
Example of Projection of a database into CDBs.
Left: sorted transactions;
Right: conditional databases of frequent items

Example of Projection

f c a m p	p: { f c a m / f c a m / c b }
f c a b m	m: { f c a / f c a / f c a b }
f b	b: { f c a / f / c }
c b p	a: { f c / f c / f c }
f c a m p	c: { f / f / f }

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 sub-tree.

Projection using MapReduce

Map inputs (transactions) key="": value	Sorted transactions (with infrequent items eliminated)	Map outputs (conditional transactions) key: value
f a c d g i m p	f c a m p	p: f c a m m: f c a a: fc c: f
a b c f l m o	f c a b m	m: f c a b b: f c a a: f c c: f
b f h j o	f b	b: f
b c k s p	c b p	p: c b
a f c e l p m n	f c a m p	b: c p: f c a m m: f c a a: f c c: f

Reduce inputs (conditional databases) key: value	Reduce outputs (patterns and supports) key: value
p:{fcam/fcam/cb}	p:3, pc:3
m: { f c a / f c a / f c a b }	m f : 3 m c : 3 m a : 3 m f c : 3 m f a : 3 m c a : 3 m f c a : 3
b: { f c a / f / c }	b : 3
a: { f c / f c / f c }	a : 3 a f : 3 a c : 3 a f c : 3
c: { f / f / f }	c : 3 c f : 3

Collaborative Filtering

[Confucius or Google QA, VLDB 2010]

Based on *membership* so far,

and *memberships* of others

$\downarrow$

Predict further *membership*

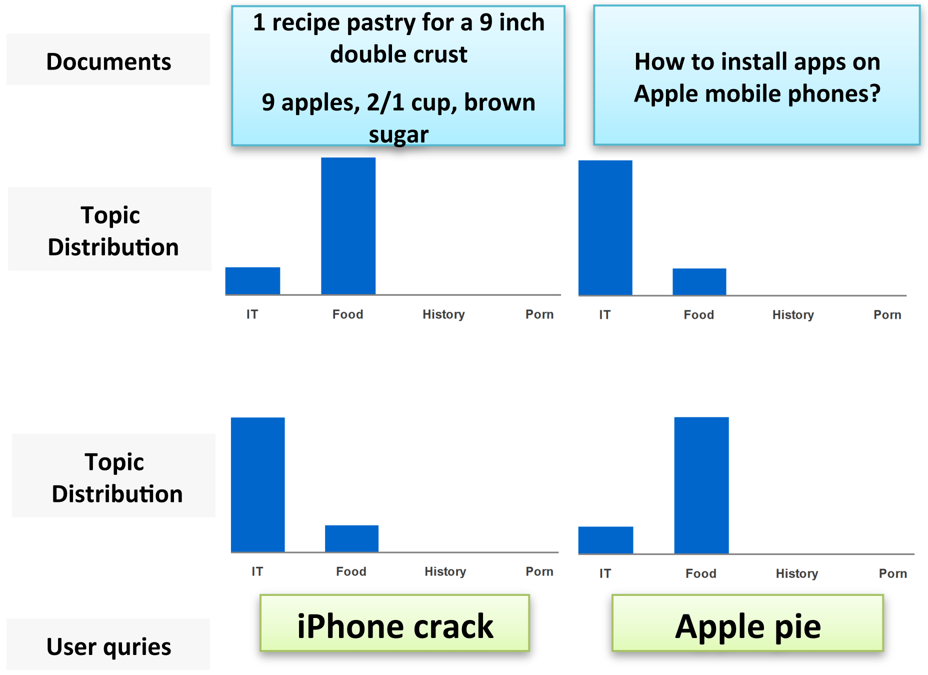
Users/Labels/Documents

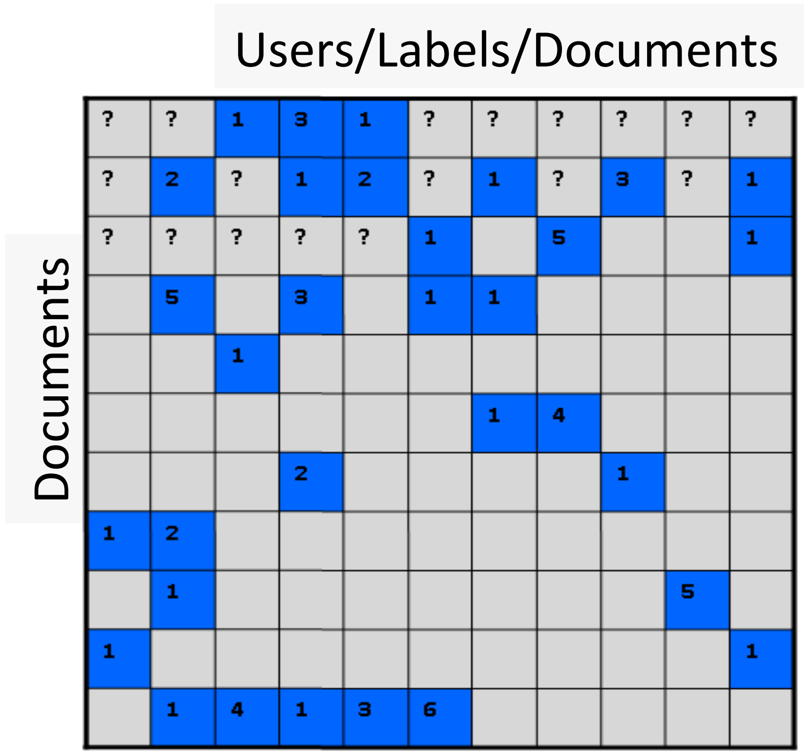
Documents

		1	1	1						
	1		1	1		1		1		1
					1		1			1
	1		1		1	1				
		1								
						1	1			
			1					1		
1	1									
	1								1	
1										1
	1	1	1	1	1					

Latent Semantic Analysis

- Search
  - Construct a latent layer for better for semantic matching
- Example:
  - iPhone crack
  - Apple pie


Documents
1 recipe pastry for a 9 inch double crust
9 apples, 2/1 cup, brown sugar
How to install apps on Apple mobile phones?
Topic Distribution
IT
Food
History
Porn
Topic Distribution
User quries
iPhone crack
Apple pie


Users/Labels/Documents
Documents

?	?	1	3	1	?	?	?	?	?	?
?	2	?	1	2	?	1	?	3	?	1
?	?	?	?	?	1		5			1
	5		3		1	1				
		1								
						1	4			
			2					1		
1	2									
	1								5	
1										1
	1	4	1	3	6					

- 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, … ,d_N}
- Vocabulary, a set of M words
  - W={w_1, … ,w_M}
- A matrix of size N * M to represent the occurrence of words in documents
  - Called the term-document matrix

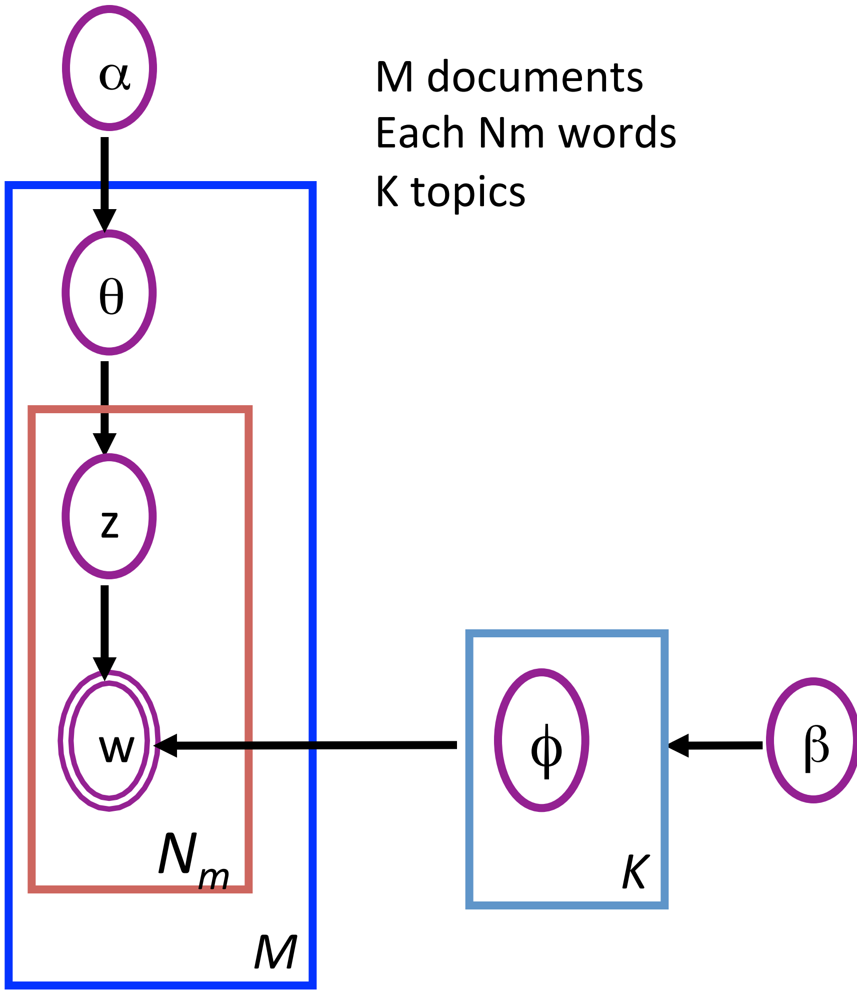
Documents, Topics, Words

- 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 ith word in a document

$$P(w_i) = \sum_{j=1}^{T} P(w_i | z_i = j) P(z_i = j)$$

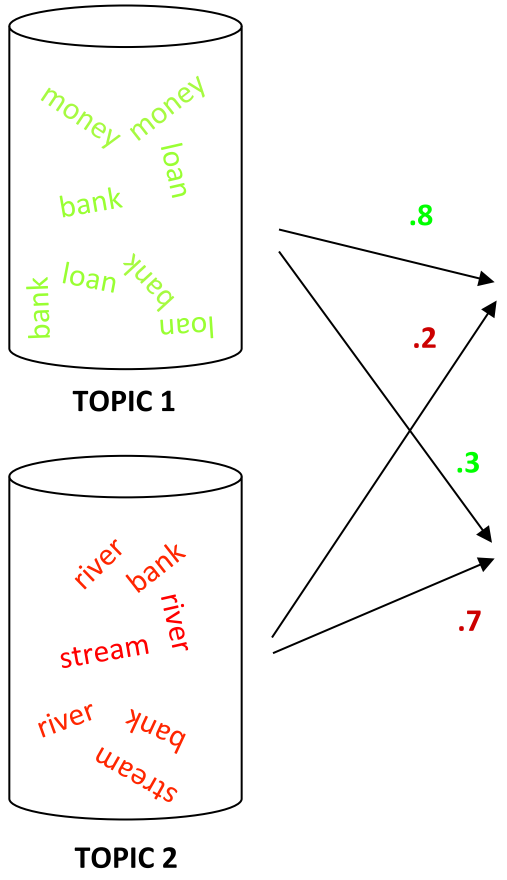
Latent Dirichlet Allocation [M. Jordan 04]

- α: uniform Dirichlet ϕ prior for per document d topic distribution (corpus level parameter)
- β: uniform Dirichlet ϕ prior for per topic z word distribution (corpus level parameter)
- θ_d is the topic distribution of doc d (document level)
- z_{dj} the topic if the jth word in d, w_{dj} the specific word (word level)



M documents
Each Nm words
K topics

Example



DOCUMENT 1: money^1 bank^1 bank^1 loan^1 river^2 stream^2 bank^1 money^1 river^2 bank^1 money^1 bank^1 loan^1 money^1 stream^2 bank^1 money^1 bank^1 bank^1 loan^1 river^2 stream^2 bank^1 money^1 river^2 bank^1 money^1 bank^1 loan^1 bank^1 money^1 stream^2

DOCUMENT 2: river^2 stream^2 bank^2 stream^2 bank^2 money^1 loan^1 river^2 stream^2 loan^1 bank^2 river^2 bank^2 bank^1 stream^2 river^2 loan^1 bank^2 stream^2 bank^2 money^1 loan^1 river^2 stream^2 bank^2 stream^2 bank^2 money^1 river^2 stream^2 loan^1 bank^2 river^2 bank^2 money^1 bank^1 stream^2 river^2 bank^2 stream^2 bank^2 money^1

Mixture topics

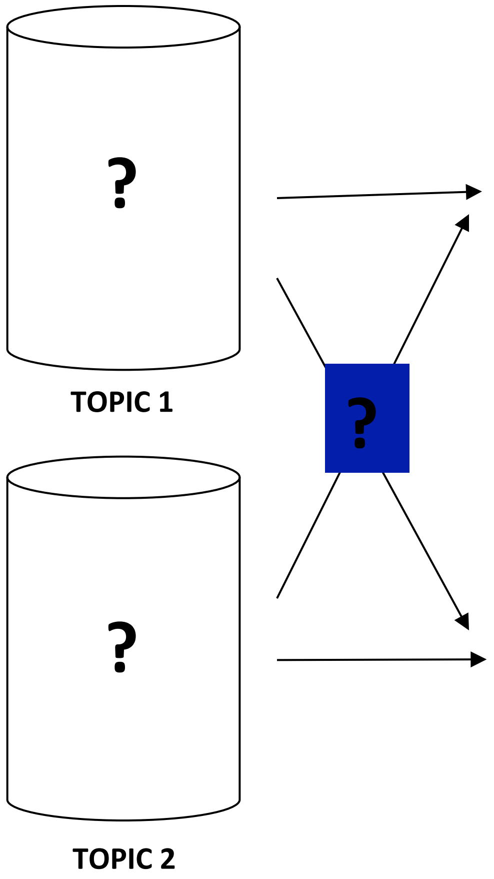
Mixture weights

Bayesian approach: use priors

Mixture weights $\sim$ Dirichlet(α)

Mixture topics $\sim$ Dirichlet(β)

Inverting ("fitting") the model



DOCUMENT 1: money$^{?}$ bank$^{?}$ bank$^{?}$ loan$^{?}$ river$^{?}$ stream$^{?}$ bank$^{?}$ money$^{?}$ river$^{?}$ bank$^{?}$ money$^{?}$ bank$^{?}$ loan$^{?}$ money$^{?}$ stream$^{?}$ bank$^{?}$ money$^{?}$ bank$^{?}$ bank$^{?}$ loan$^{?}$ river$^{?}$ stream$^{?}$ bank$^{?}$ money$^{?}$ river$^{?}$ 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$^{?}$ loan$^{?}$ river$^{?}$ stream$^{?}$ bank$^{?}$ stream$^{?}$ bank$^{?}$ money$^{?}$ river$^{?}$ stream$^{?}$ loan$^{?}$ bank$^{?}$ river$^{?}$ bank$^{?}$ money$^{?}$ bank$^{?}$ stream$^{?}$ river$^{?}$ bank$^{?}$ stream$^{?}$ bank$^{?}$ money$^{?}$

Mixture components

Mixture weights

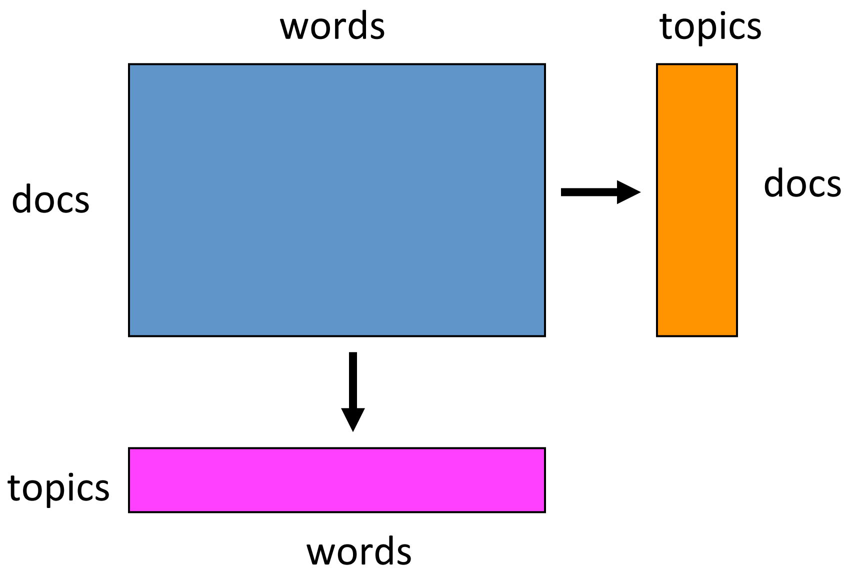
LDA Gibbs Sampling: Inputs And Outputs

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.:

PRINTING	PLAY	TEAM	JUDGE	HYPOTHESIS	STUDY
PAPER	PLAYS	GAME	TRIAL	EXPERIMENT	TEST
PRINT	STAGE	BASKETBALL	COURT	SCIENTIFIC	STUDYING
PRINTED	AUDIENCE	PLAYERS	CASE	OBSERVATIONS	HOMEWORK
TYPE	THEATER	PLAYER	JURY	SCIENTISTS	NEED
PROCESS	ACTORS	PLAY	ACCUSED	EXPERIMENTS	CLASS
INK	DRAMA	PLAYING	GUILTY	SCIENTIST	MATH
PRESS	SHAKESPEARE	SOCCER	DEFENDANT	EXPERIMENTAL	TRY
IMAGE	ACTOR	PLAYED	JUSTICE	TEST	TEACHER
PRINTER	THEATRE	BALL	EVIDENCE	METHOD	WRITE
PRINTS	PLAYWRIGHT	TEAMS	WITNESSES	HYPOTHESES	PLAN
PRINTERS	PERFORMANCE	BASKET	CRIME	TESTED	ARITHMETIC
COPY	DRAMATIC	FOOTBALL	LAWYER	EVIDENCE	ASSIGNMENT
COPIES	COSTUMES	SCORE	WITNESS	BASED	PLACE
FORM	COMEDY	COURT	ATTORNEY	OBSERVATION	STUDIED
OFFSET	TRAGEDY	GAMES	HEARING	SCIENCE	CAREFULLY
GRAPHIC	CHARACTERS	TRY	INNOCENT	FACTS	DECIDE
SURFACE	SCENES	COACH	DEFENSE	DATA	IMPORTANT
PRODUCED	OPERA	GYM	CHARGE	RESULTS	NOTEBOOK
CHARACTERS	PERFORMED	SHOT	CRIMINAL	EXPLANATION	REVIEW

Polysemy

PRINTING	**PLAY**	TEAM	JUDGE	HYPOTHESIS	STUDY
PAPER	PLAYS	GAME	TRIAL	EXPERIMENT	**TEST**
PRINT	STAGE	BASKETBALL	**COURT**	SCIENTIFIC	STUDYING
PRINTED	AUDIENCE	PLAYERS	CASE	OBSERVATIONS	HOMEWORK
TYPE	THEATER	PLAYER	JURY	SCIENTISTS	NEED
PROCESS	ACTORS	**PLAY**	ACCUSED	EXPERIMENTS	CLASS
INK	DRAMA	PLAYING	GUILTY	SCIENTIST	MATH
PRESS	SHAKESPEARE	SOCCER	DEFENDANT	EXPERIMENTAL	TRY
IMAGE	ACTOR	PLAYED	JUSTICE	**TEST**	TEACHER
PRINTER	THEATRE	BALL	**EVIDENCE**	METHOD	WRITE
PRINTS	PLAYWRIGHT	TEAMS	WITNESSES	HYPOTHESES	PLAN
PRINTERS	PERFORMANCE	BASKET	CRIME	TESTED	ARITHMETIC
COPY	DRAMATIC	FOOTBALL	LAWYER	**EVIDENCE**	ASSIGNMENT
COPIES	COSTUMES	SCORE	WITNESS	BASED	PLACE
FORM	COMEDY	**COURT**	ATTORNEY	OBSERVATION	STUDIED
OFFSET	TRAGEDY	GAMES	HEARING	SCIENCE	CAREFULLY
GRAPHIC	**CHARACTERS**	TRY	INNOCENT	FACTS	DECIDE
SURFACE	SCENES	COACH	DEFENSE	DATA	IMPORTANT
PRODUCED	OPERA	GYM	CHARGE	RESULTS	NOTEBOOK
CHARACTERS	PERFORMED	SHOT	CRIMINAL	EXPLANATION	REVIEW

Three documents with the word "play"

(numbers & colors → topic assignments)

A **Play**^{082} is written^{082} to be performed^{082} on a stage^{082} before a live^{093} audience^{082} or before motion^{270} picture^{004} or television^{004} cameras^{004} (for later^{054} viewing^{004} by large^{202} audiences^{082}). A **Play**^{082} is written^{082} because playwrights^{082} have something

He was listening^{077} to music^{077} coming^{009} from a passing^{043} riverboat. The music^{077} had already captured^{006} his heart^{157} as well as his ear^{119}. It was jazz^{077}. Bix beiderbecke had already had music^{077} lessons^{077}. He wanted^{268} to **play**^{077} the cornet. And he wanted^{268} to **play**^{077} jazz^{077}

Jim^{296} **plays**^{166} the game^{166}. Jim^{296} likes^{081} the game^{166} for one. The game^{166} book^{254} helps^{081} jim^{296}. Don^{180} comes^{040} into the house^{038}. Don^{180} and jim^{296} read^{254} the game^{166} book^{254}. The boys^{020} see a game^{166} for two. The two boys^{020} **play**^{166} the game^{166}

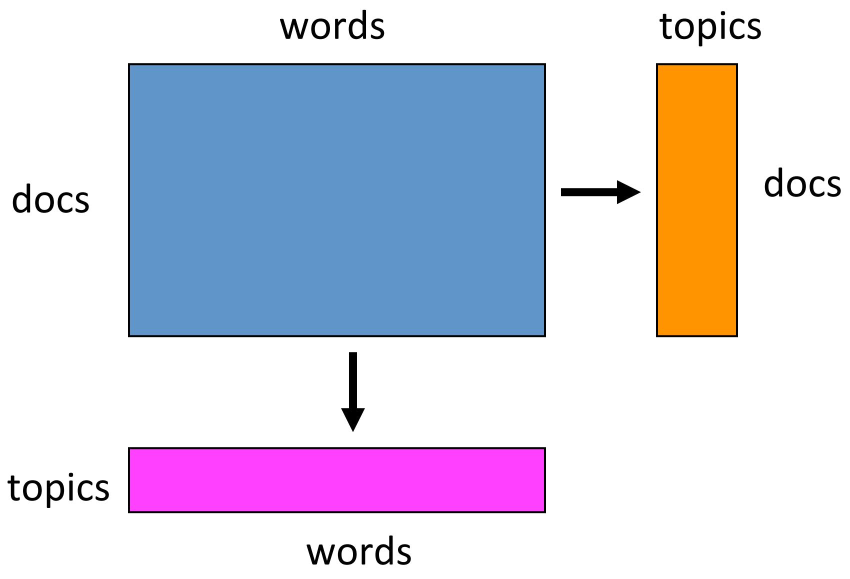
LDA Gibbs Sampling: Inputs And Outputs

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



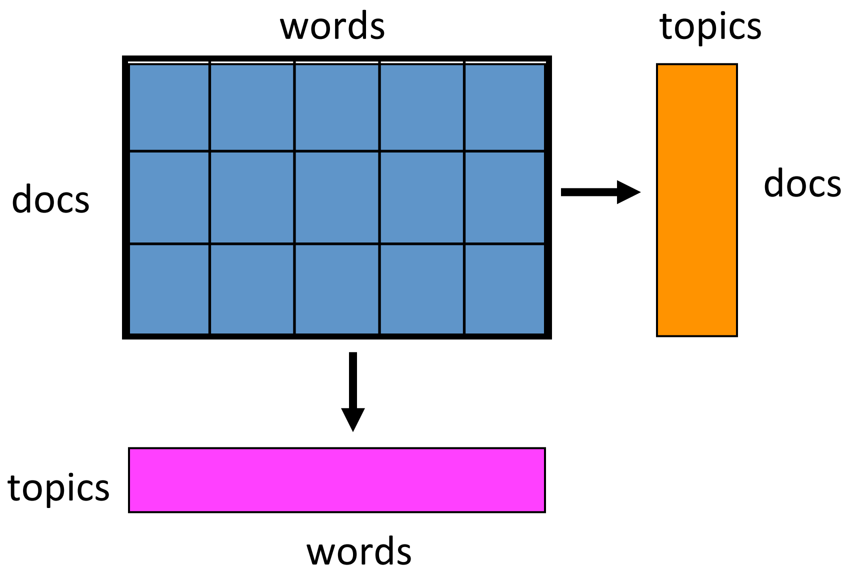
LDA Gibbs Sampling: Inputs And Outputs

Inputs:

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

Outputs:

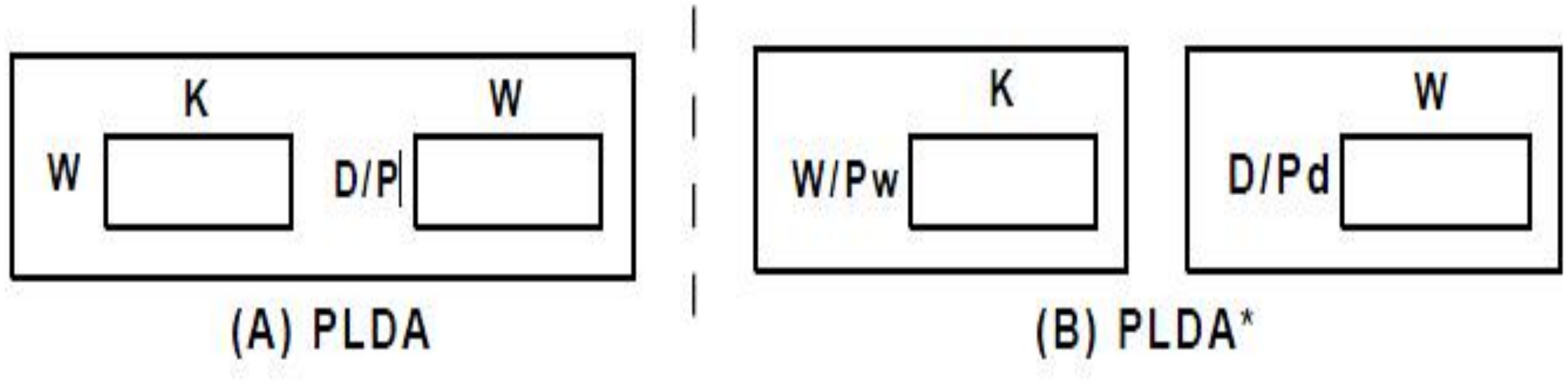
1. A co-occurrence matrix of topics and documents
2. A co-occurrence matrix of topics and words



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 <u>a, a, a</u>, <u>b, b</u>, <u>c, c, c</u>, 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 tf + tu < ts (4(A) is good, 4(B) suboptimal)

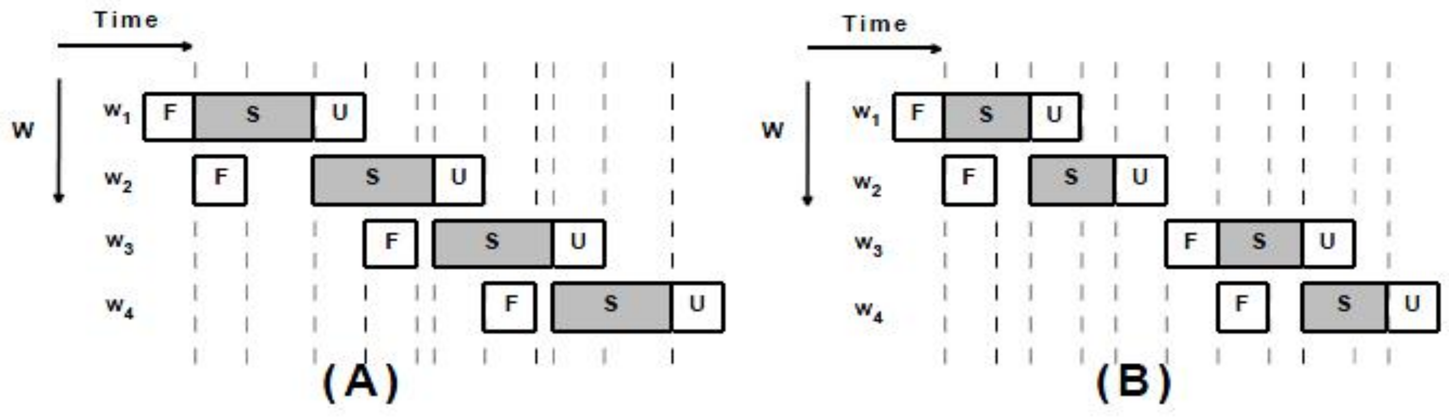


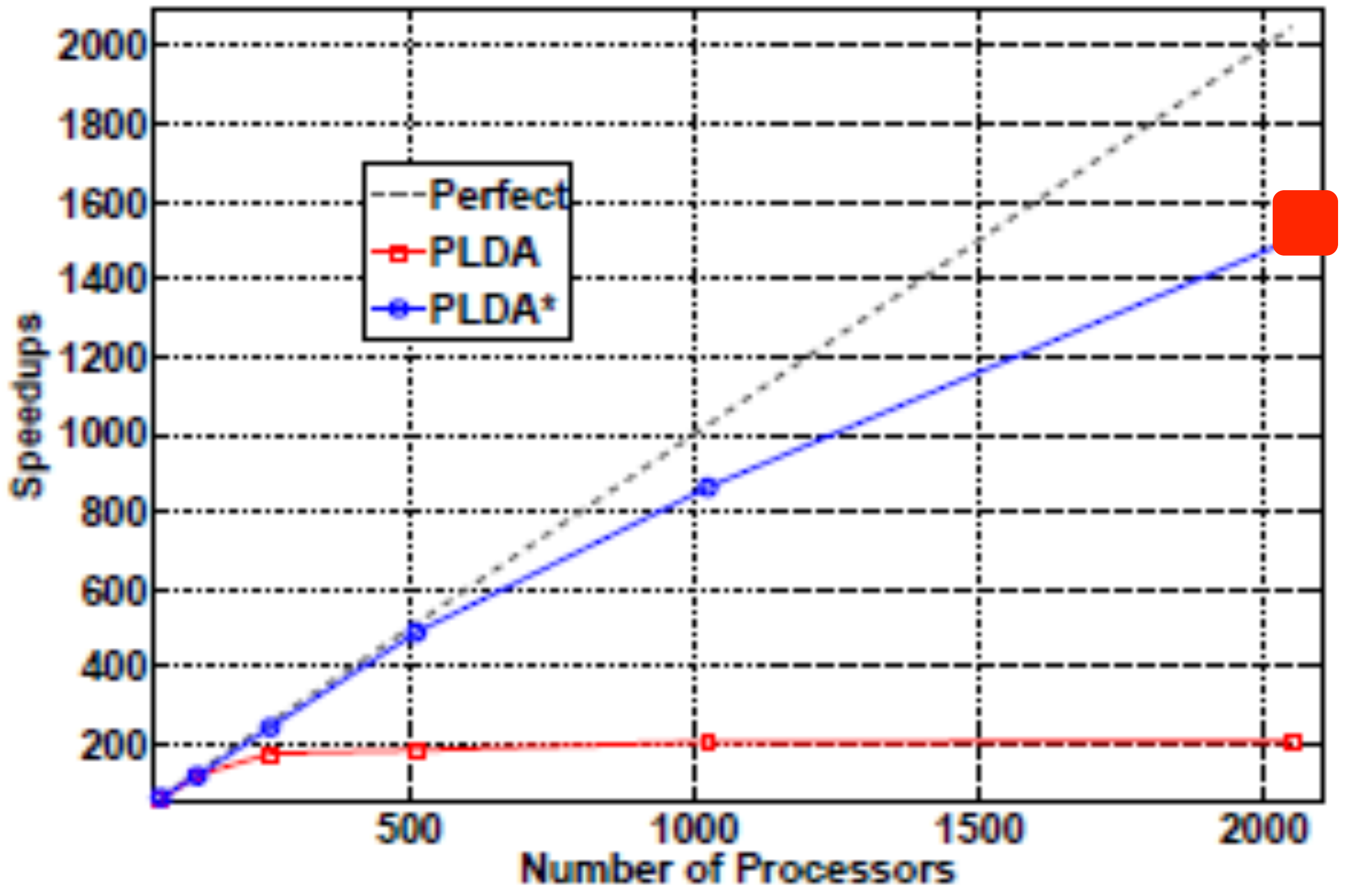
Fig. 4: Pipeline-based Gibbs Sampling in PLDA*. (A): $t_s \geq t_f + t_u$. (B): $t_s < t_f + t_u$.

MapReduce VS. MPI?

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

Speedup

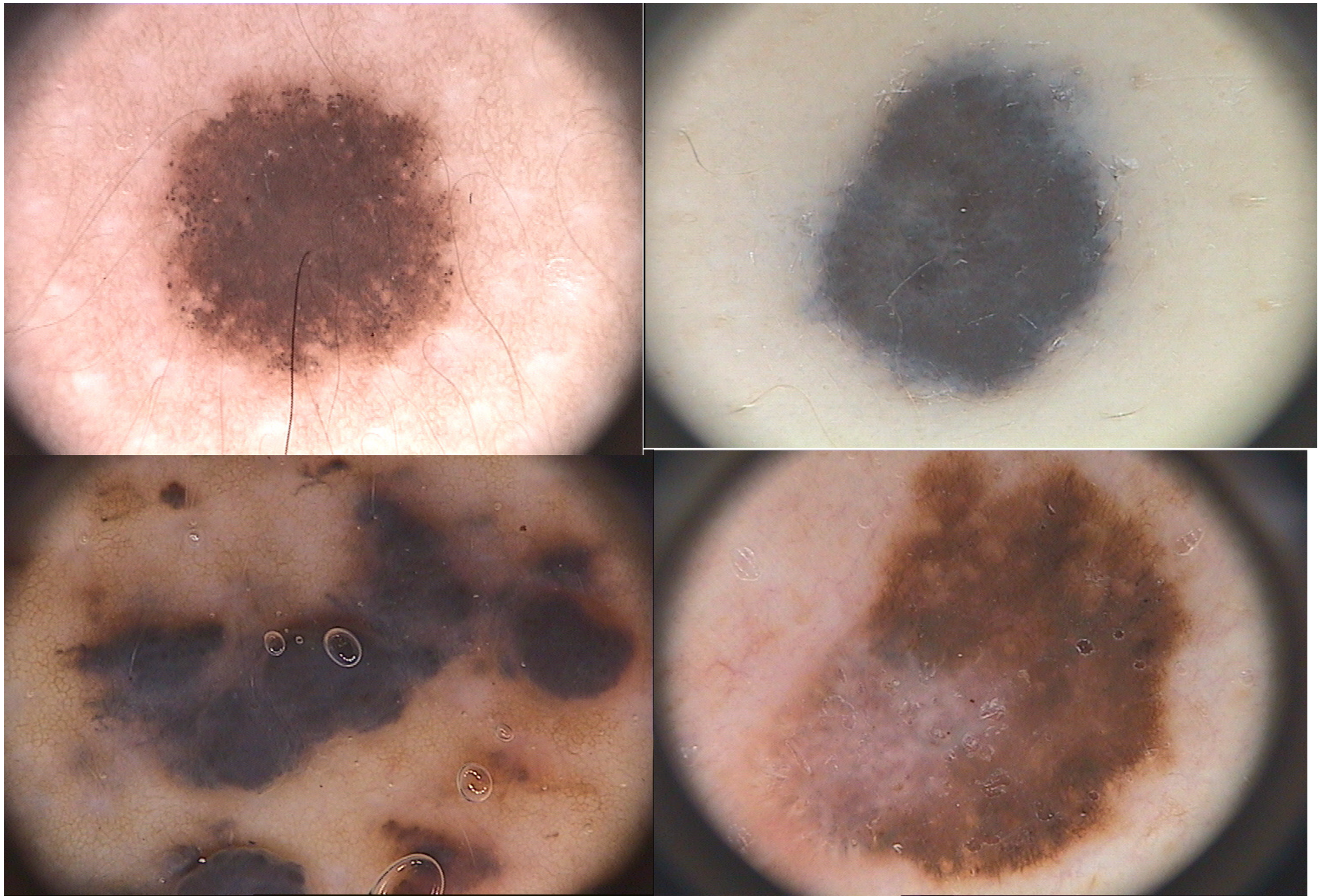
1,500x using 2,000 machines



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

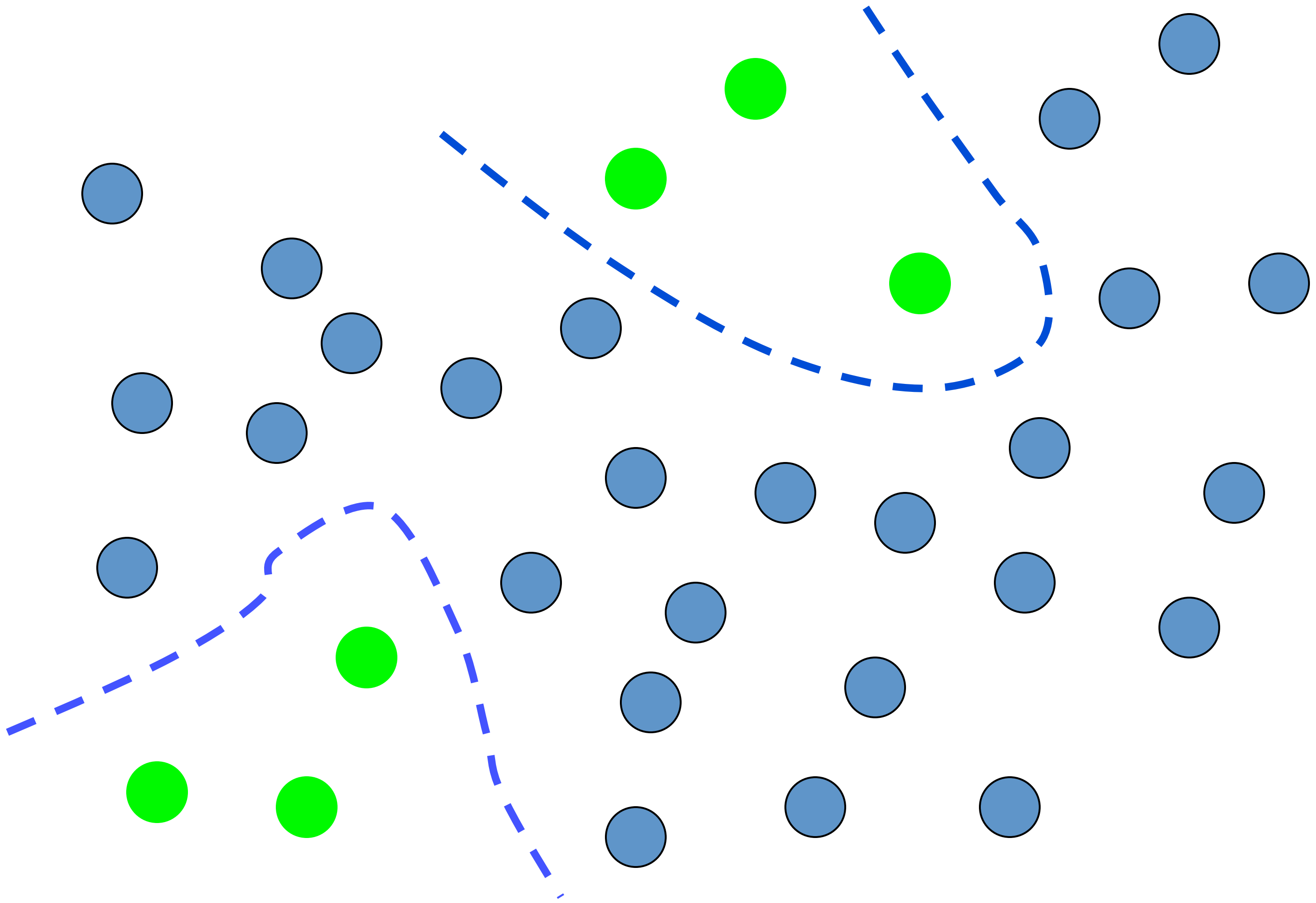
- Acquire labeled data (most data are unlabeled)
- Formulate distance function
- Train a classifier
- Classify unlabeled data
  - Fast
  - Low power consumption

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)

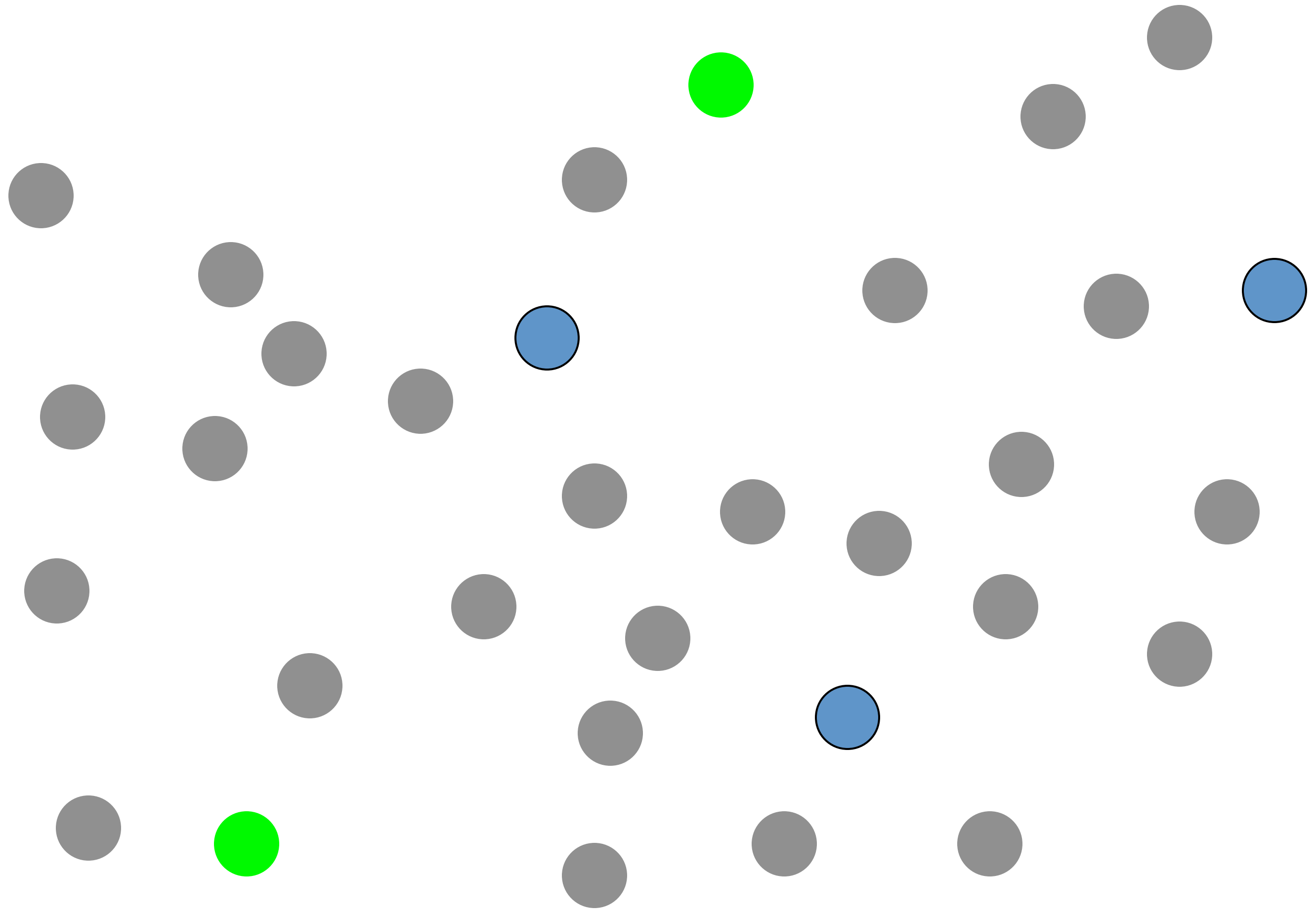
IR → A Classification Problem

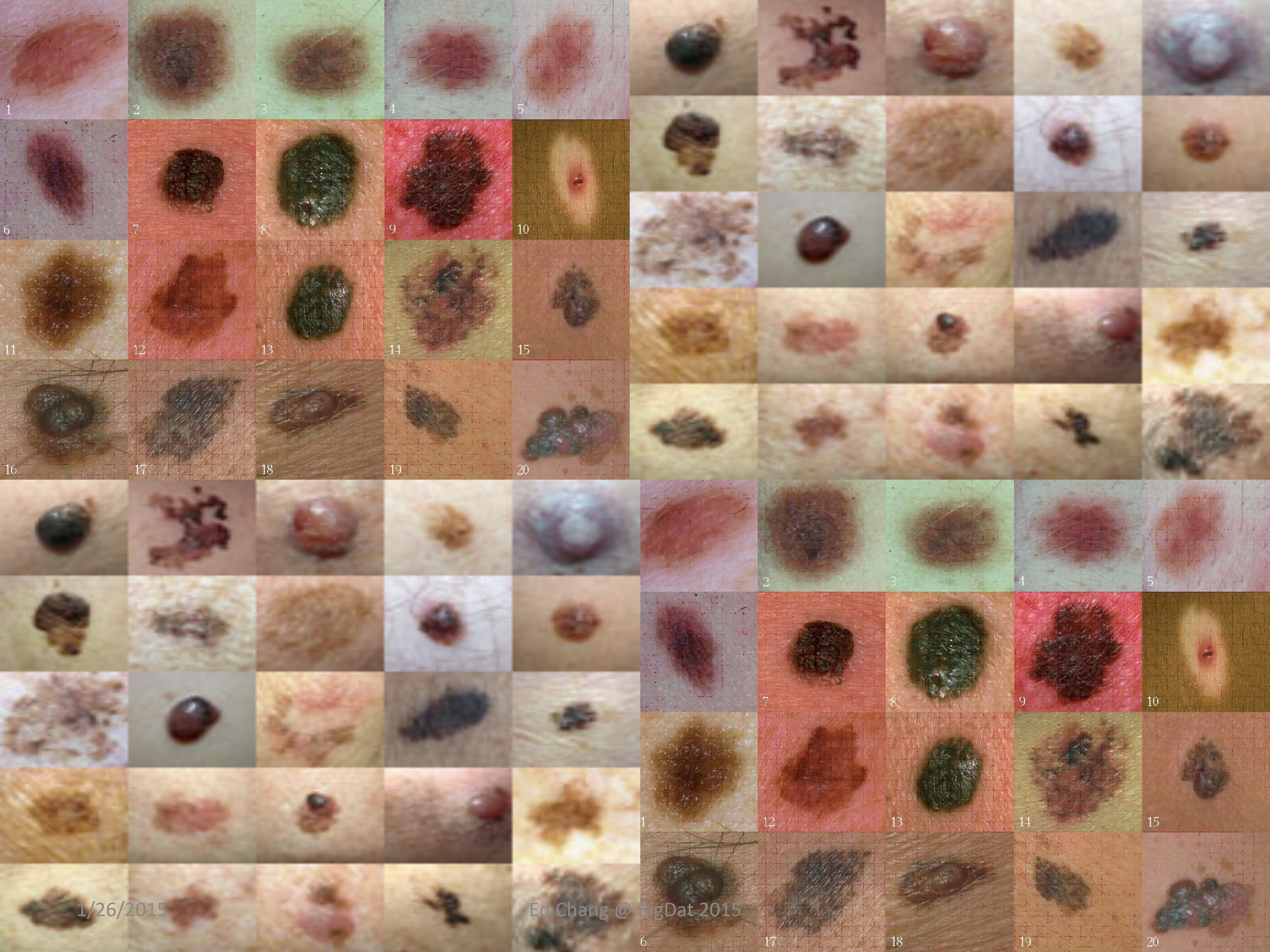
Use SVMActive to Acquire Training Data



IR → A Classification Problem

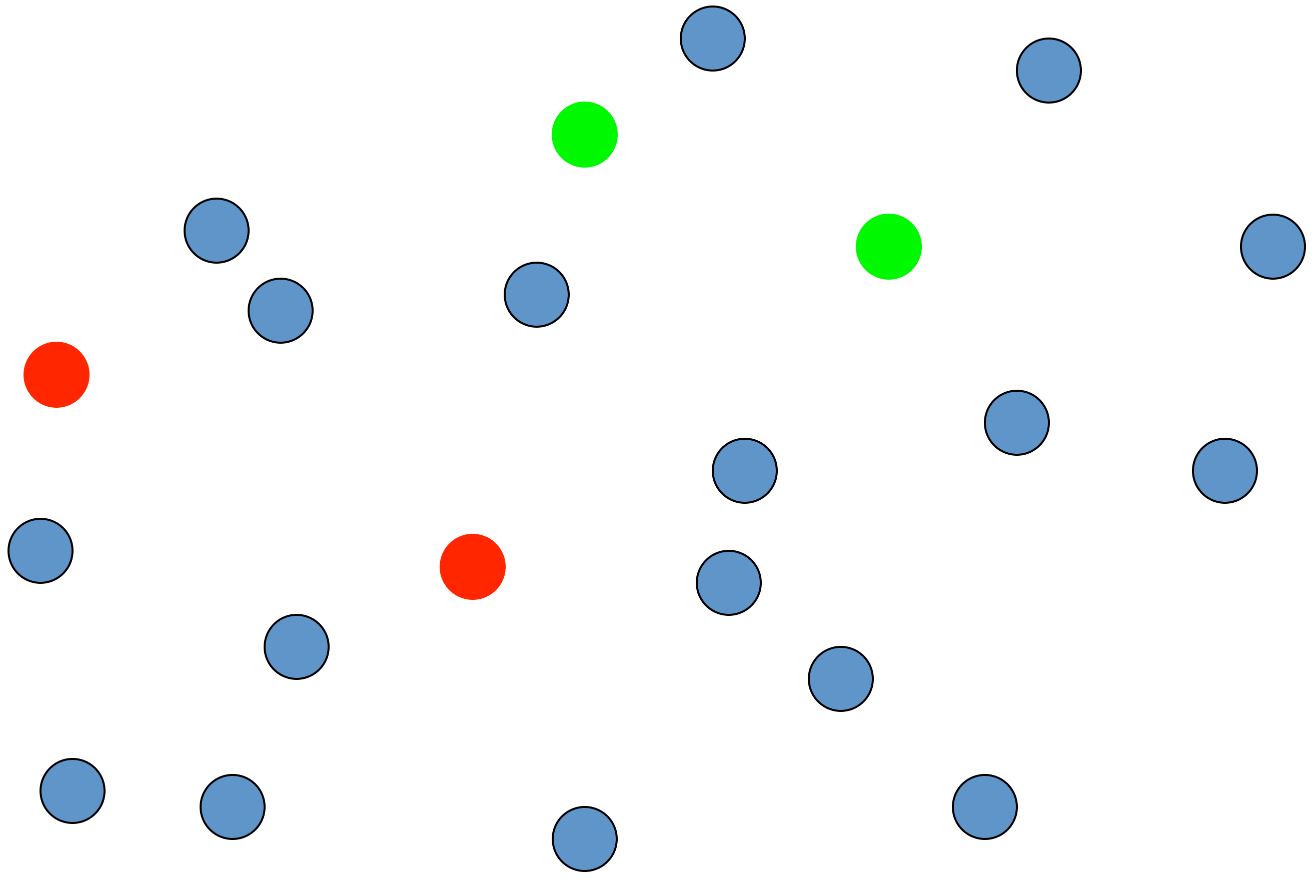
Most Data are Unlabeled



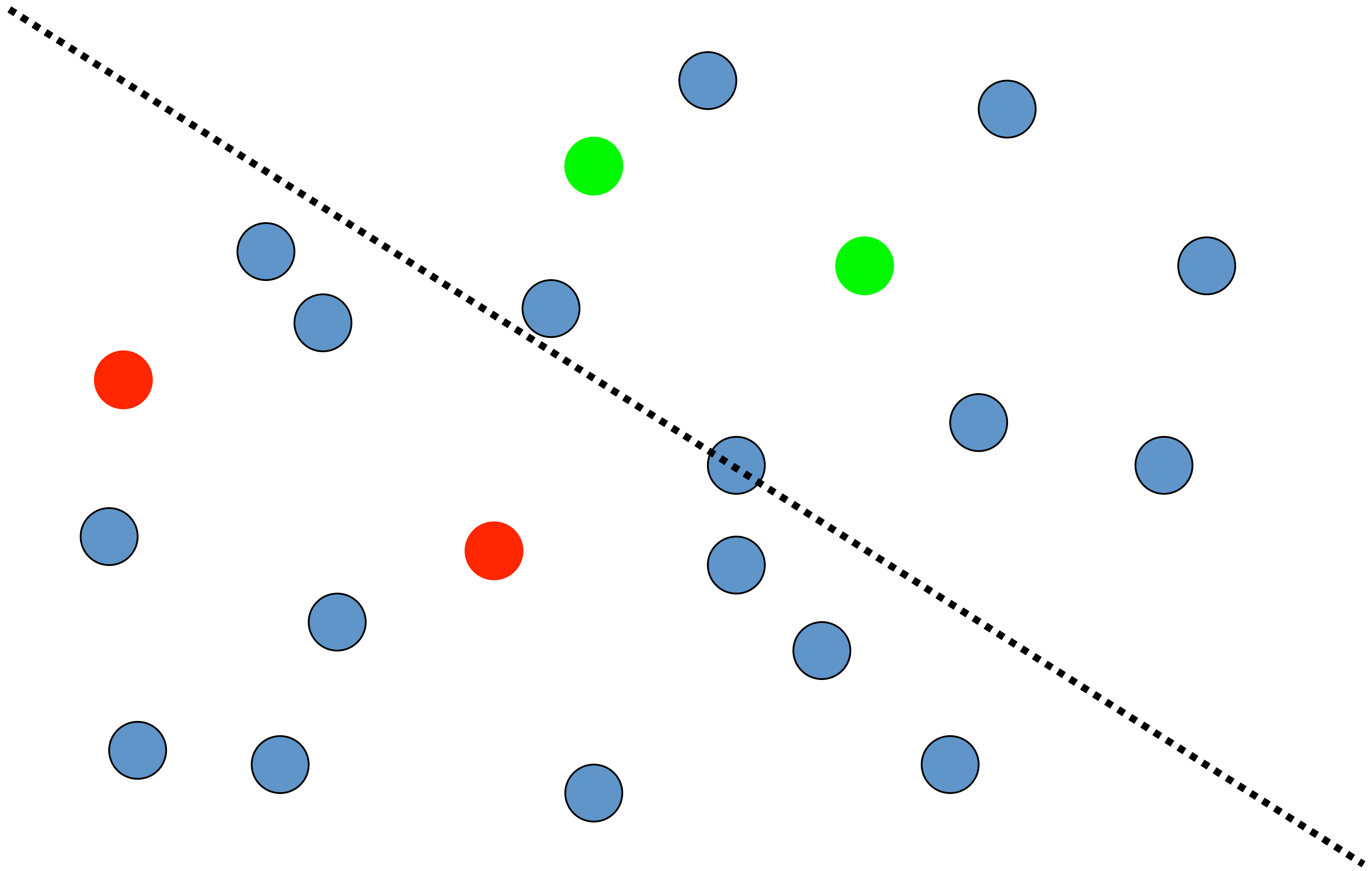

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

Step #1: Solicit Labels

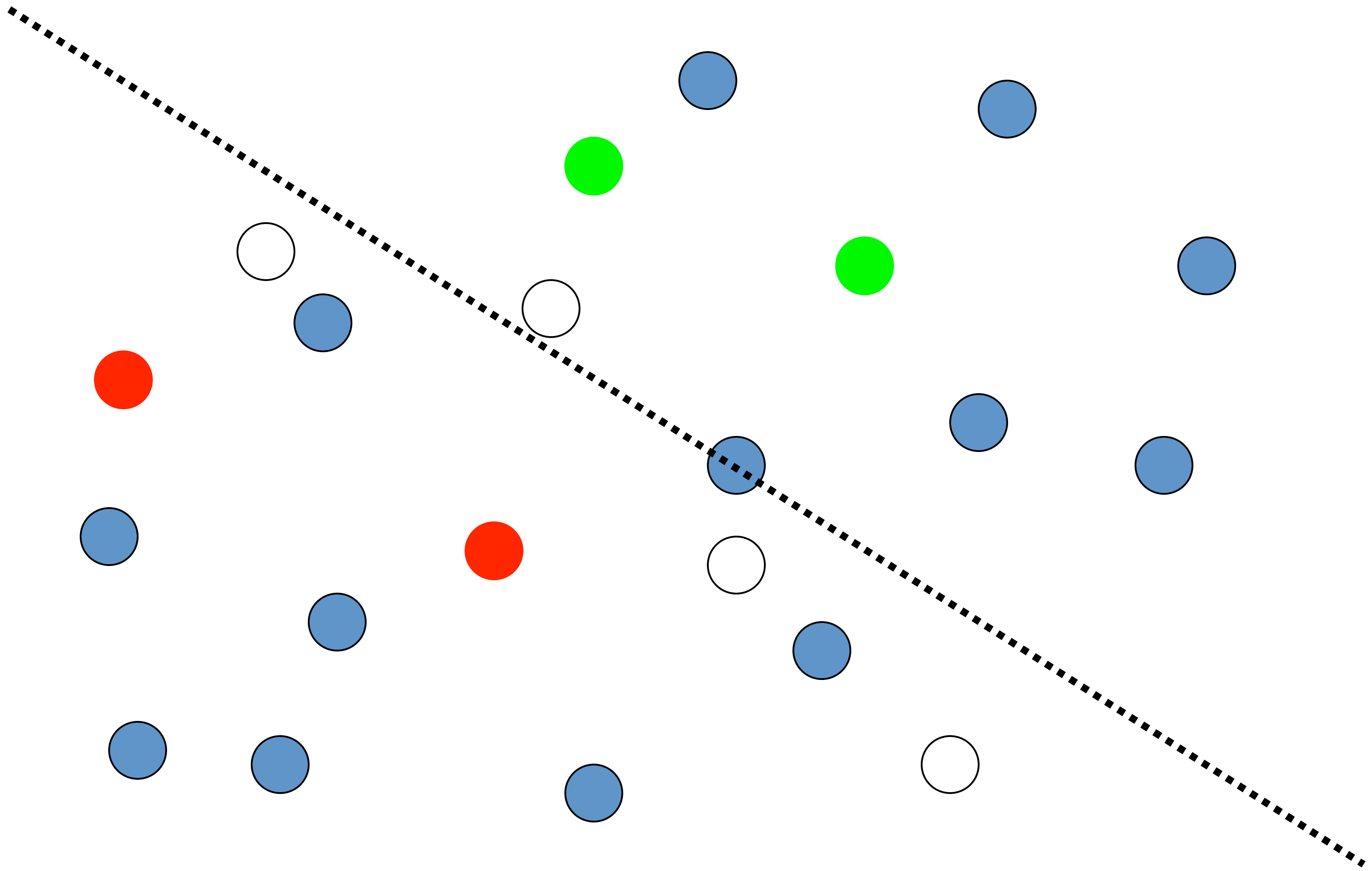
Via Active Learning [MM 01]

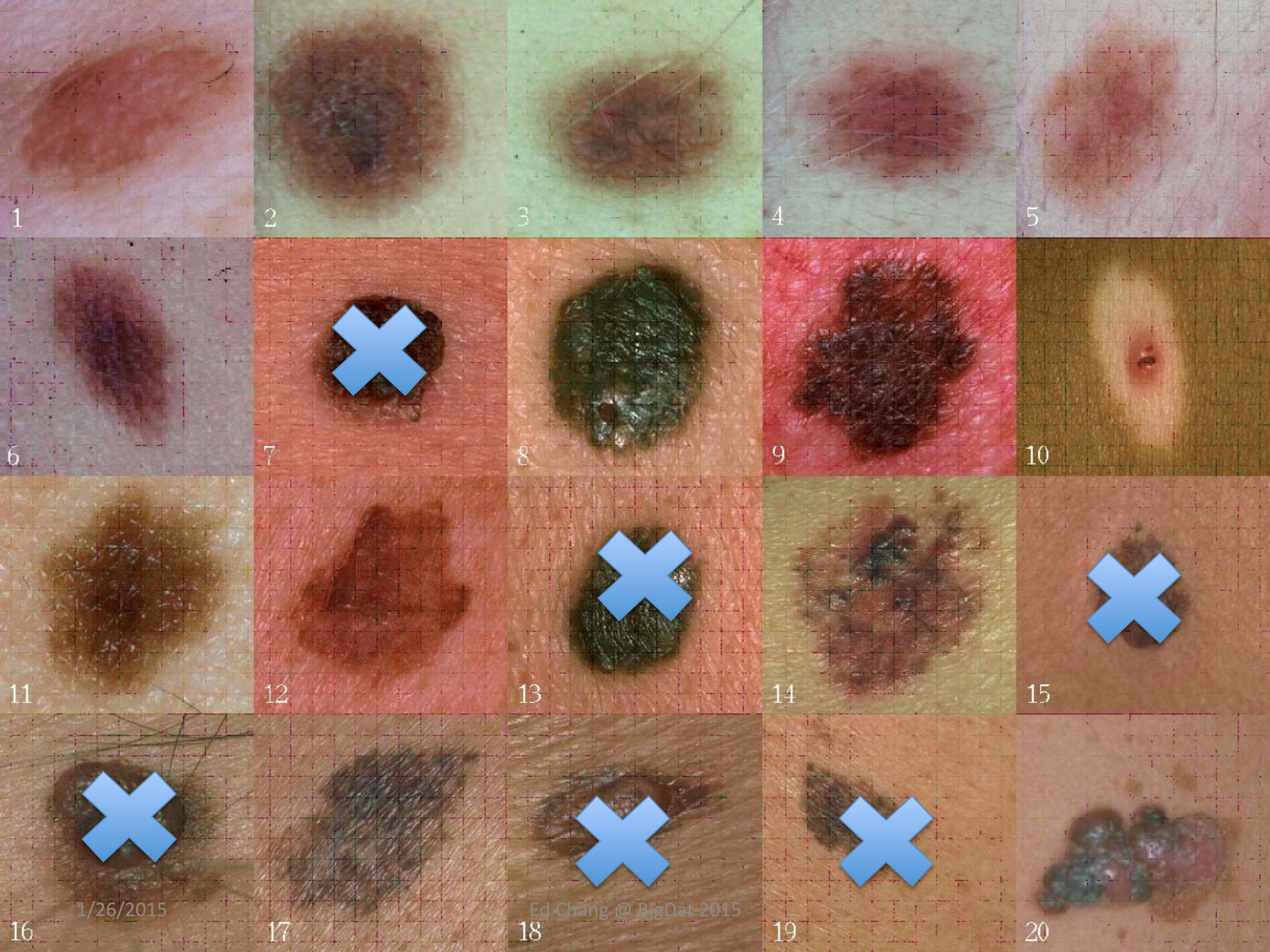


Step #2: Compute Boundary

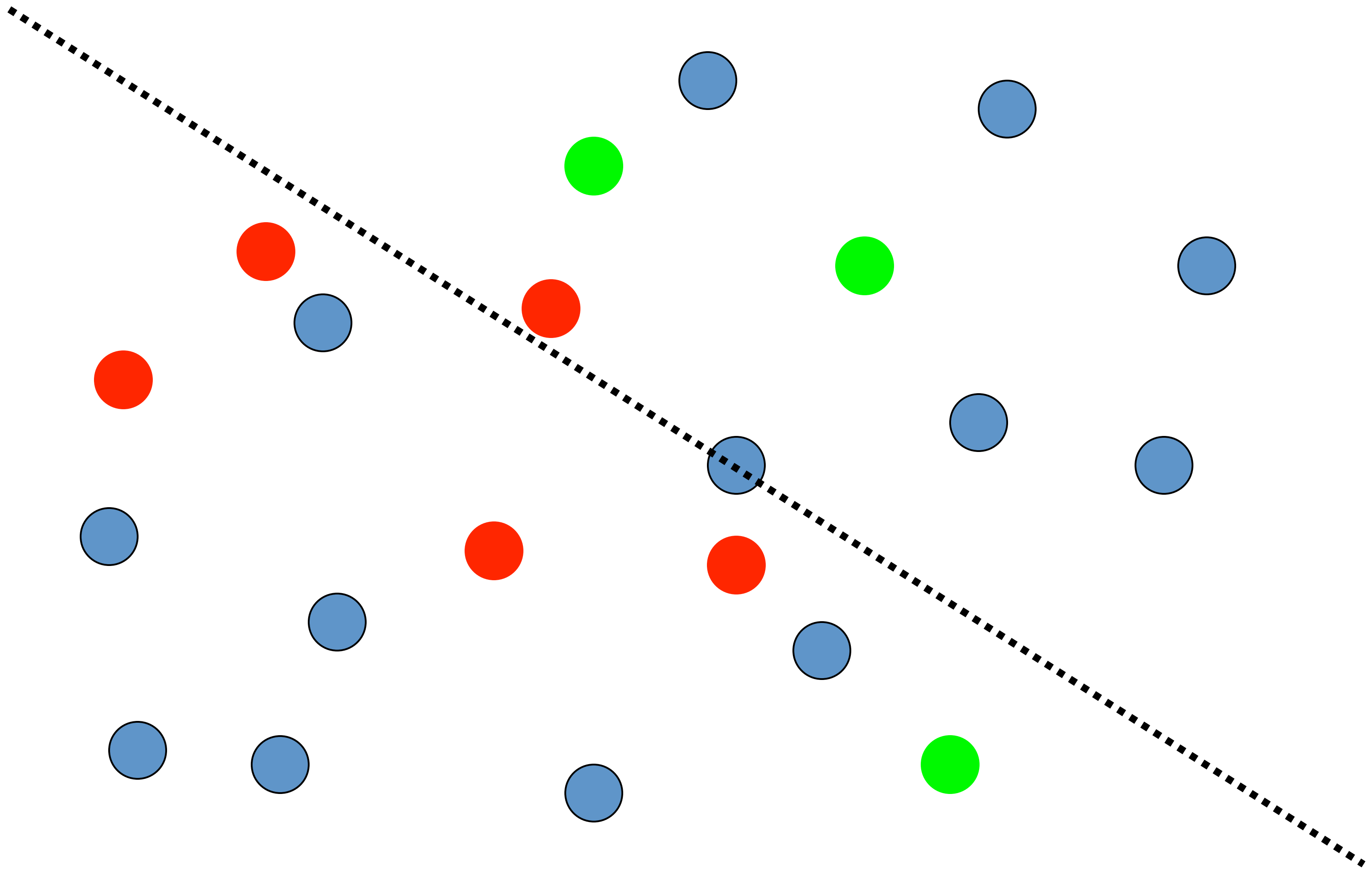


Step #3: Identify Useful Samples

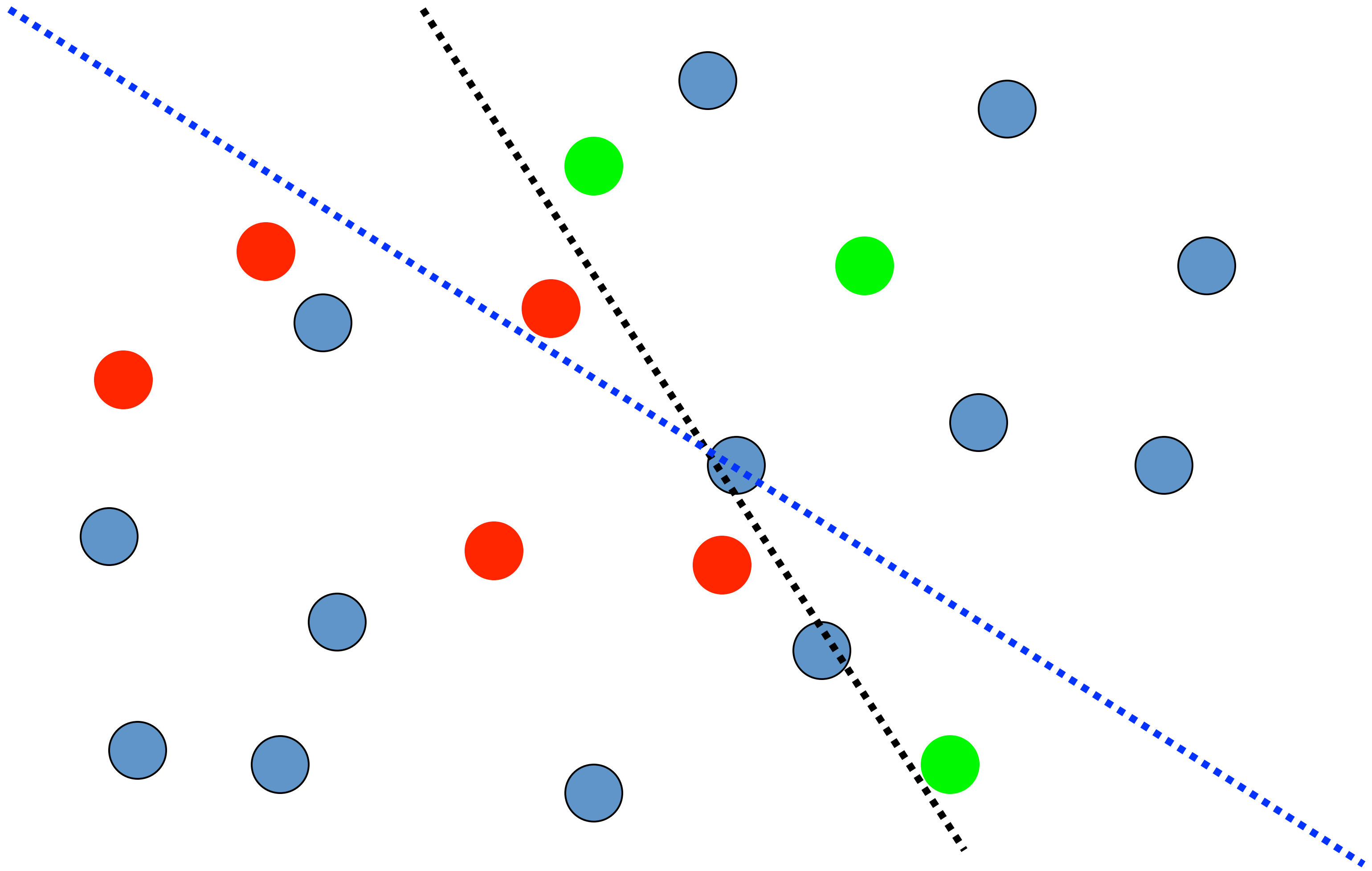



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

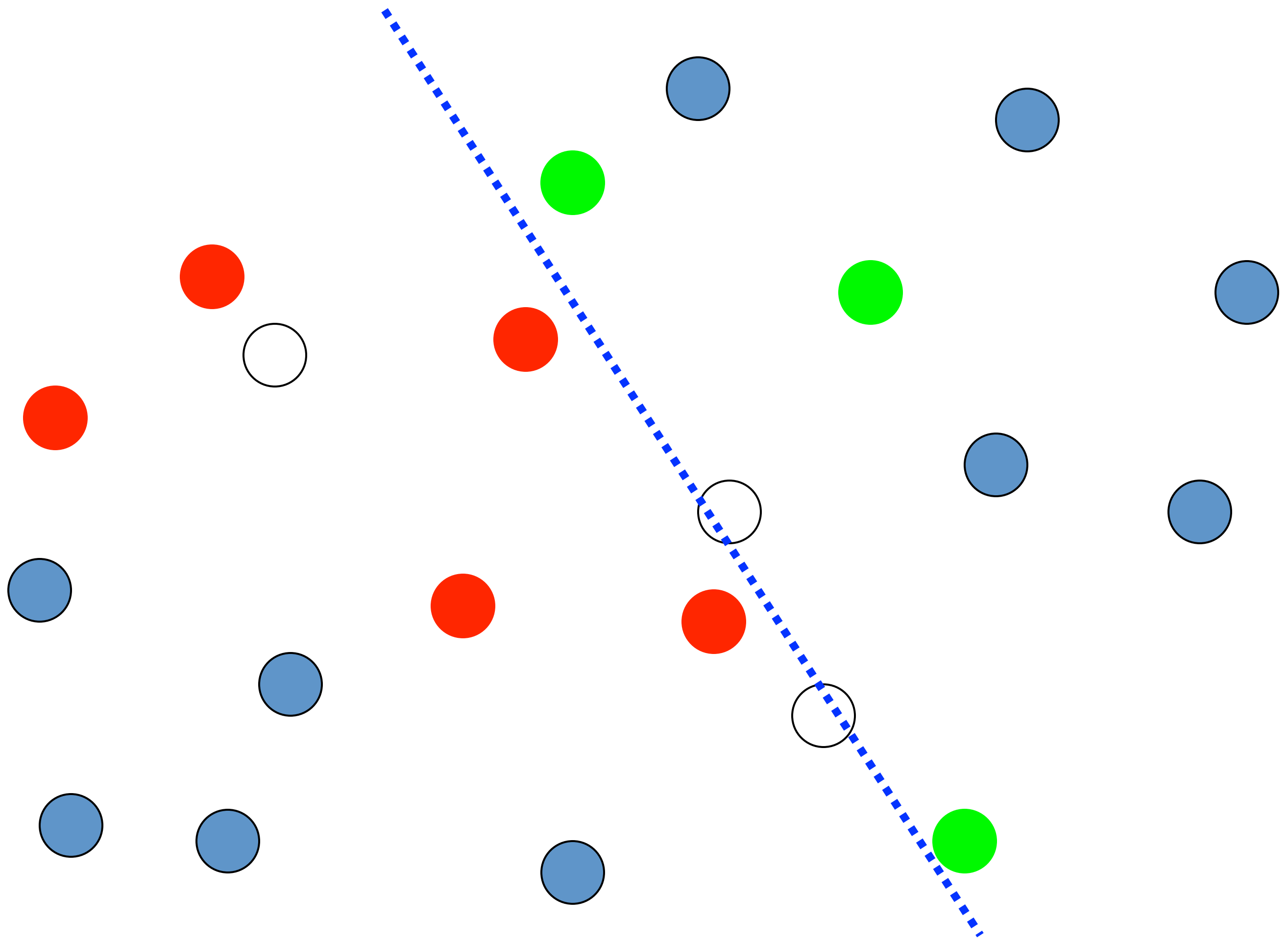
Step #4: Solicit Feedback



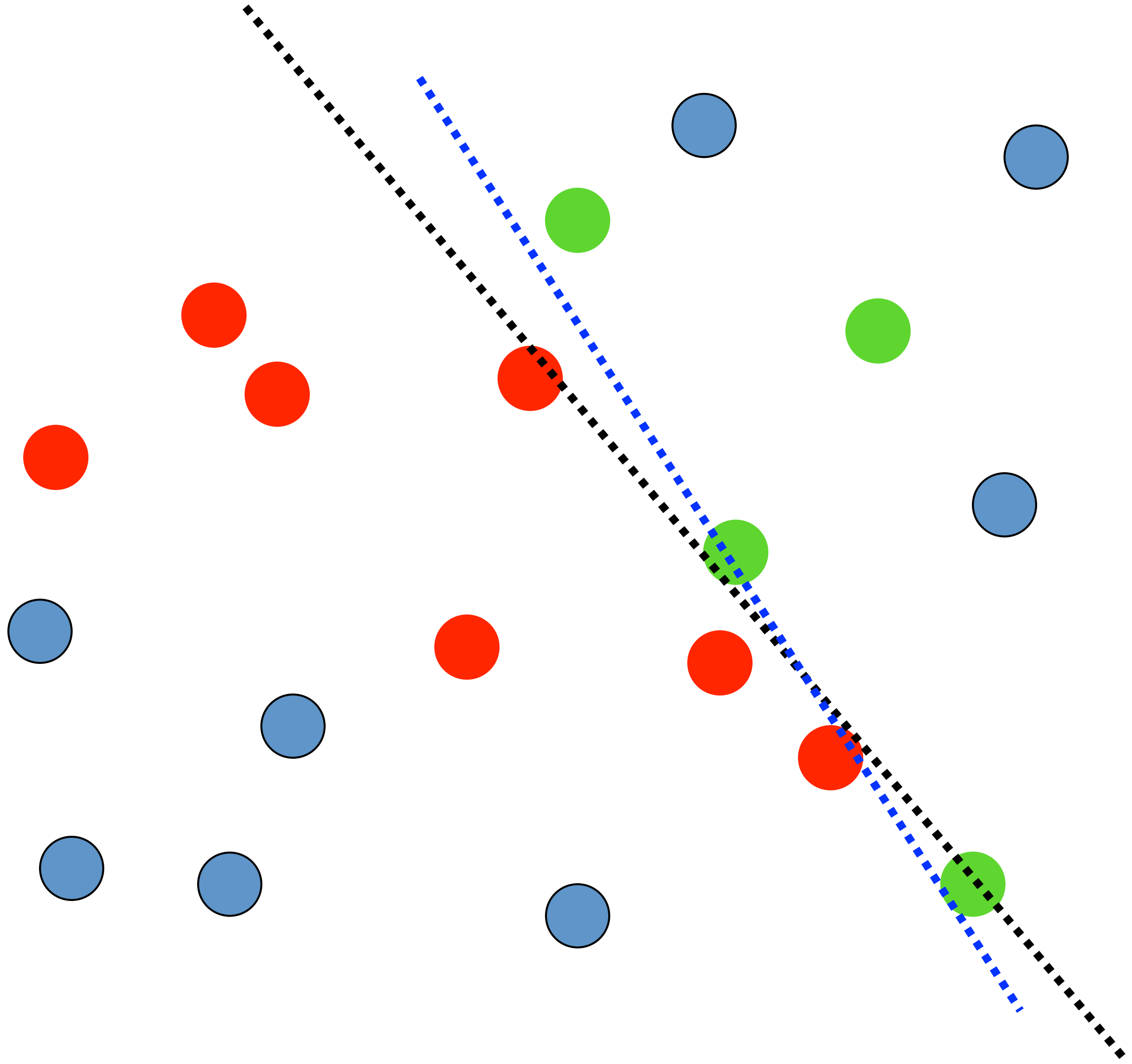
Step #5: Refine Boundary



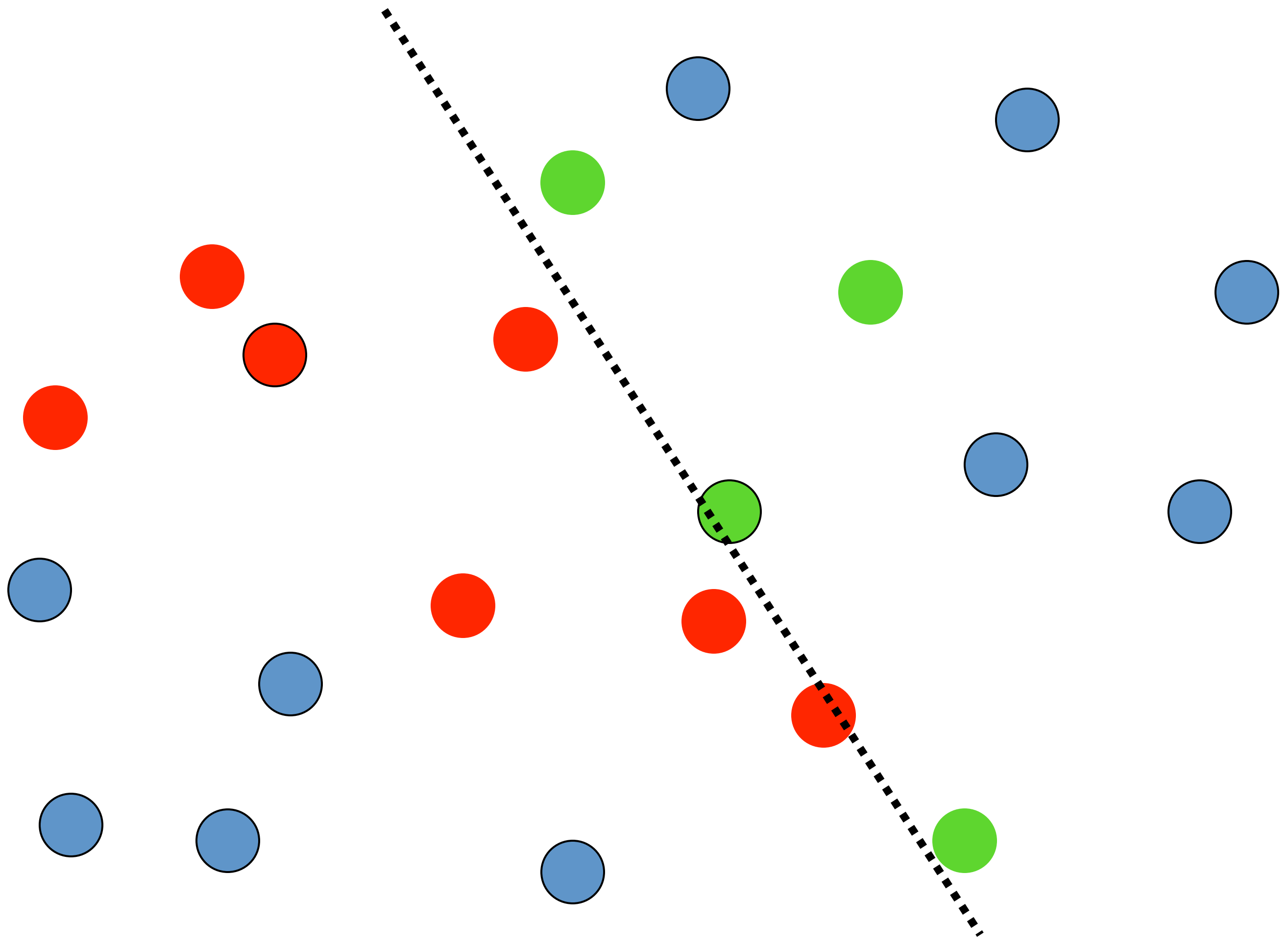
Step #6: Identify Samples



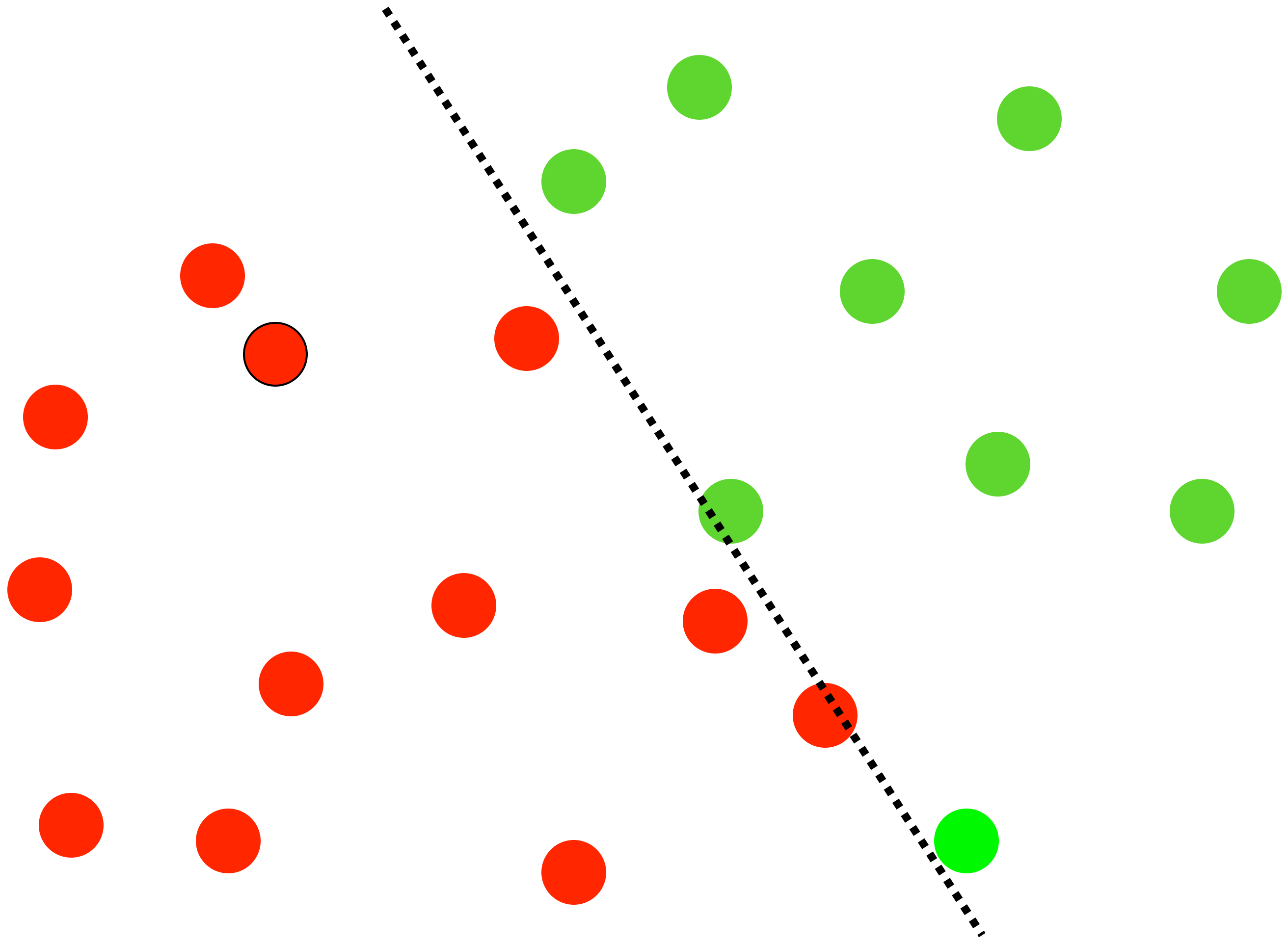
Step #7: User Feedback



Step #8: Refine Boundary



Step #9: Classify Data



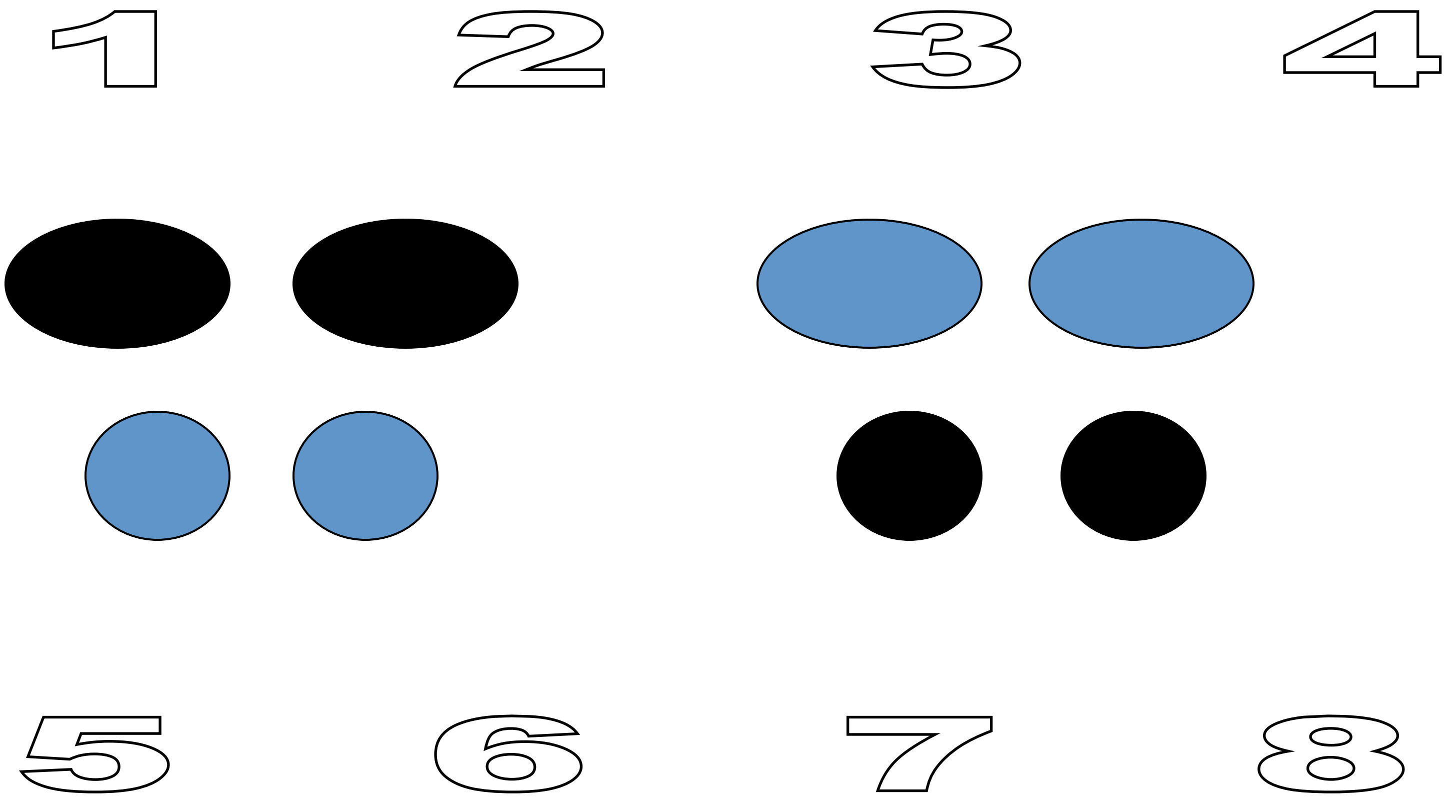
Observations

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

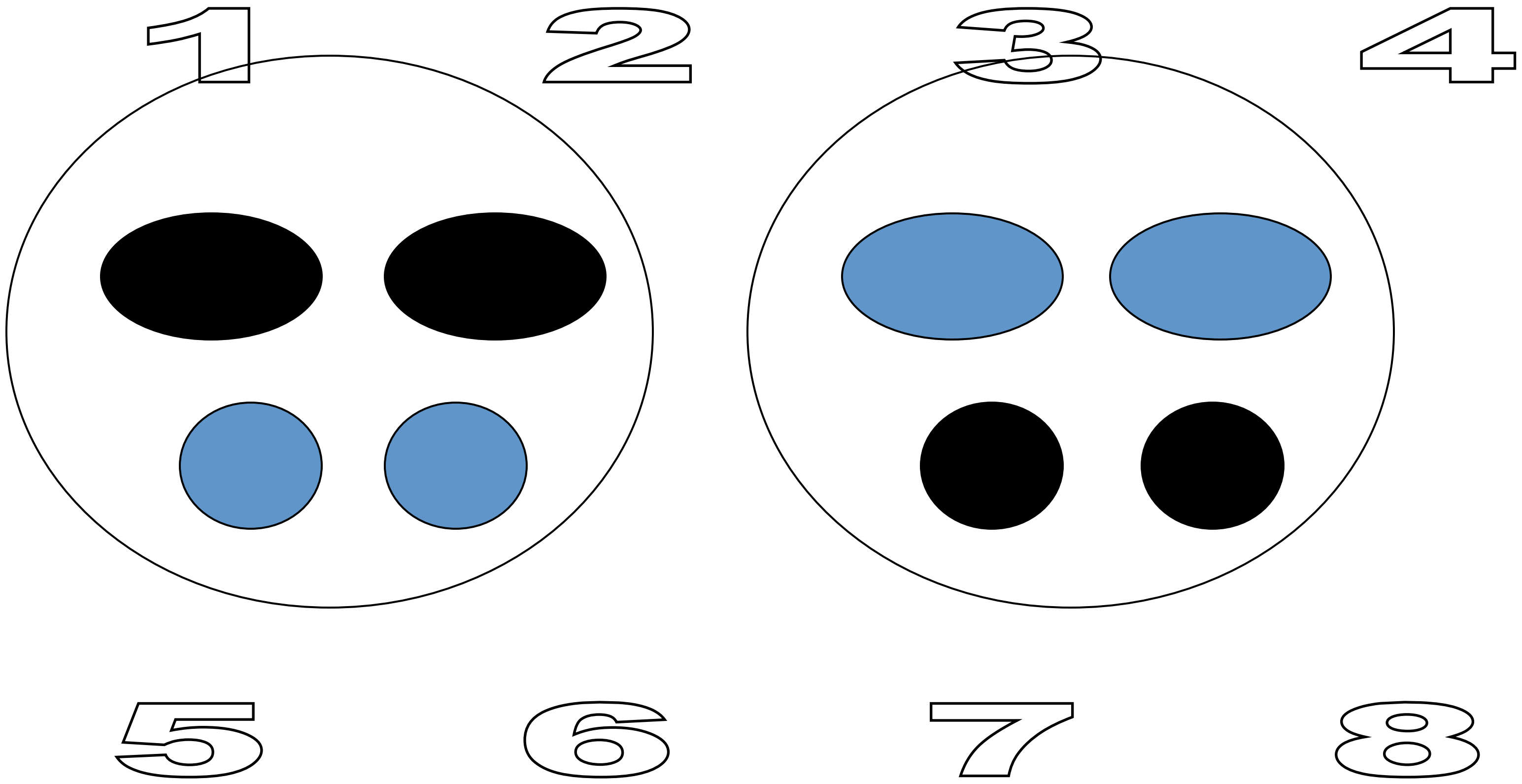
- Next, how to quantify similarity?
  - One way is to hand-craft a kernel matrix
  - The other is to learn a good manifold

Similarity?

Distance Function Formulation



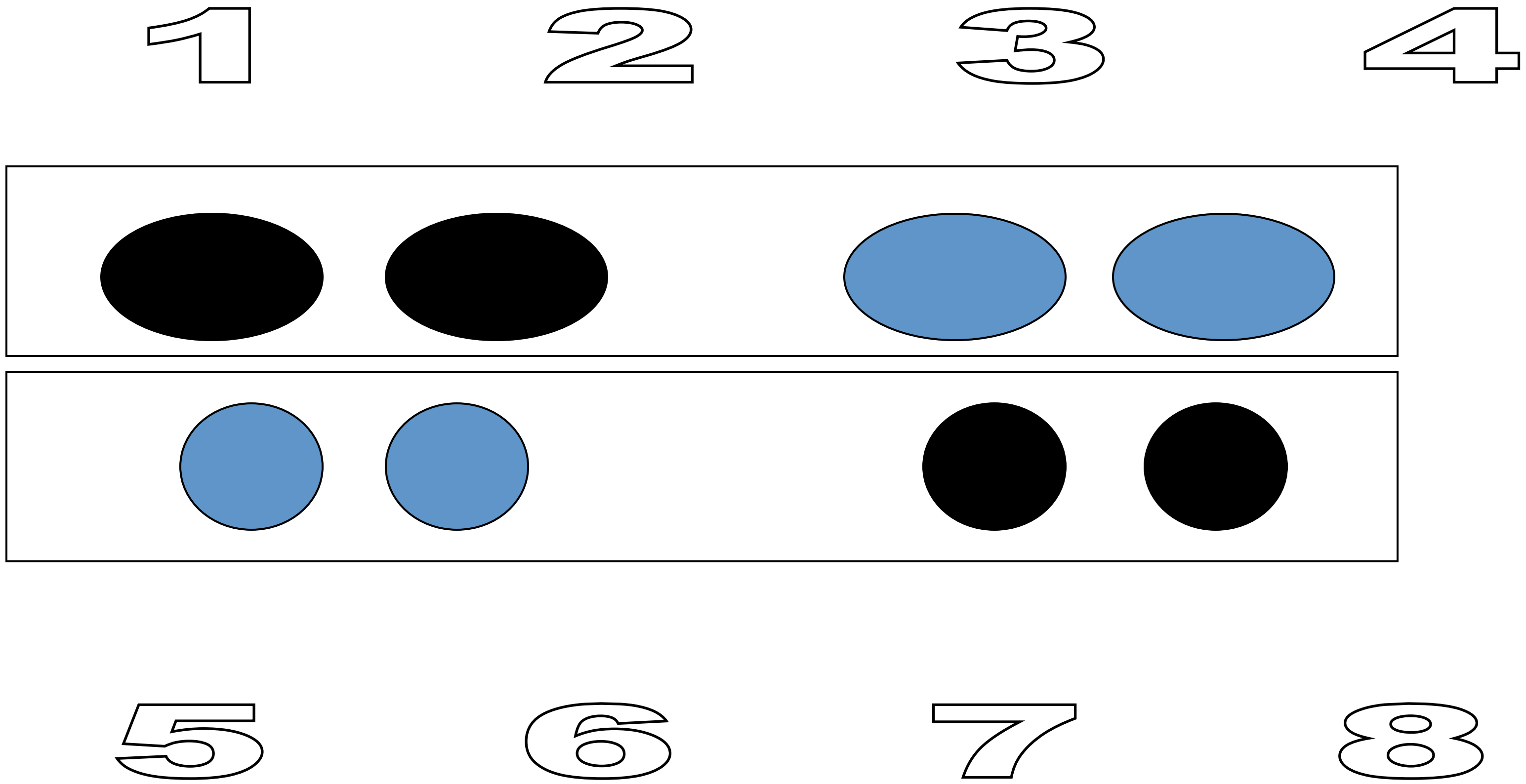
Group by Proximity



Group by Proximity

	x1	x2	x3	x4	x5	x6	x7	x8
x1	1	.7	.4	.3	.7	.6	.2	.1
x2		1	.4	.3	.6	.7	.3	.2
x3			1	.7	.3	.4	.7	.6
x4				1	.1	.2	.6	.7
x5					1	.7	.3	.2
x6						1	.6	.4
x7							1	.7
x8								1

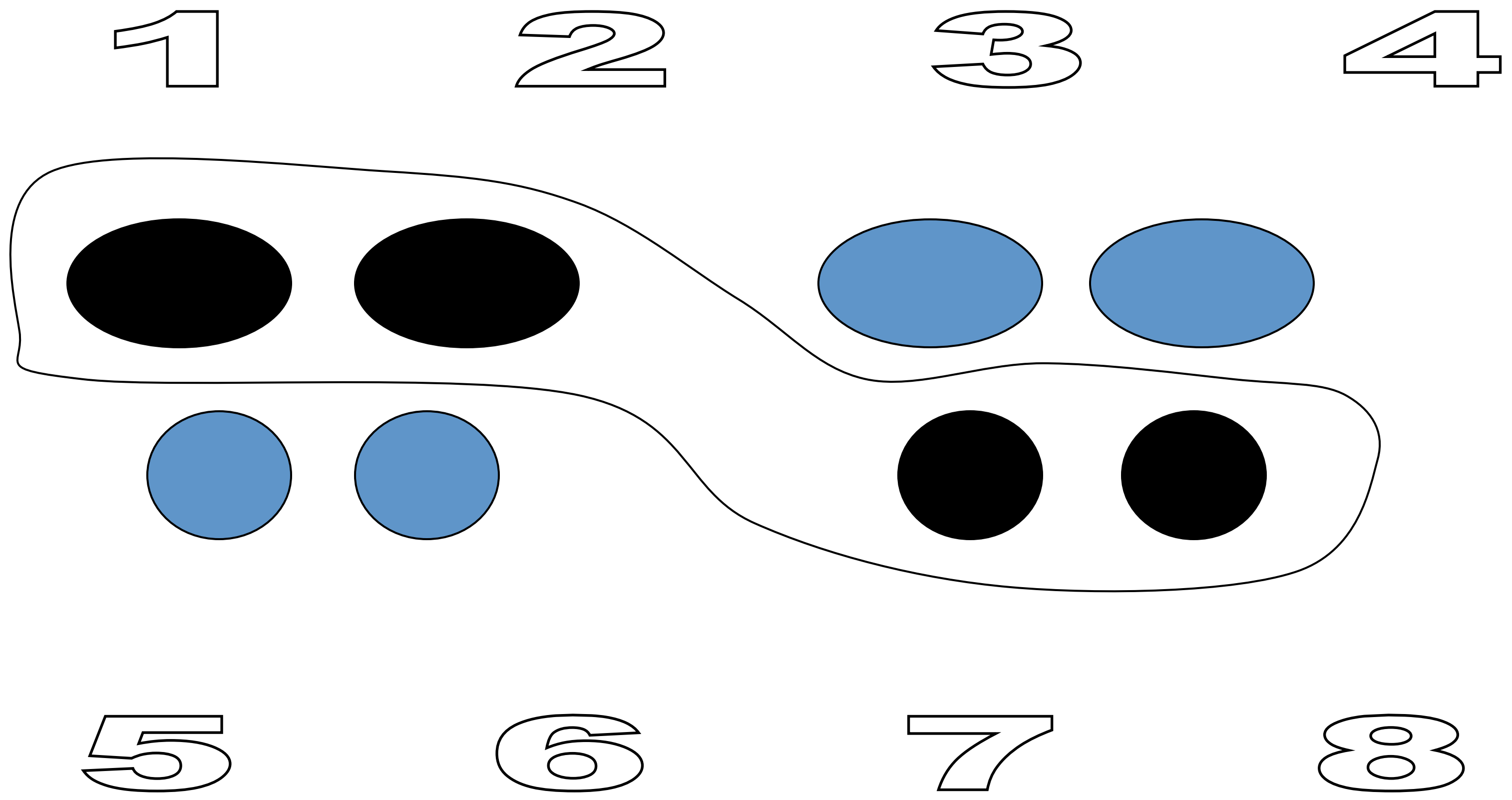
Group by Shape



Group by Shape

	x1	x2	x3	x4	x5	x6	x7	x8
x1	1	.7	.7	.7	.2	.2	.2	.2
x2		1	.7	.7	.2	.2	.2	.2
x3			1	.7	.2	.2	.2	.2
x4				1	.2	.2	.2	.2
x5					1	.7	.7	.7
x6						1	.7	.7
x7							1	.7
x8								1

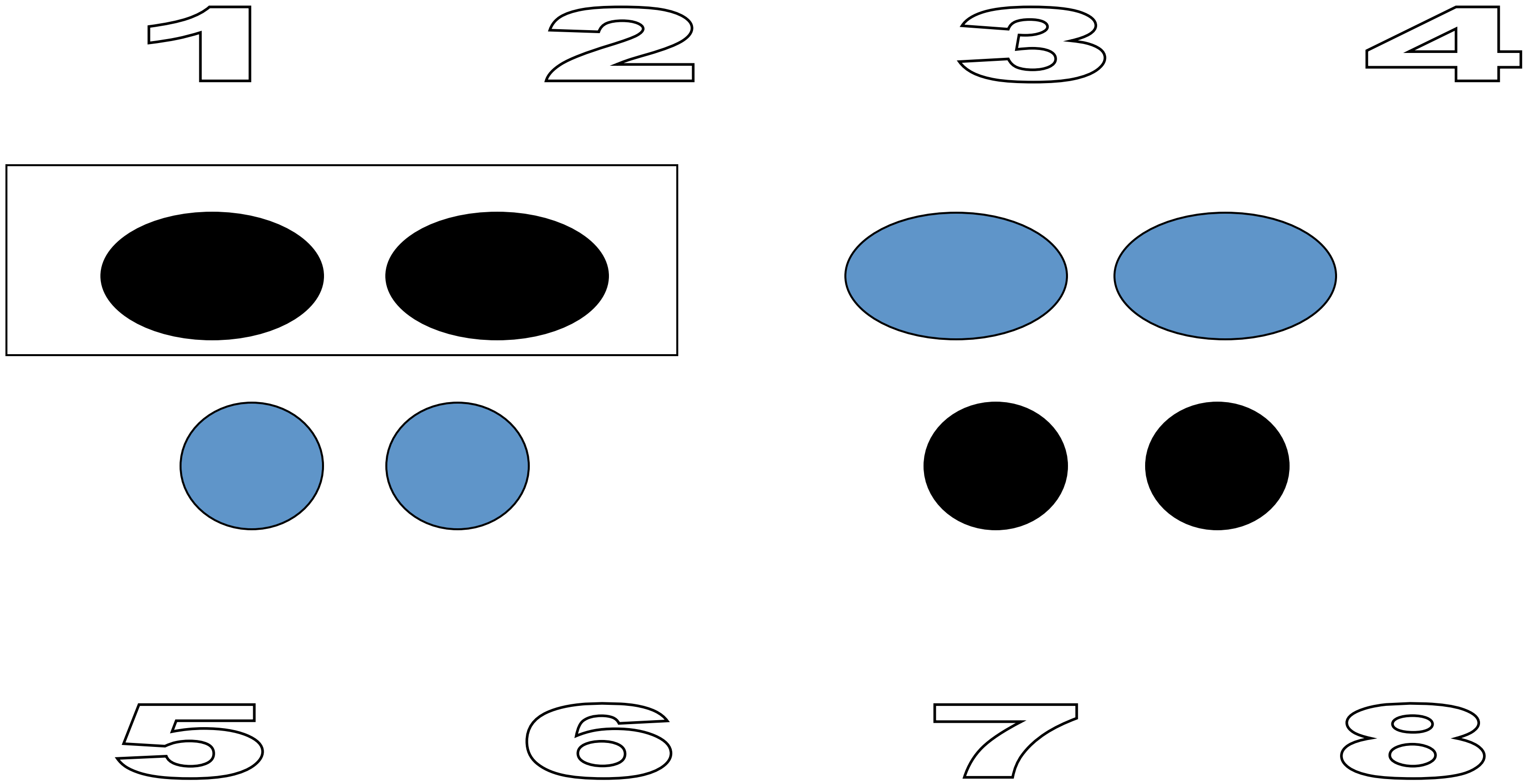
Group by Color

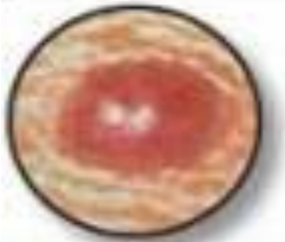
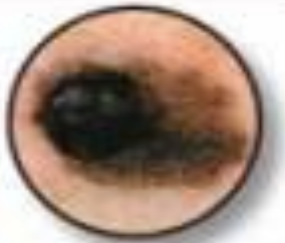
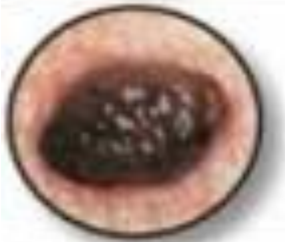
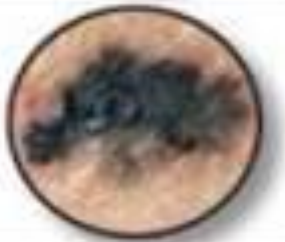
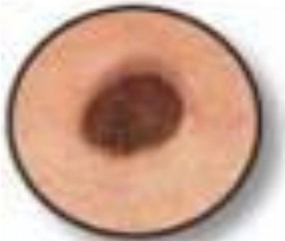
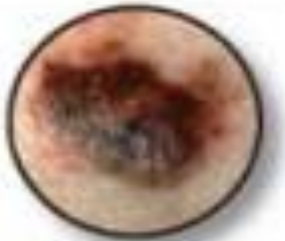
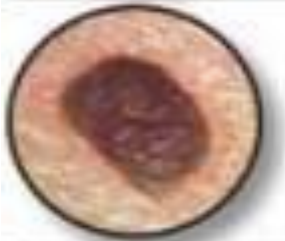
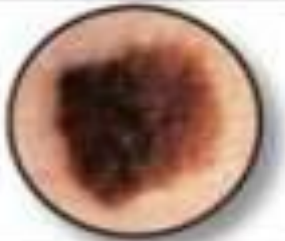
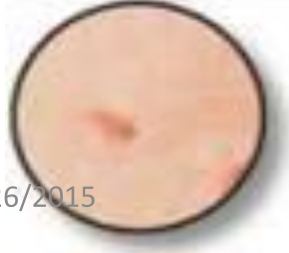
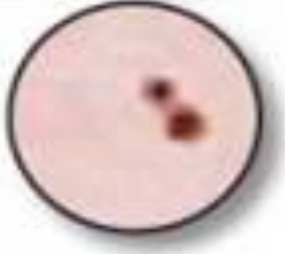


Group by Color

	x1	x2	x3	x4	x5	x6	x7	x8
x1	1	.7	.3	.3	.3	.2	.2	.7
x2		1	.3	.3	.3	.3	.7	.7
x3			1	.7	.7	.7	.3	.3
x4				1	.7	.7	.3	.3
x5					1	.7	.3	.3
x6						1	.3	.3
x7							1	.7
x8								1

Similarity?
Distance Function Formulation



NORMAL		CANCEROUS
	**"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.	
	**"B" IS FOR BORDER** • The edges of an early melanoma tend to be uneven, crusty or notched.	
	**"C" IS FOR COLOR** • Healthy moles are uniform in color. A variety of colors, especially white and/or blue, is bad.	
	**"D" IS FOR DIAMETER** • Melanomas are usually larger in diameter than a pencil eraser, although they can be smaller.	
	**"E" IS FOR EVOLVING** • When a mole changes in size, shape or color, or begins to bleed or scab, this points to danger.	

Group by Labels
Update the Kernel Matrix

	x1	x2	x3	x4	x5	x6	x7	x8
x1	1	.7	.3	.3	.3	.2	.2	.7
x2		1	.7	.3	.3	.3	.2	.7
x3			1	.7	.7	.7	.3	.3
x4				1	.7	.7	.3	.3
x5					1	.7	.3	.3
x6						1	.3	.3
x7							1	.7
x8								1

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

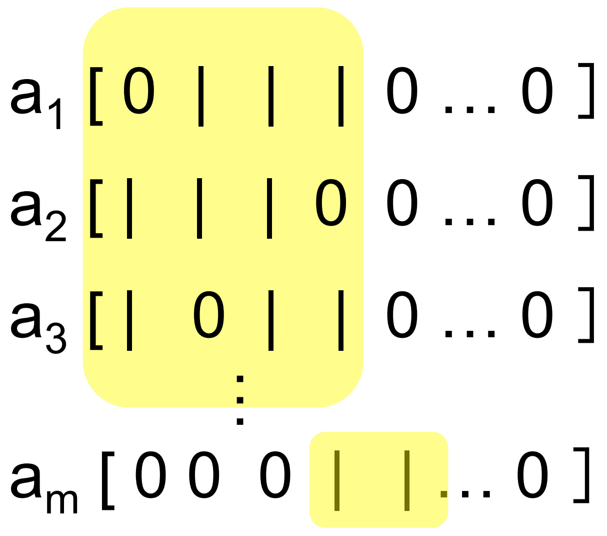
- Objects are similar in all or some respects
- Minkowski Function
  - $D = (\Sigma_{i = 1..M} (p_i - q_i)^n)^{1/n}$
- Weighted Minkowski Function
  - $D = (\Sigma_{i = 1..M,} w_i(p_i - q_i)^n)^{1/n}$
- Same w is imposed to app pairs of objects p and q

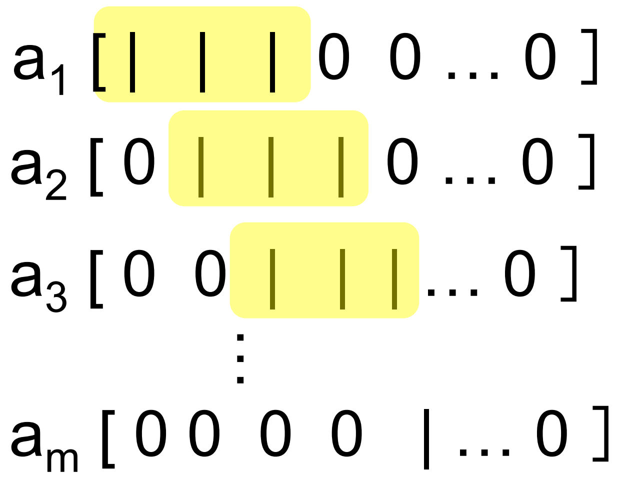
$$\begin{array}{c}
[\ 0\ |\ 0\ |\ 0\ \ 0\ ] \\
[\ 0\ |\ 0\ |\ 0\ \ 0\ ] \\
[\ 0\ |\ 0\ |\ 0\ \ 0\ ] \\
\vdots \\
[\ 0\ |\ 0\ |\ 0\ \ 0\ ]
\end{array}$$

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)





Lecture #2 Preview

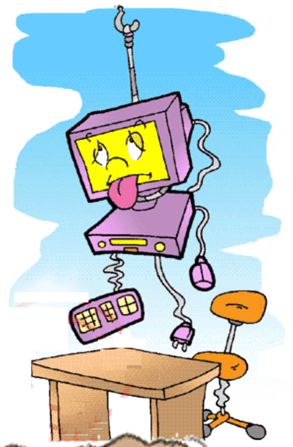
- How can deep learning help learn features?
- Sparse coding confirms DPF on the right track

- 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

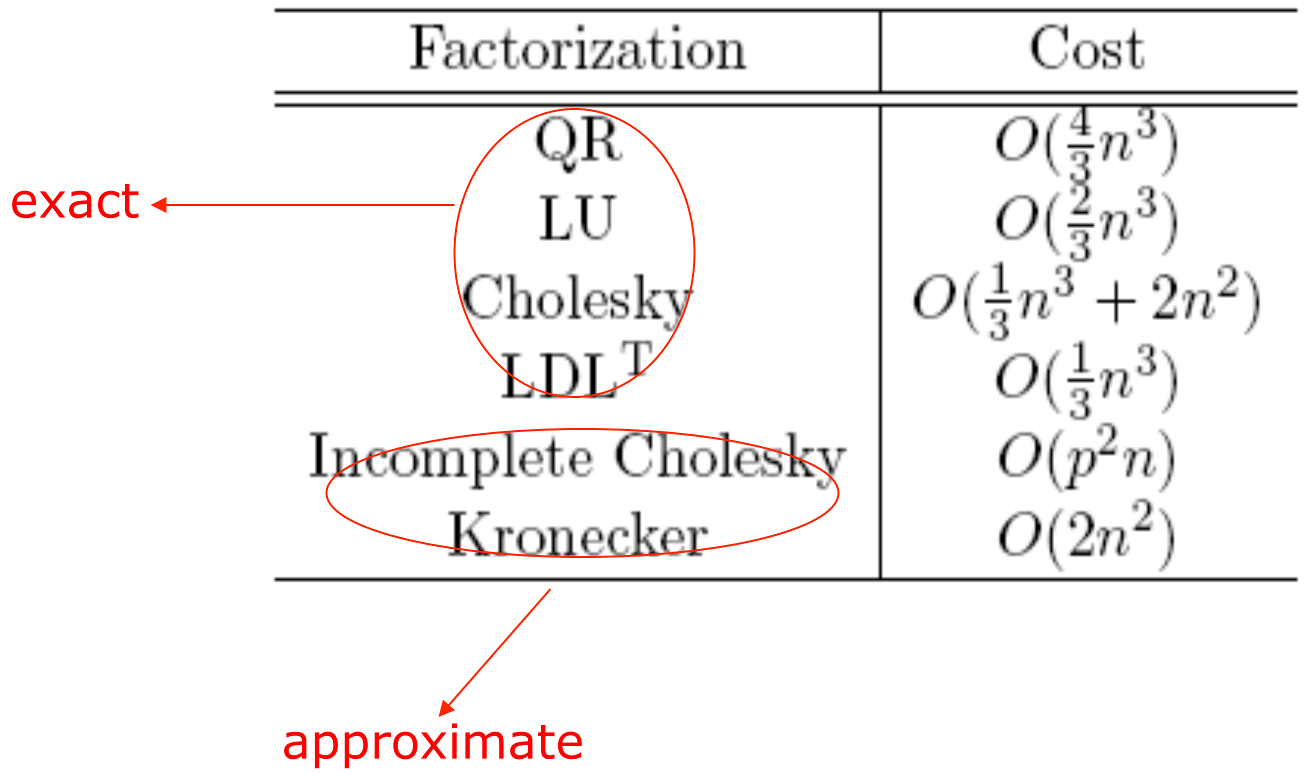
Time consuming – 1M dataset, 8 days

... .
.



Memory consuming – 1M dataset, 10G

Matrix Factorization Alternatives



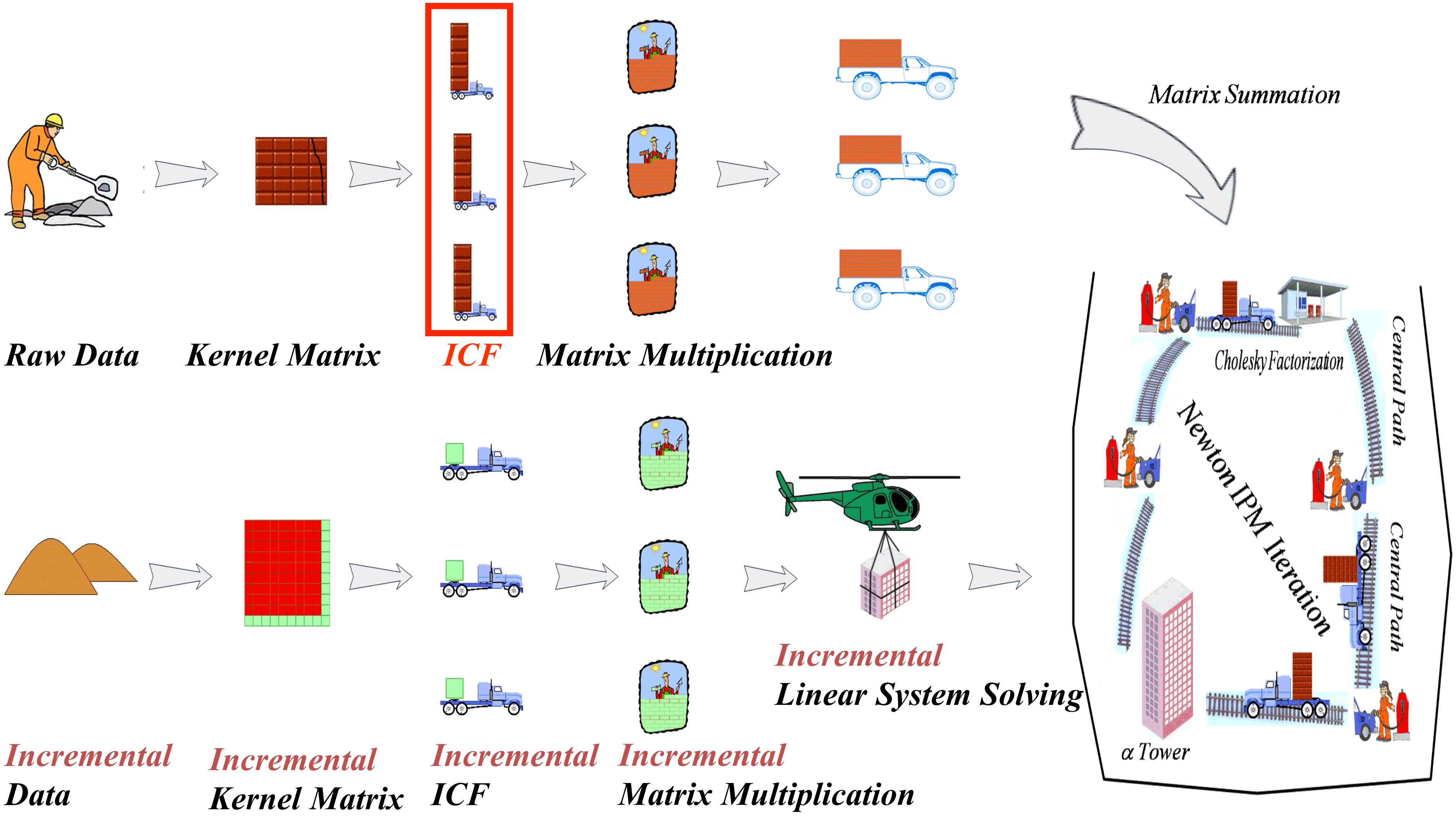
Factorization	Cost
QR	$O(\frac{4}{3}n^3)$
LU	$O(\frac{2}{3}n^3)$
Cholesky	$O(\frac{1}{3}n^3 + 2n^2)$
$\mathrm{LDL}^{\mathrm{T}}$	$O(\frac{1}{3}n^3)$
Incomplete Cholesky	$O(p^2n)$
Kronecker	$O(2n^2)$

exact
approximate

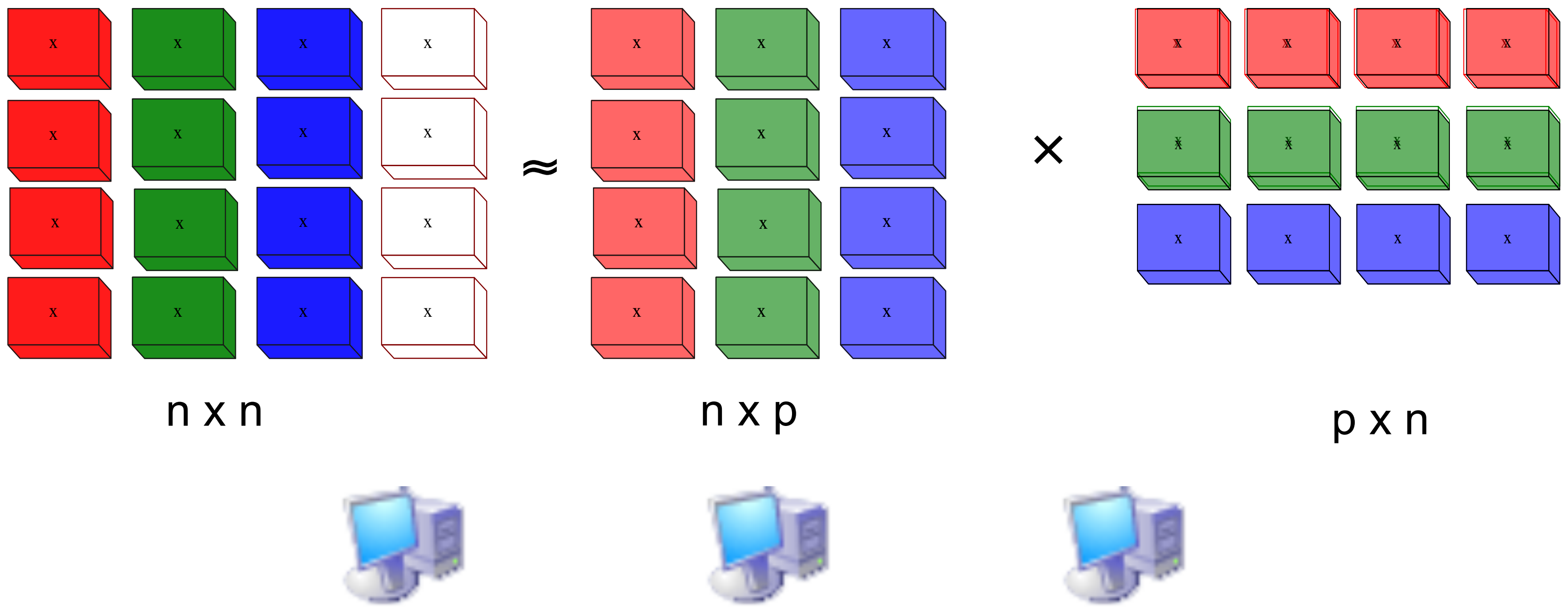
PSVM [E. Chang, et al, NIPS 07]

- 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 x B) x C
  - D = A x (B x C)

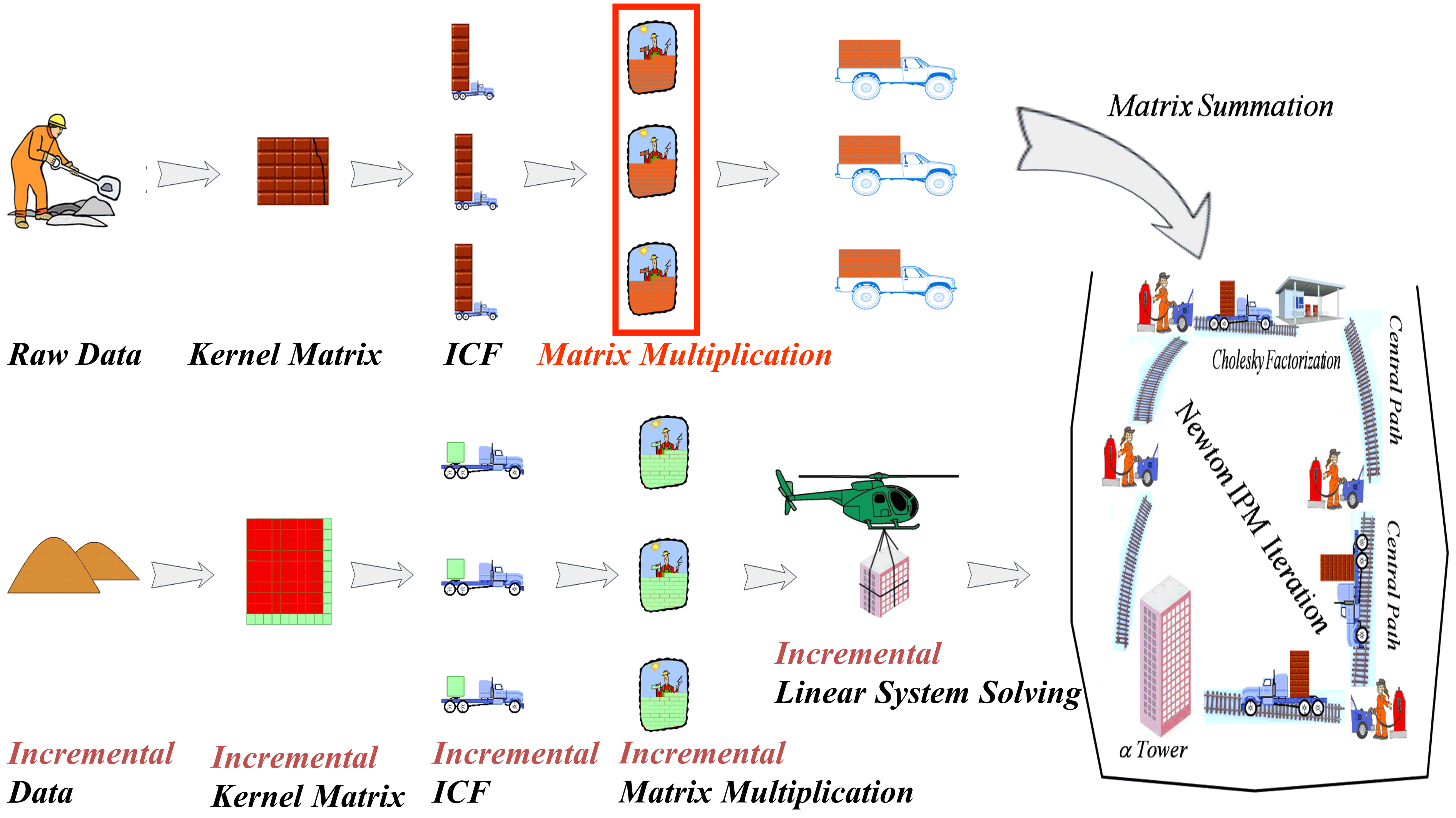
Parallelized and Incremental SVM



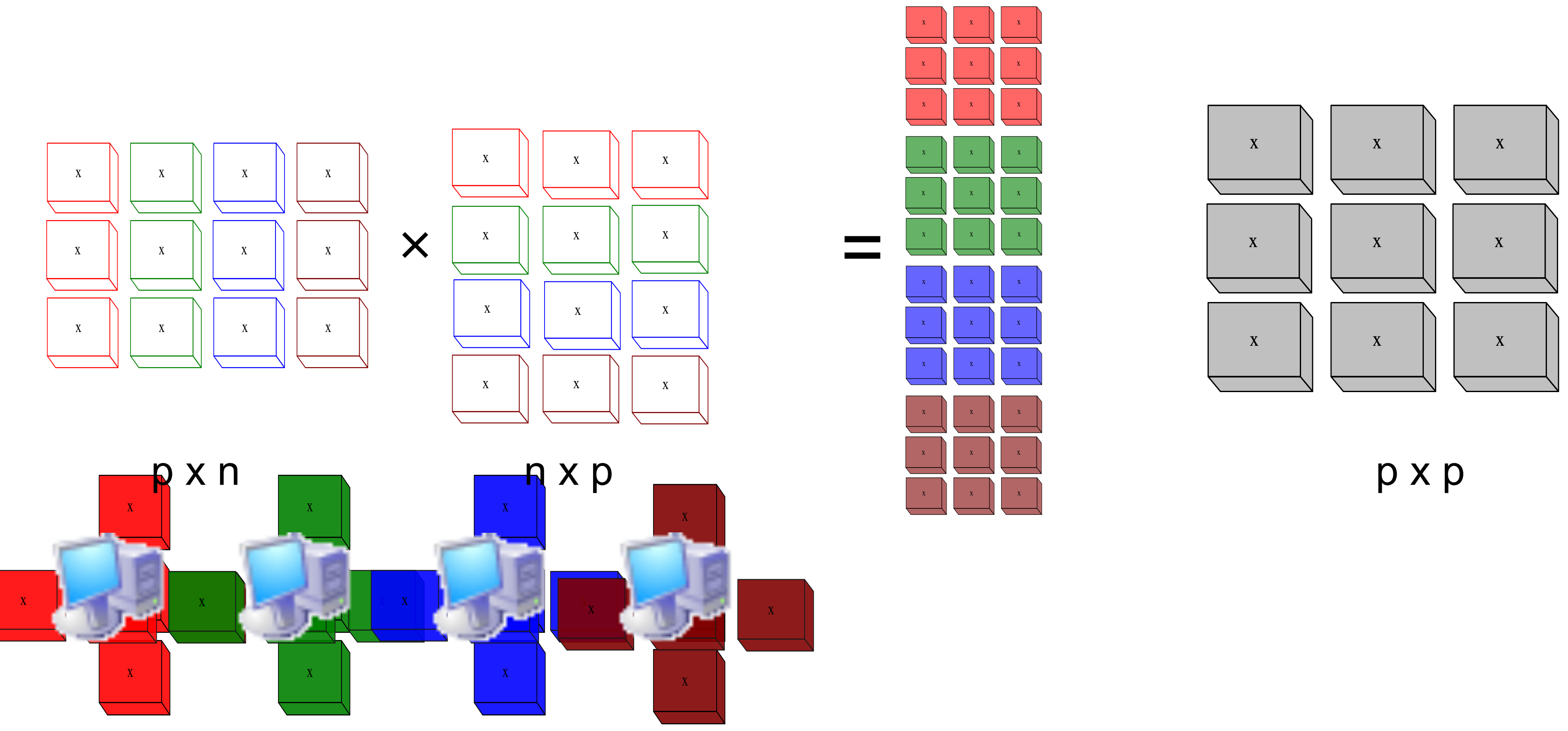
Incomplete Cholesky Factorization (ICF)



Parallelized and Incremental SVM



Matrix Product



Speedup

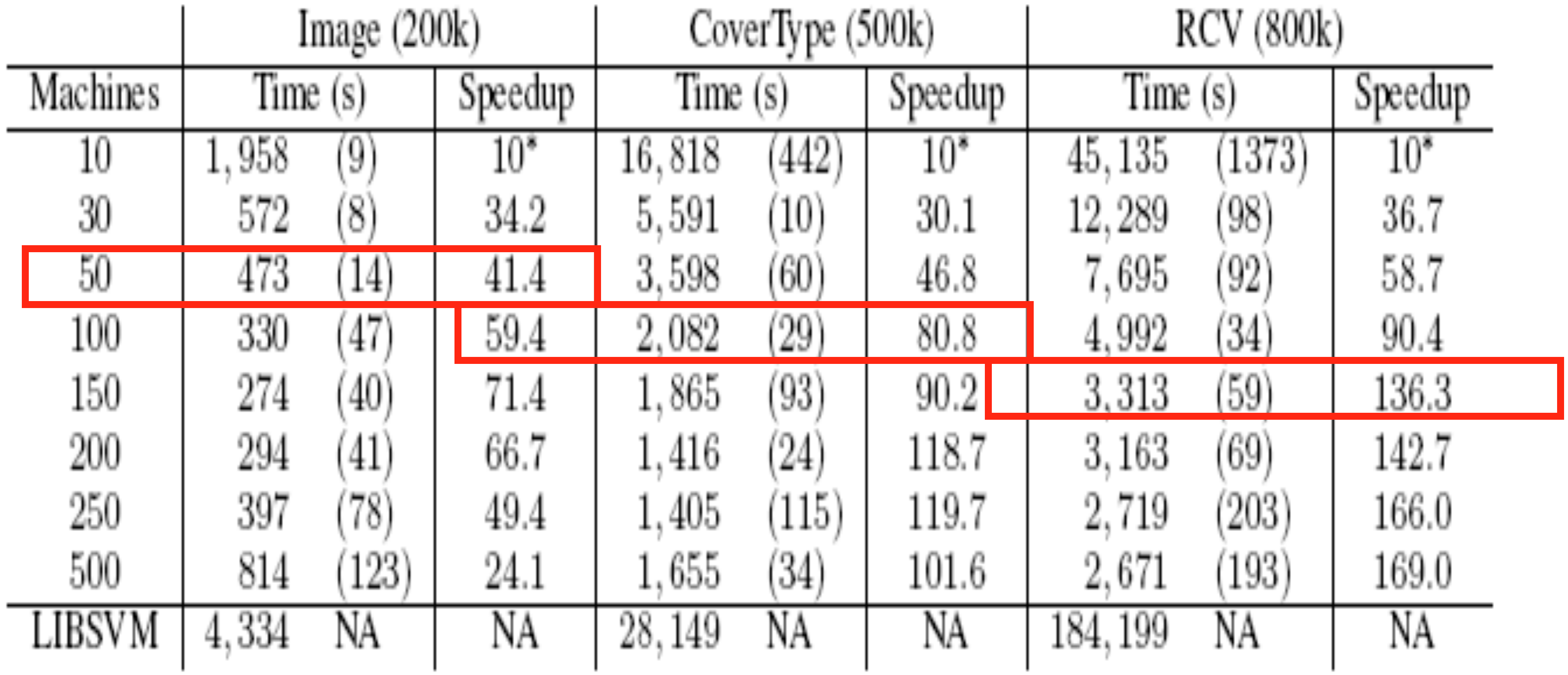
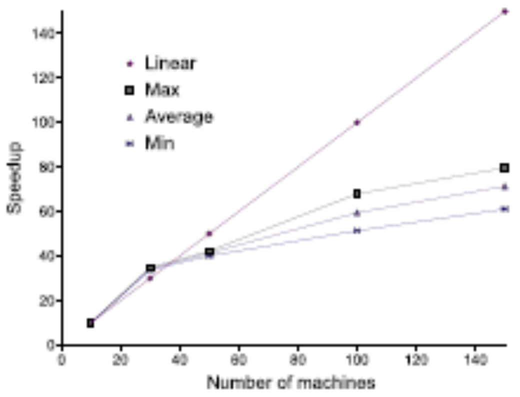
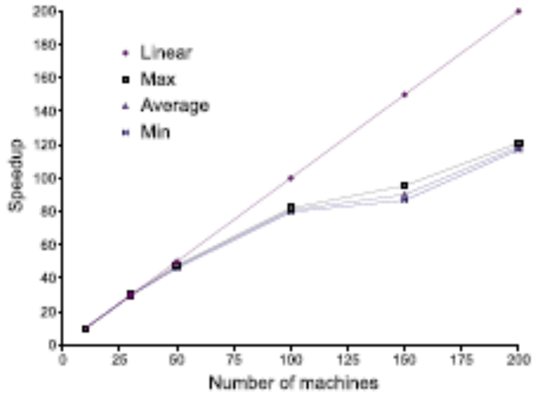
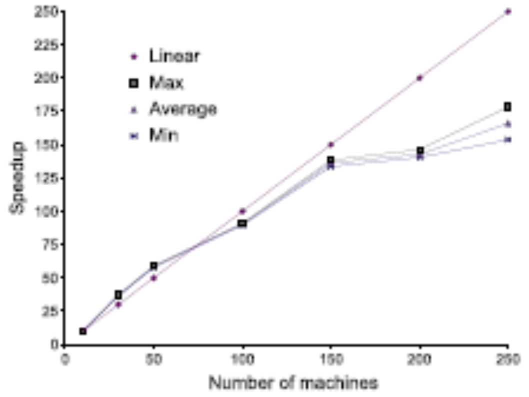


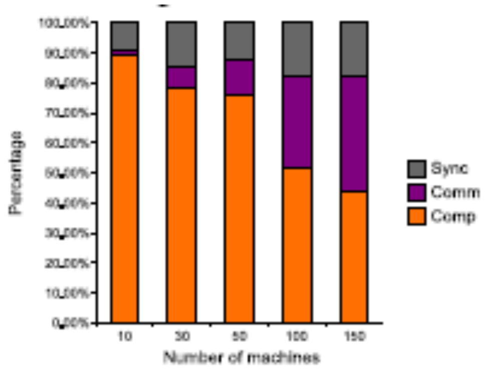
	Image (200k)			CoverType (500k)			RCV (800k)		
Machines	Time (s)		Speedup	Time (s)		Speedup	Time (s)		Speedup
10	1,958	(9)	10*	16,818	(442)	10*	45,135	(1373)	10*
30	572	(8)	34.2	5,591	(10)	30.1	12,289	(98)	36.7
50	473	(14)	41.4	3,598	(60)	46.8	7,695	(92)	58.7
100	330	(47)	59.4	2,082	(29)	80.8	4,992	(34)	90.4
150	274	(40)	71.4	1,865	(93)	90.2	3,313	(59)	136.3
200	294	(41)	66.7	1,416	(24)	118.7	3,163	(69)	142.7
250	397	(78)	49.4	1,405	(115)	119.7	2,719	(203)	166.0
500	814	(123)	24.1	1,655	(34)	101.6	2,671	(193)	169.0
LIBSVM	4,334	NA	NA	28,149	NA	NA	184,199	NA	NA

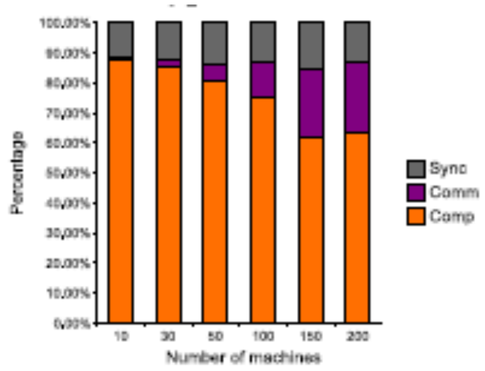
Overheads

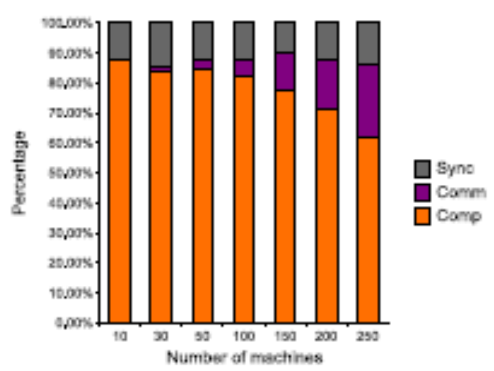

Linear
Max
Average
Min
Speedup
Number of machines


Linear
Max
Average
Min
Speedup
Number of machines


Linear
Max
Average
Min
Speedup
Number of machines


Sync
Comm
Comp
Percentage
Number of machines


Sync
Comm
Comp
Percentage
Number of machines


Sync
Comm
Comp
Percentage
Number of machines

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

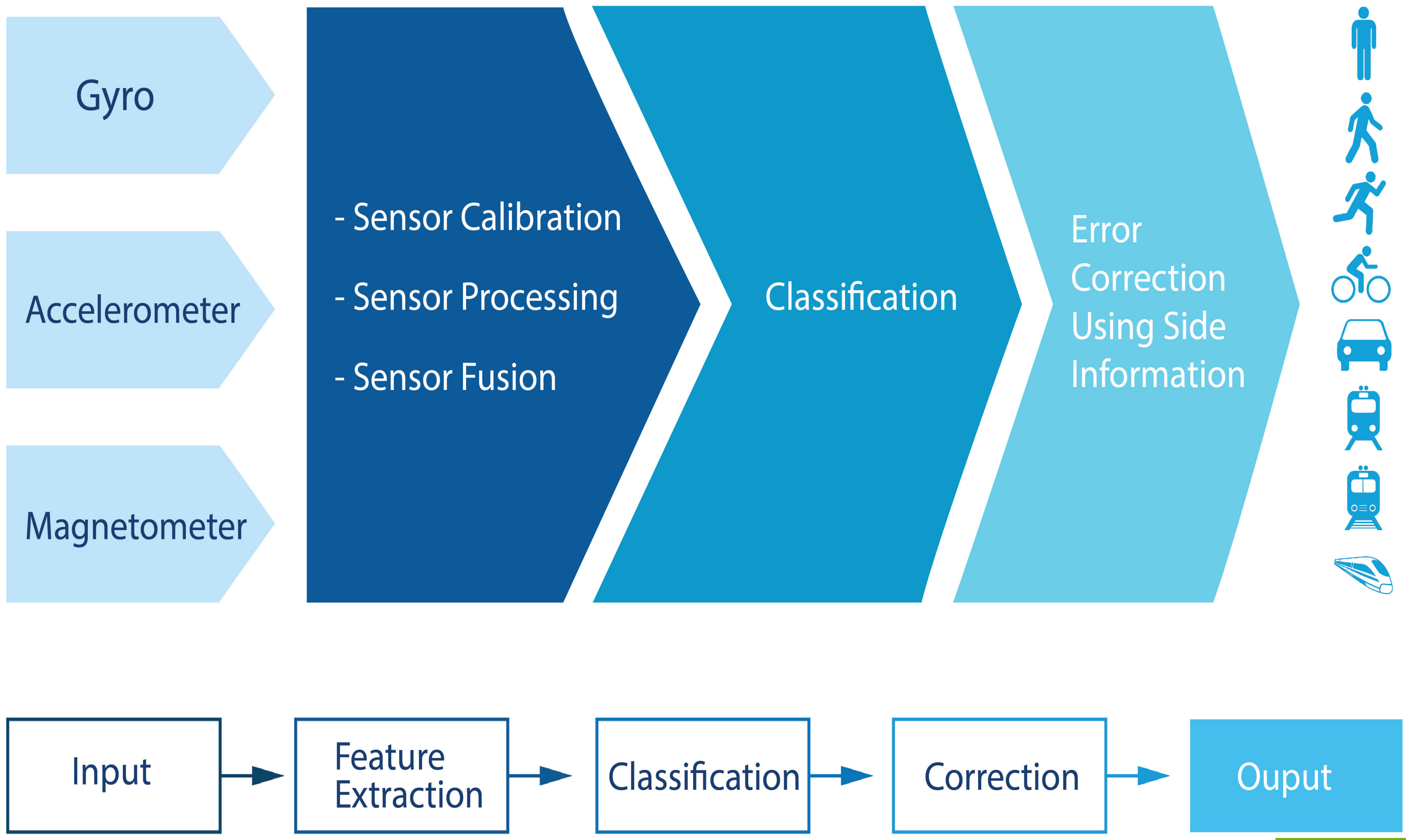
[Chang, et al. VLDB 2013, 2014]

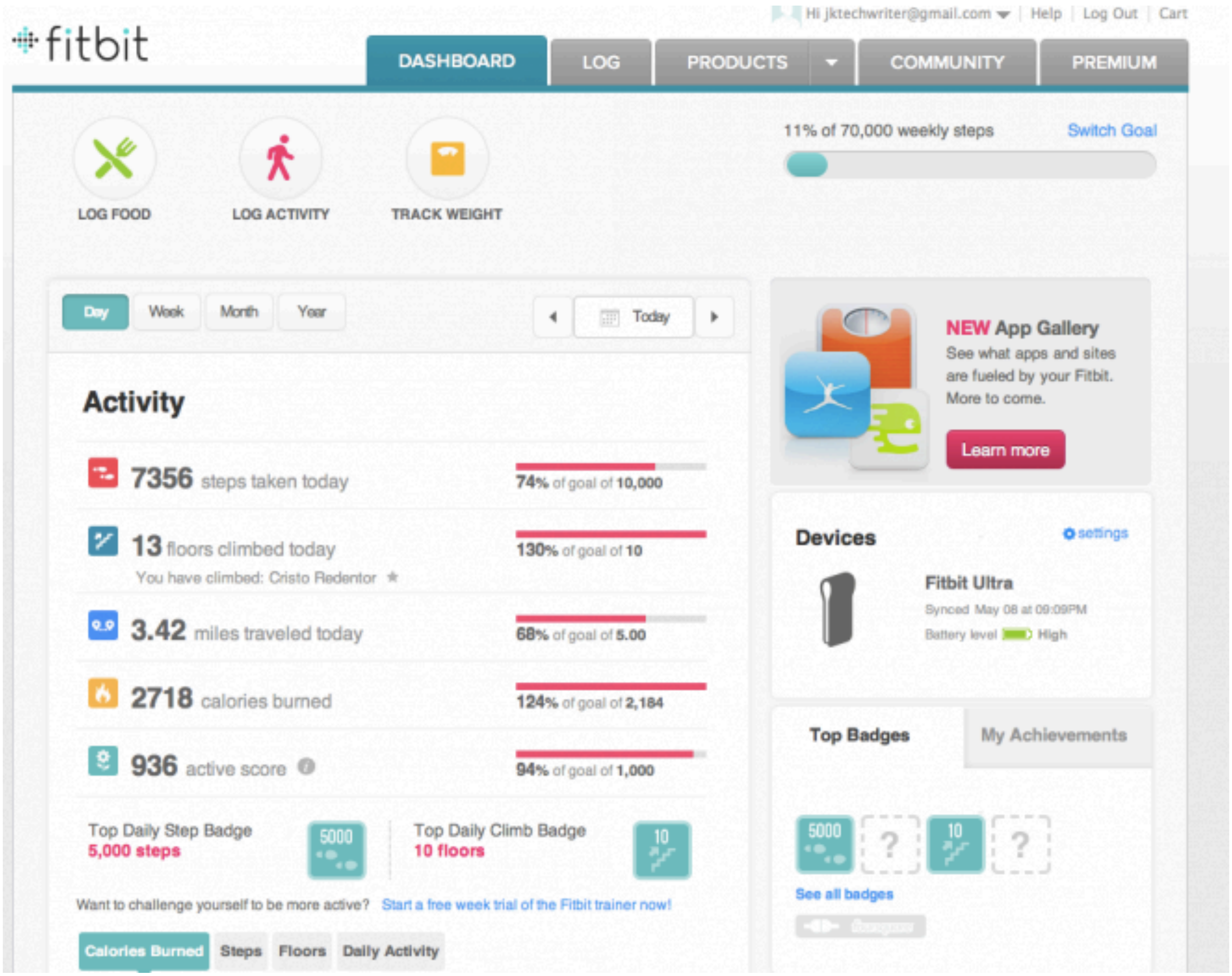
1/26/2015

Transportation-mode Detection

Transportation Mode Detection

[Chang, et al., VLDB 2013, 2014]




Hi jktechwriter@gmail.com | Help | Log Out | Cart

fitbit

DASHBOARD | LOG | PRODUCTS | COMMUNITY | PREMIUM

LOG FOOD | LOG ACTIVITY | TRACK WEIGHT

11% of 70,000 weekly steps — Switch Goal

Day | Week | Month | Year | Today

Activity

7356 steps taken today	74% of goal of 10,000
13 floors climbed today You have climbed: Cristo Redentor ★	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!

Calories Burned | Steps | Floors | Daily Activity

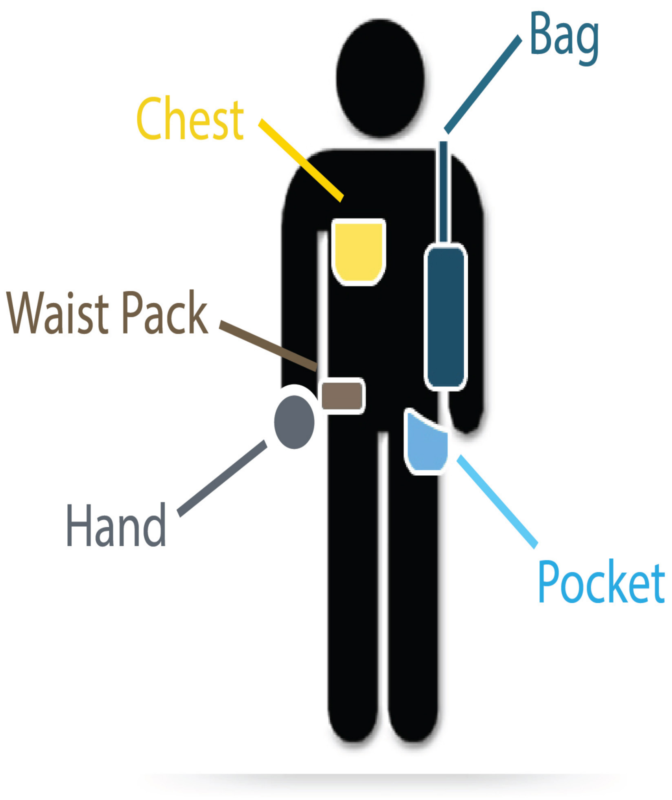
NEW App Gallery
See what apps and sites are fueled by your Fitbit. More to come.
Learn more

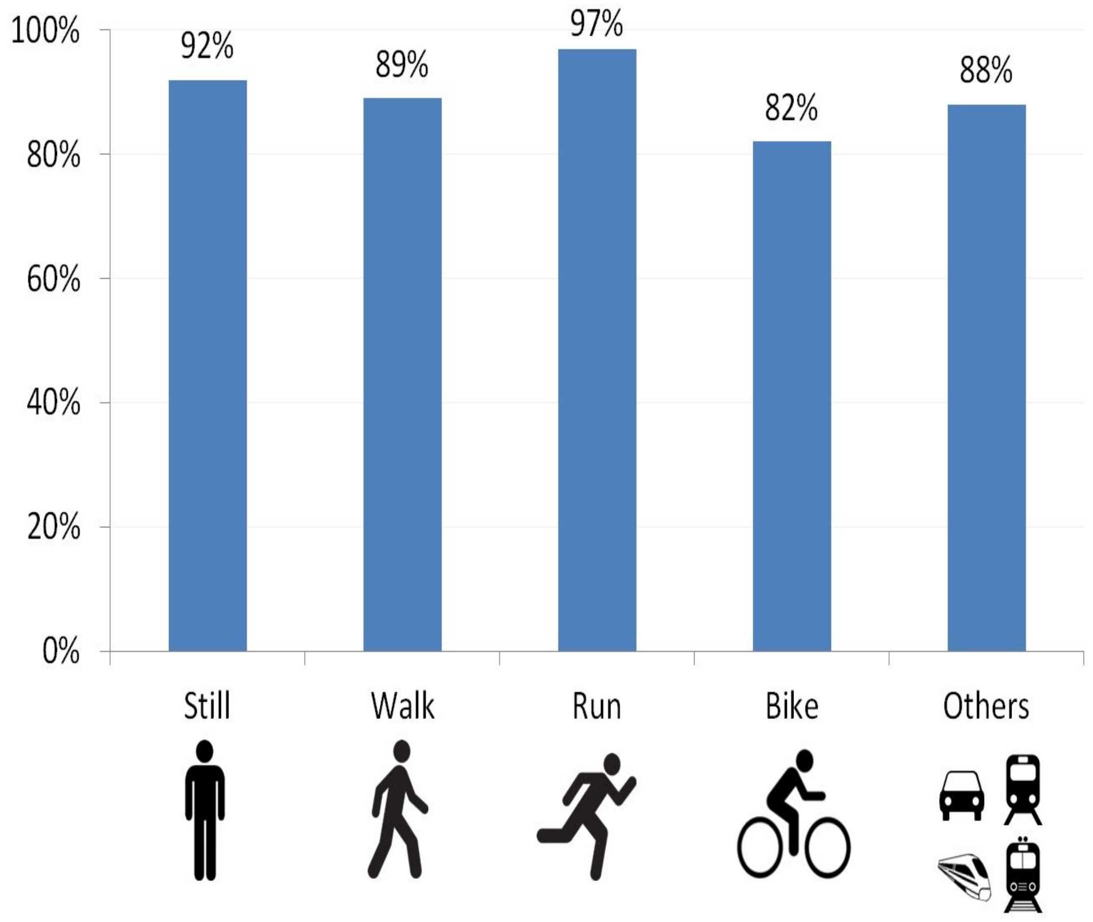
Devices — settings
Fitbit Ultra
Synced May 08 at 09:09PM
Battery level High

Top Badges | My Achievements

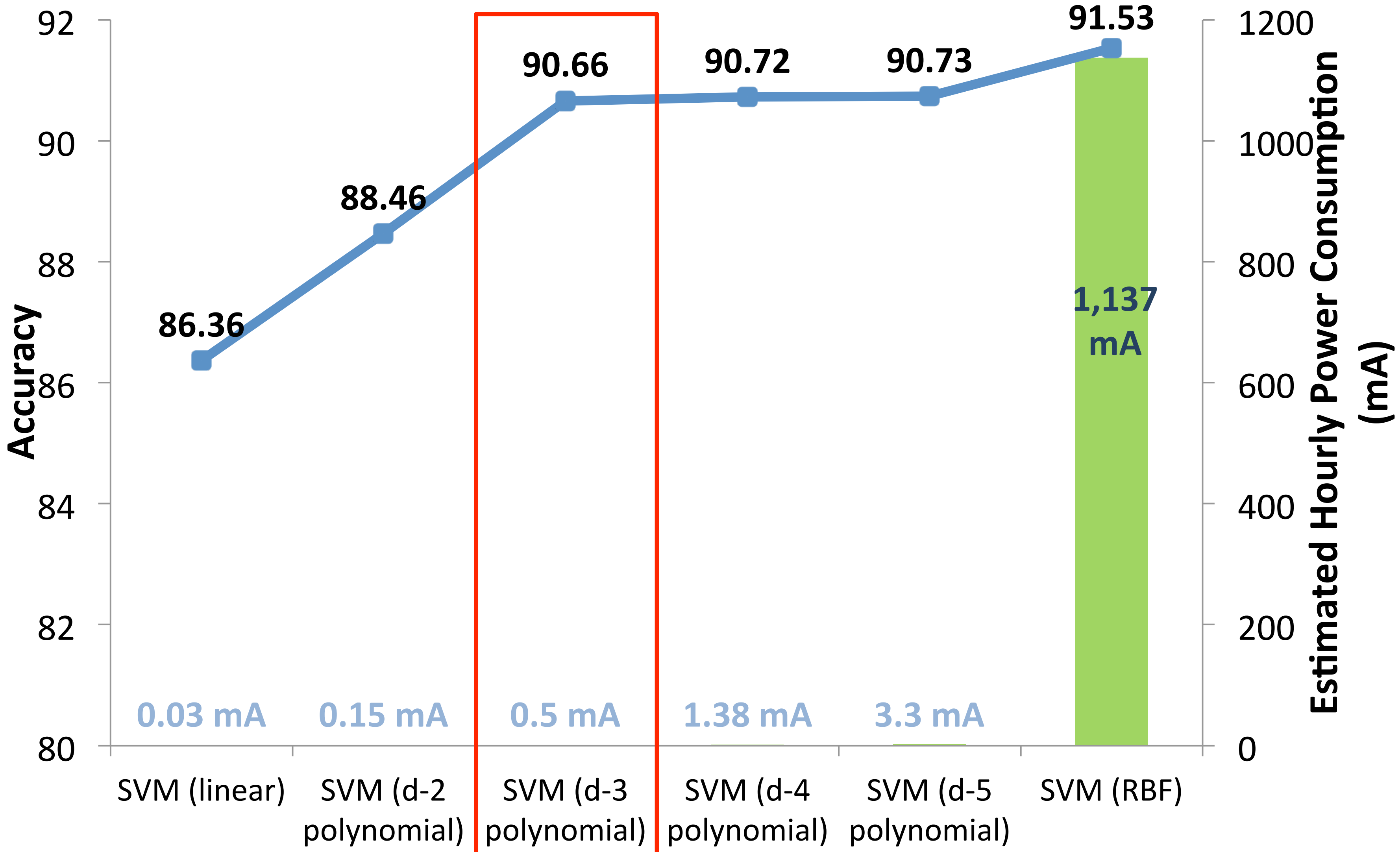
See all badges

Data Driven Classification





Sensor Hub Saving 1/2



SVMs → Max Margin M

- Min $|w|^2/2$
  - subject to $y_i(x_i w+b) \geq 1$
  - $i = 1,\dots,N$
- $Lp = \min_{w,b} |w|^2/2 + \sum_{i=1..N} \alpha_i[y_i(x_i w+b)-1]$
- $w = \sum_{i=1..N} \alpha_i y_i x_i$
- $0 = \sum_{i=1..N} \alpha_i y_i$

Wolfe Dual

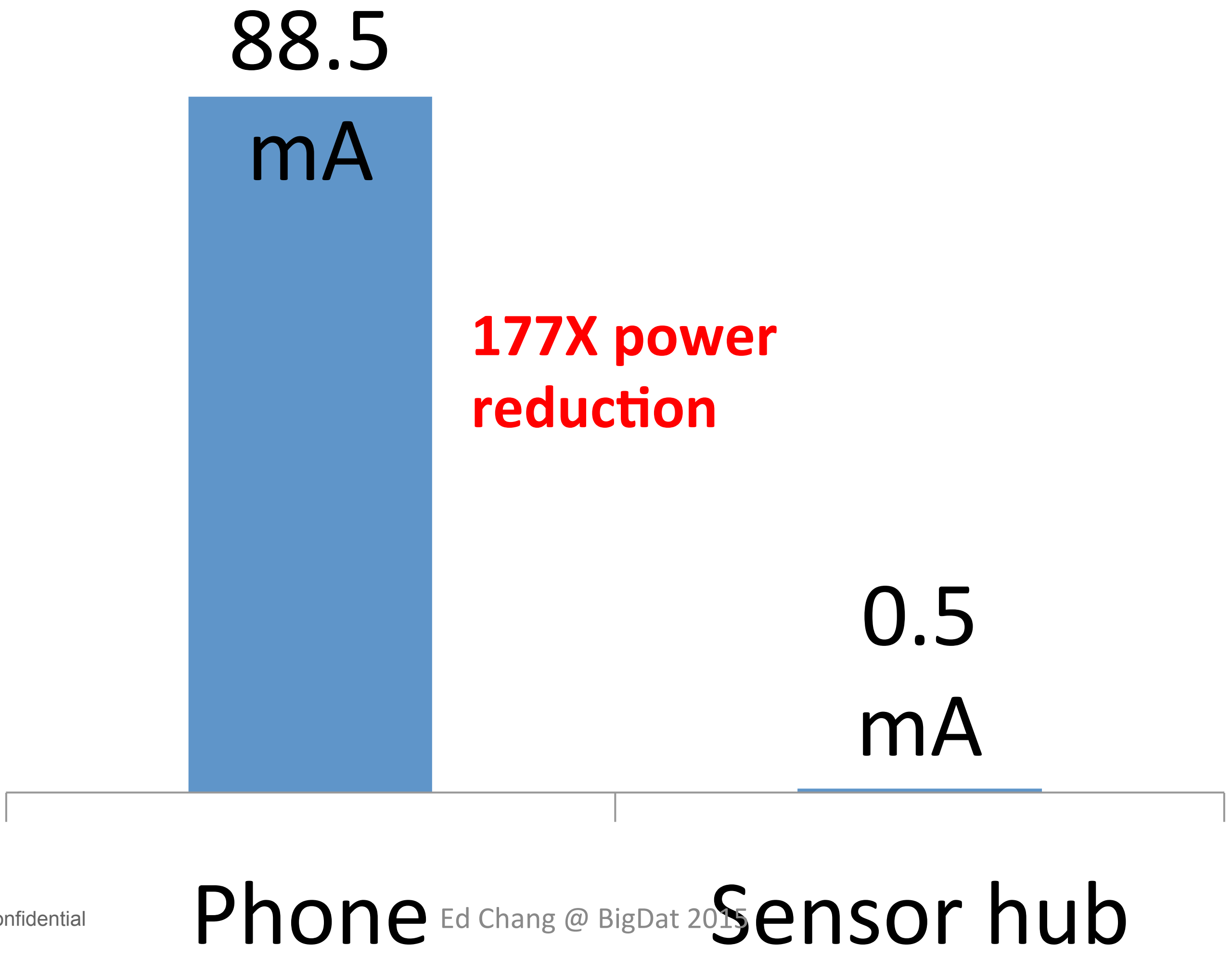
- Ld = $\sum_{i=1..N} \alpha$ - 1/2 $\sum\sum_{i,j=1..N} \alpha_i \alpha_j y_i y_j x_i x_j$
- Subject to
  - $\alpha_i \geq 0$
  - $\alpha_i [y_i(x_i w+b)-1] = 0$
  - KKT conditions
    - $\alpha_i > 0$, $y_i(x_i w+b) = 1$ (Support Vectors)
    - $\alpha_i = 0$, $y_i(x_i w+b) > 1$

Class Prediction

- $y_q = w\ x_q + b$
- $w = \sum_{i=1..N} \alpha_i y_i x_i$
- $y_q = \mathrm{sign}(\sum_{i=1..N} \alpha_i y_i (x_i . x_q) + b)$

Sensor Hub Saving 2/2

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



Applications & Algorithms

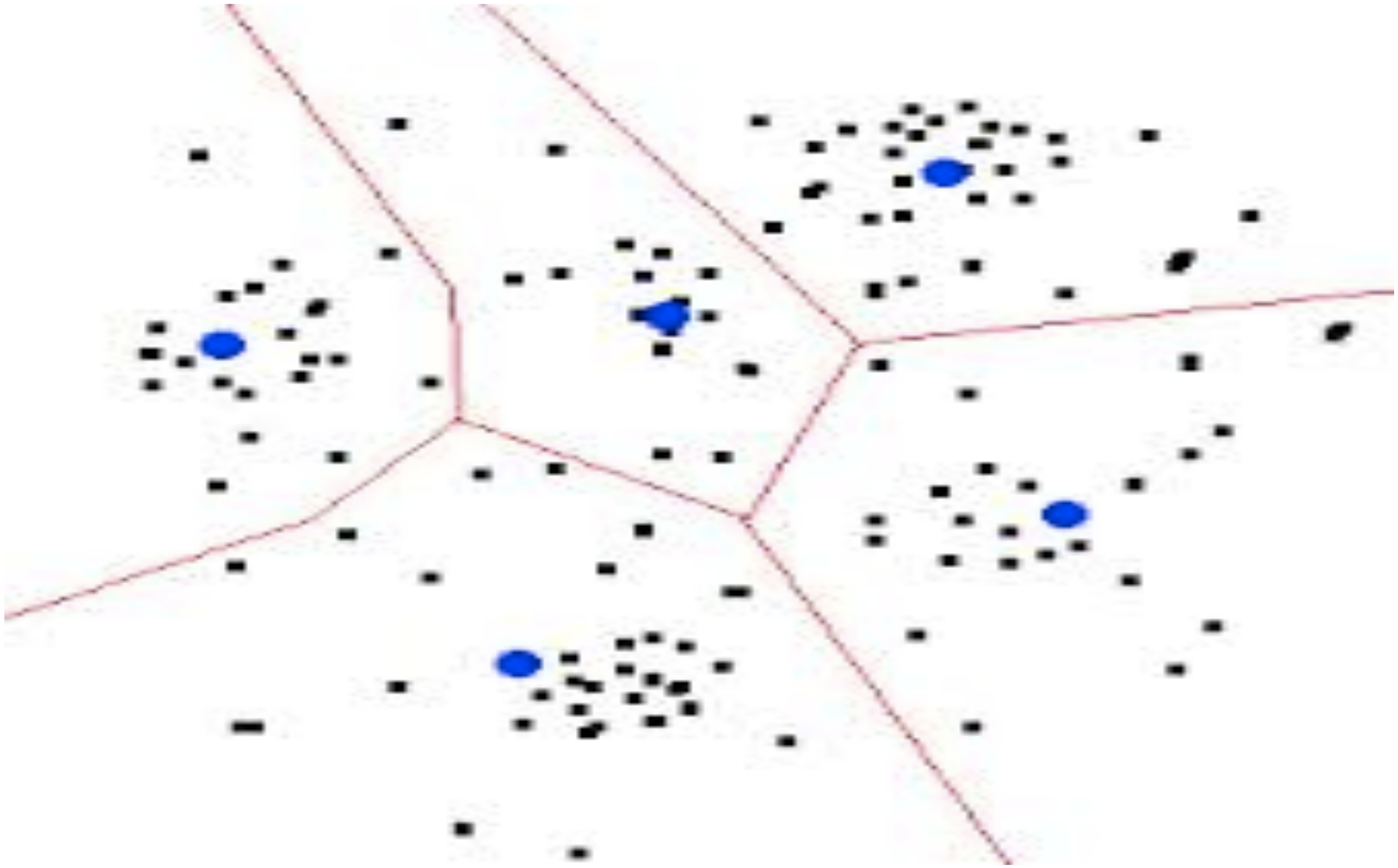
- Applications
  - HTC XPRICE Tricorder
  - Context-aware Computing
- Key Algorithms
  - Frequent Itemset Mining [ACM RS 08]
  - Latent Dirichlet Allocation [WWW 09, TIST 10]
  - Support Vector Machines [MM 01, MS 03, NIPS 07, VLDB 14]
  - Spectral Clustering [ECML 08, PAMI 10]
  - Deep Learning [NIPS 12, OSDI 14]
- Perspectives and Opportunities

Clustering

Most Widely Used Pattern Recognition Subroutine

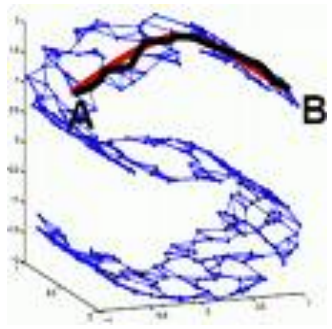
- Microarray Data Analysis
- Ultrasound Image Segmentation
- Document Pattern Discovery
- High-dimensional Data Indexing

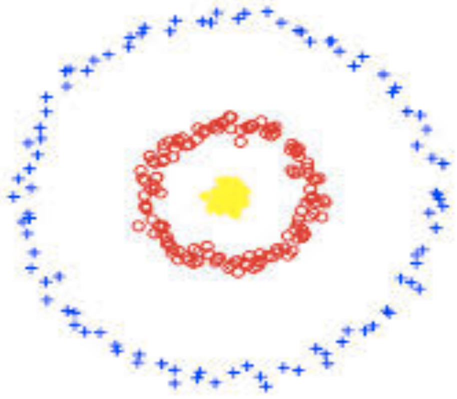
K Means



Spectral Clustering [A. Ng, M. Jordan]

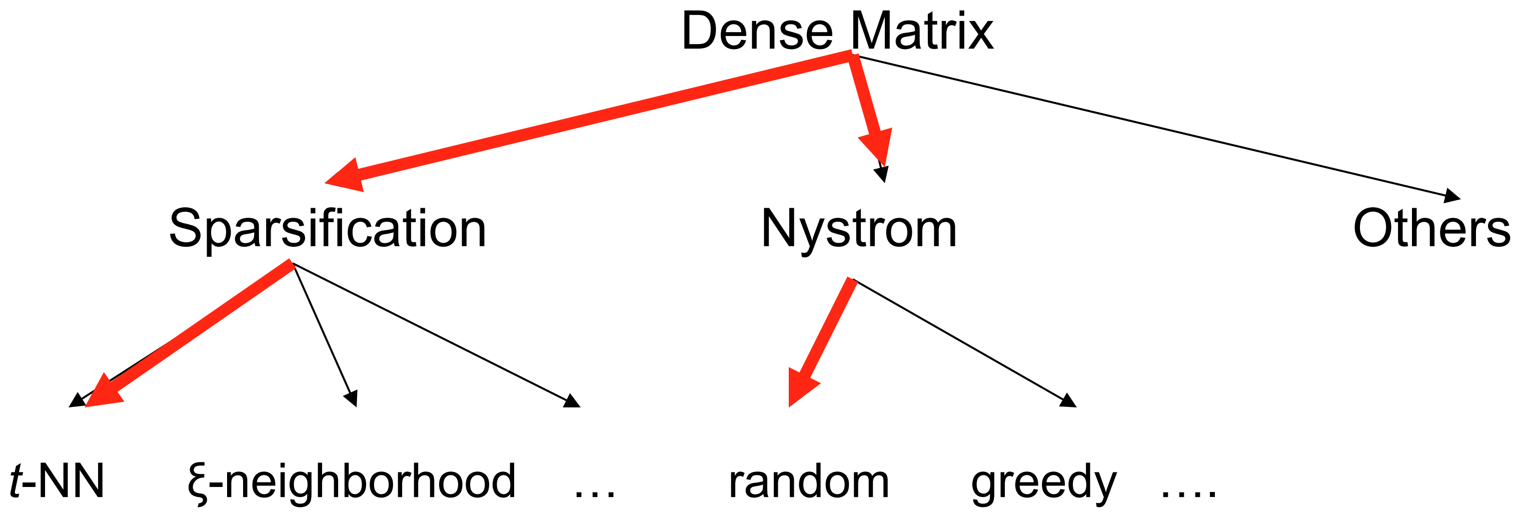
- Exploit *pairwise similarity* of data instances
- Key steps
  - Construct pairwise similarity matrix
    - e.g., using Geodisc distance
  - Compute the Laplacian matrix
  - Apply eigendecomposition
  - Perform *k*-means





Scalability Problem

- Quadratic computation of *n*x*n* matrix
- Approximation methods



Sparsification vs. Sampling

- Construct the dense similarity matrix S
- Sparsify S
- Compute Laplacian matrix L

$$L = I - D^{-1/2} S D^{-1/2}, \quad D_{ii} = \sum_{j=1}^{n} S_{ij}$$

- Apply *ARPACLK* on L
- Use *k*-means to cluster rows of V into *k* groups

- Randomly sample *l* points, where *l* << *n*
- Construct dense similarity matrix [A B] between *l* and *n* points
- Normalize A and B to be in Laplacian form

$R = A + A^{-1/2} B B^T A^{-1/2}$;

$R = U \Sigma U^T$

- *k*-means

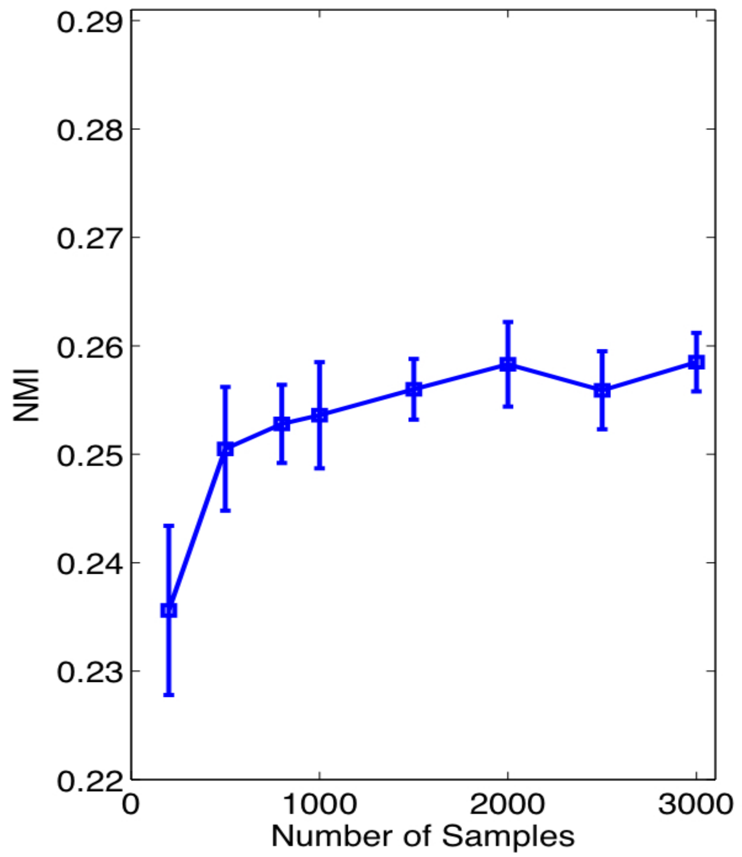
Empirical Study [song, et al., ecml 08]

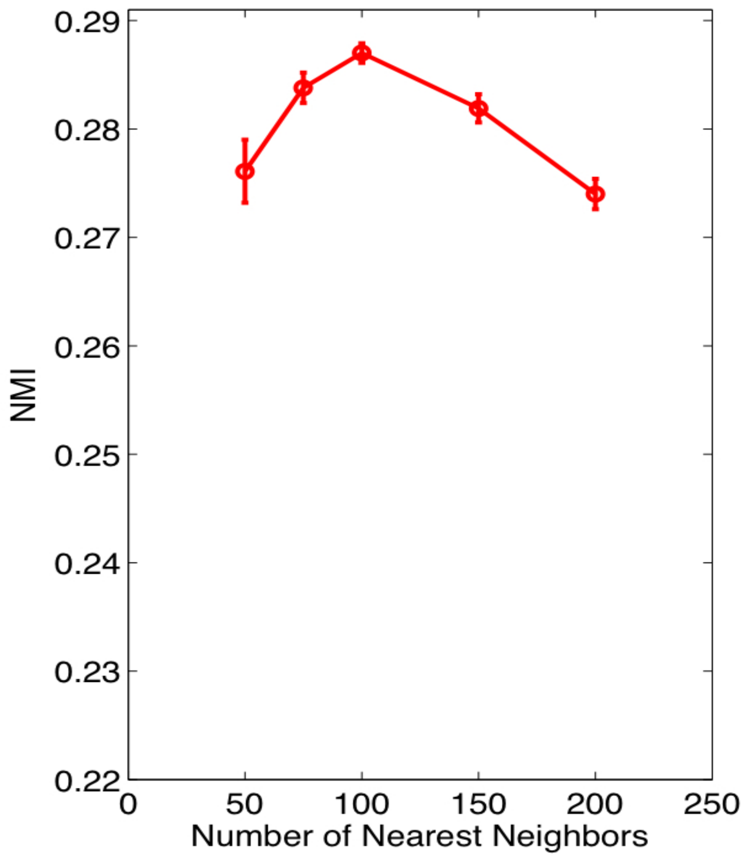
- Dataset: RCV1 (Reuters Corpus Volume I)
  - A filtered collection of *193,944* documents in *103* categories
- Photo set: PicasaWeb
  - *637,137* photos
- Experiments
  - Clustering quality vs. computational time
    - Measure the similarity between CAT and CLS
    - Normalized Mutual Information (NMI)

$$NMI(CAT;CLS) = \frac{I(CAT;CLS)}{\sqrt{H(CAT)H(CLS)}}$$

  - Scalability

NMI Comparison (on RCV1)



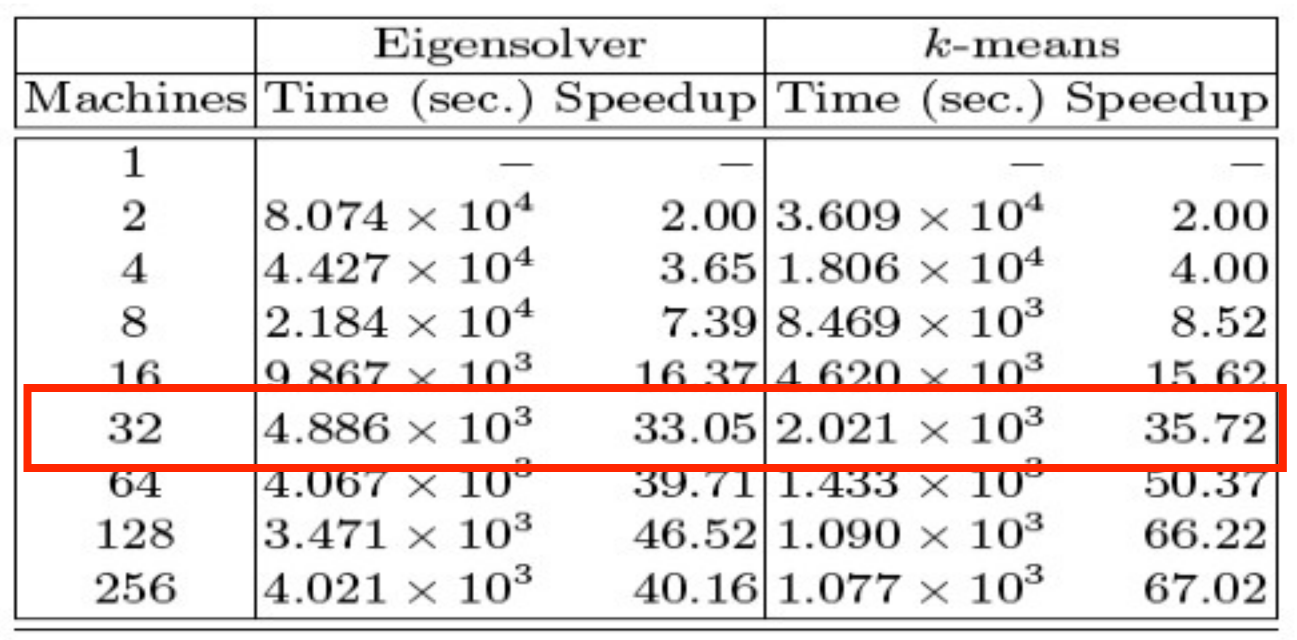


Nystrom method

Sparse matrix approximation

Speedup Test on *637,137* Photos

- K = 1000 clusters



	Eigensolver		*k*-means	
Machines	Time (sec.)	Speedup	Time (sec.)	Speedup
1	—	—	—	—
2	8.074×10^4	2.00	3.609×10^4	2.00
4	4.427×10^4	3.65	1.806×10^4	4.00
8	2.184×10^4	7.39	8.469×10^3	8.52
16	9.867×10^3	16.37	4.620×10^3	15.62
32	4.886×10^3	33.05	2.021×10^3	35.72
64	4.067×10^3	39.71	1.433×10^3	50.37
128	3.471×10^3	46.52	1.090×10^3	66.22
256	4.021×10^3	40.16	1.077×10^3	67.02

- Achiever linear speedup when using 32 machines, after that, sub-linear speedup because of increasing communication and sync time

Sparsification vs. Sampling

	Sparsification	Nystrom, random sampling
Information	Full n x n similarity scores	None
Pre-processing Complexity (bottleneck)	$O(n^2)$ worst case; easily parallizable	O(nl), l << n
Effectiveness	Good	Not bad (Jitendra M., PAMI)

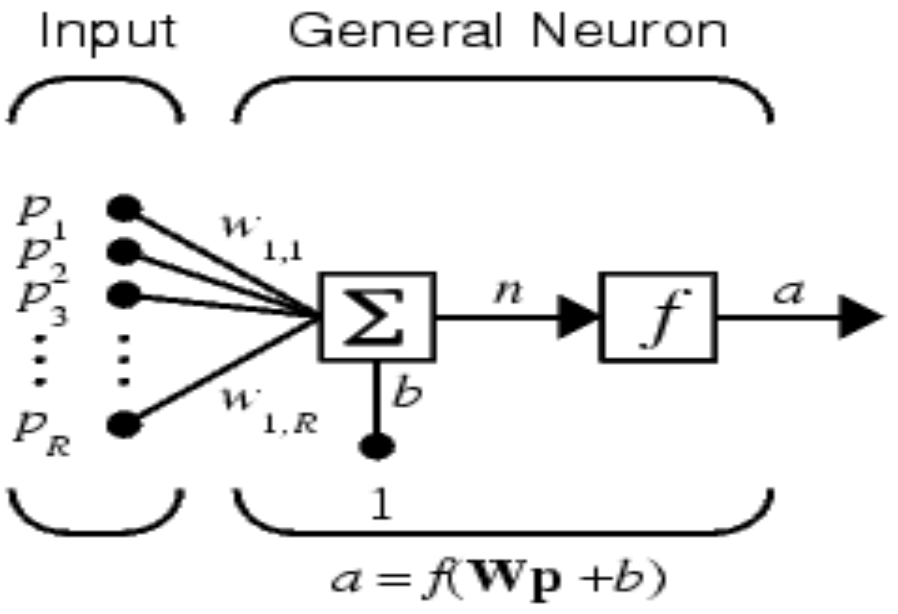
Applications & Algorithms

- 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.



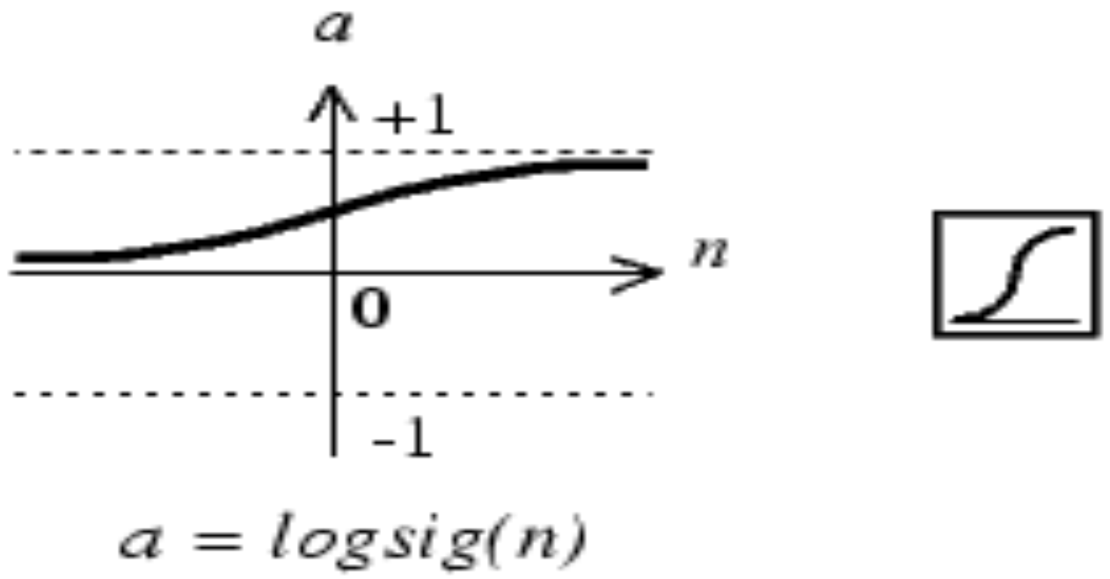
Where

R = number of elements in input vector

NN Model

Transfer Functions (Activition Function)

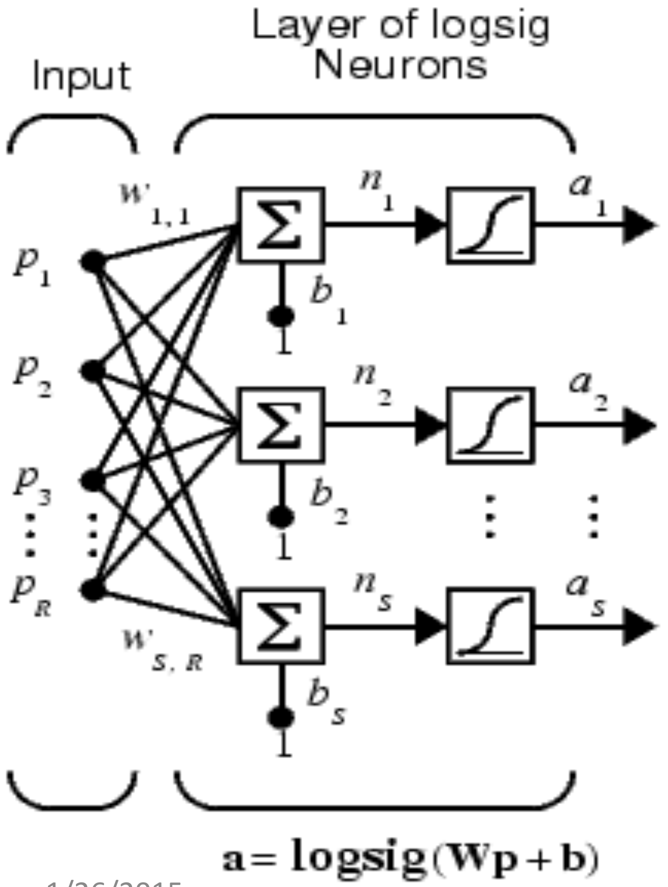
Multilayer networks often use **the log-sigmoid** transfer function **logsig**. The function logsig generates outputs between **0** and **1** as the neuron's net input goes from negative to positive infinity



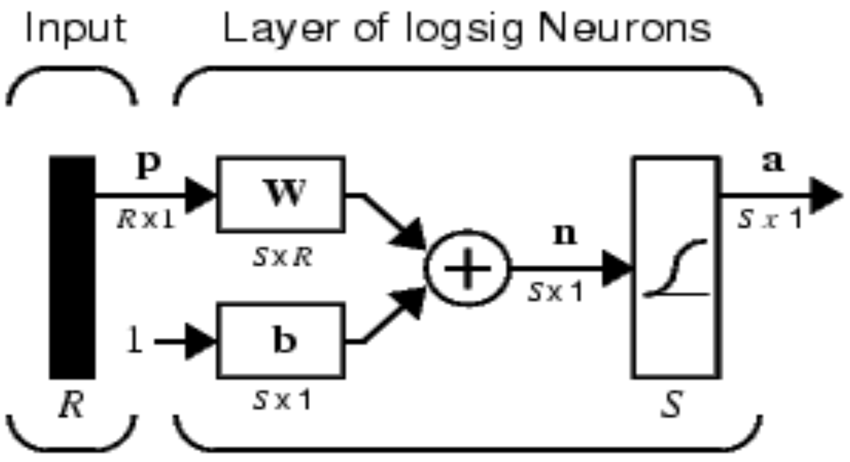
Log-Sigmoid Transfer Function

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.



$$\mathbf{a} = \mathbf{logsig}(\mathbf{Wp} + \mathbf{b})$$



$$\mathbf{a} = \mathbf{logsig}(\mathbf{Wp} + \mathbf{b})$$

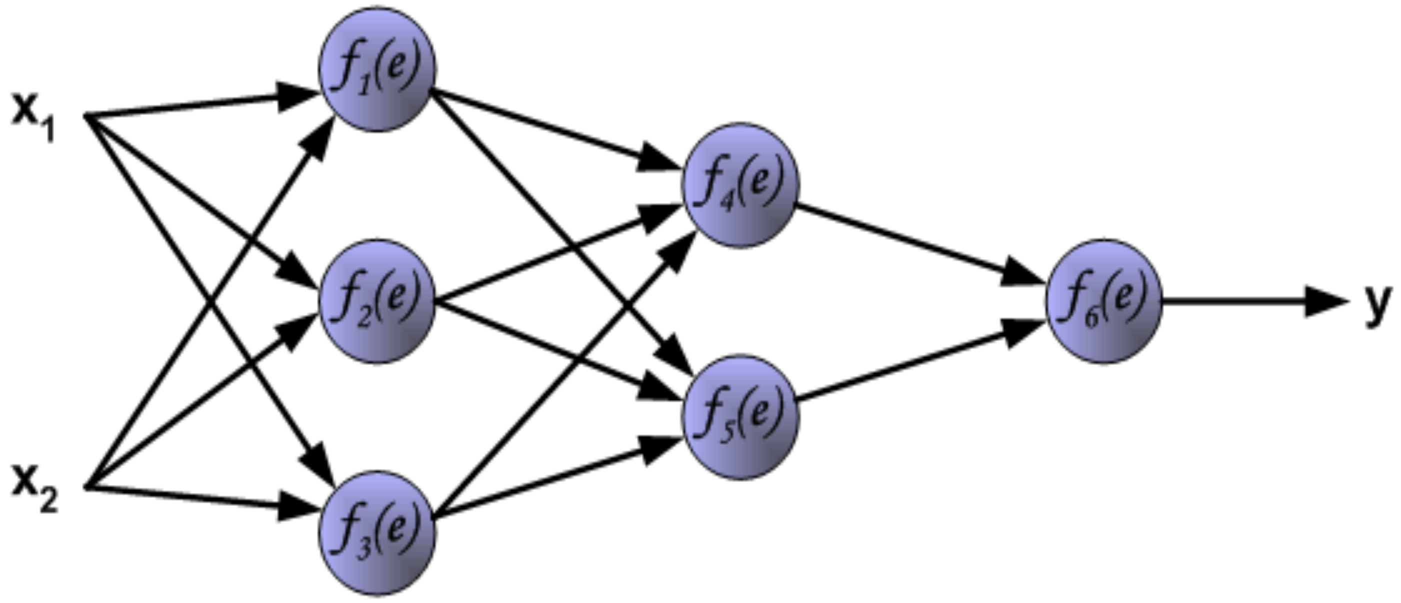
Where...

R = number of elements in input vector

S = number of neurons in layer

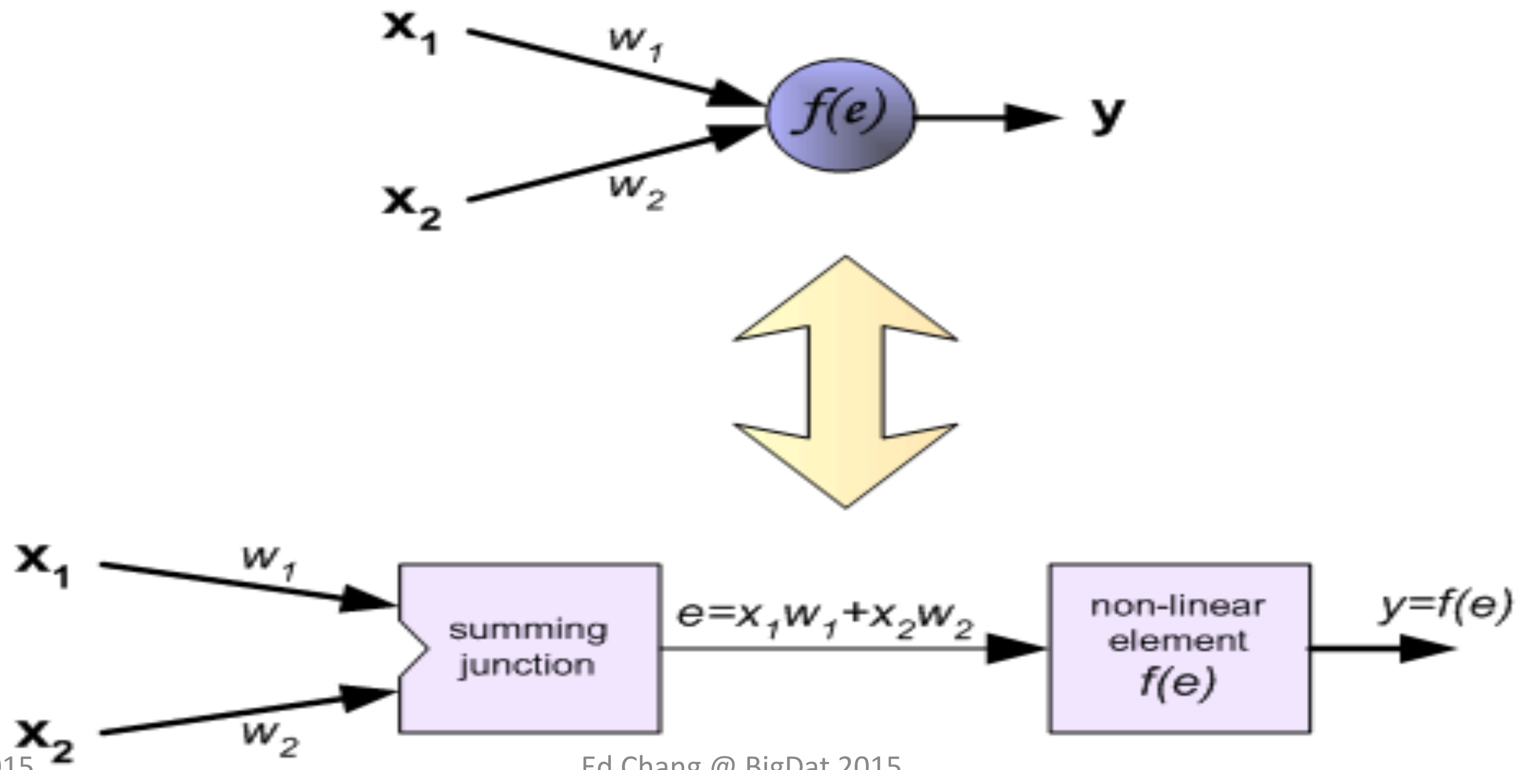
NN Model Learning Algorithm

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



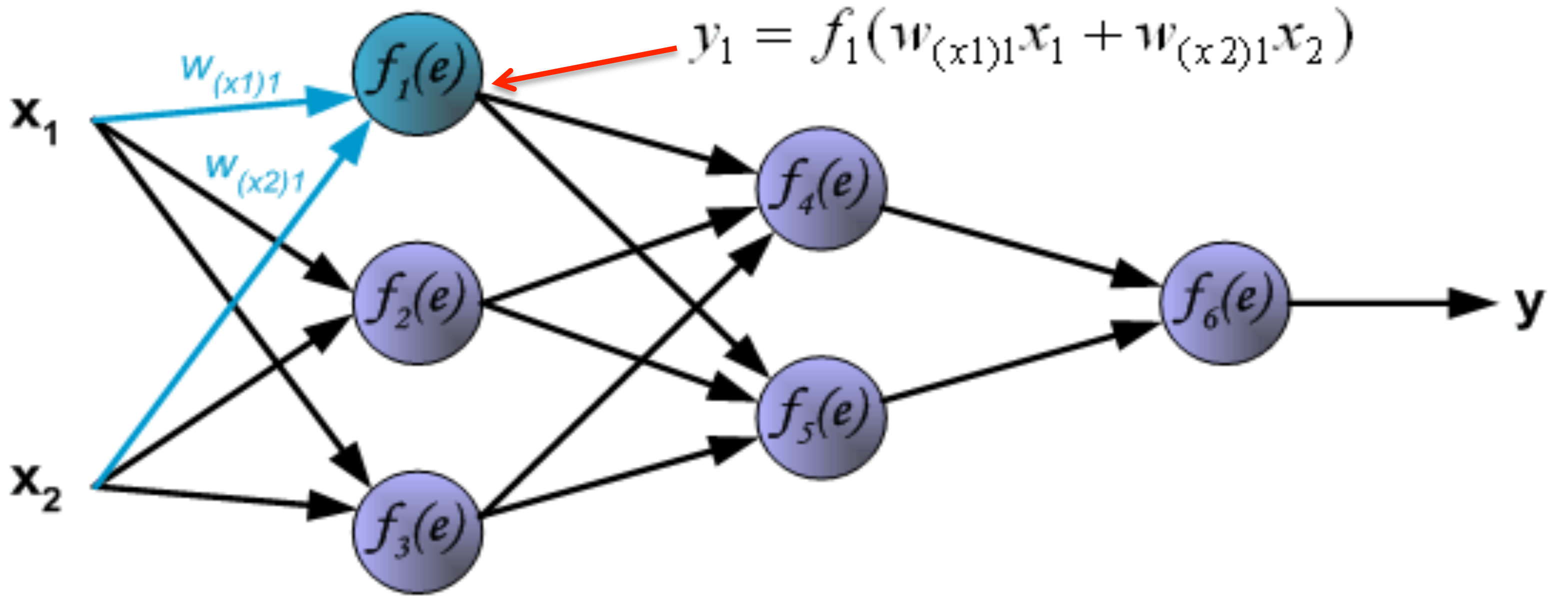
Learning Algorithm: Backpropagation

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

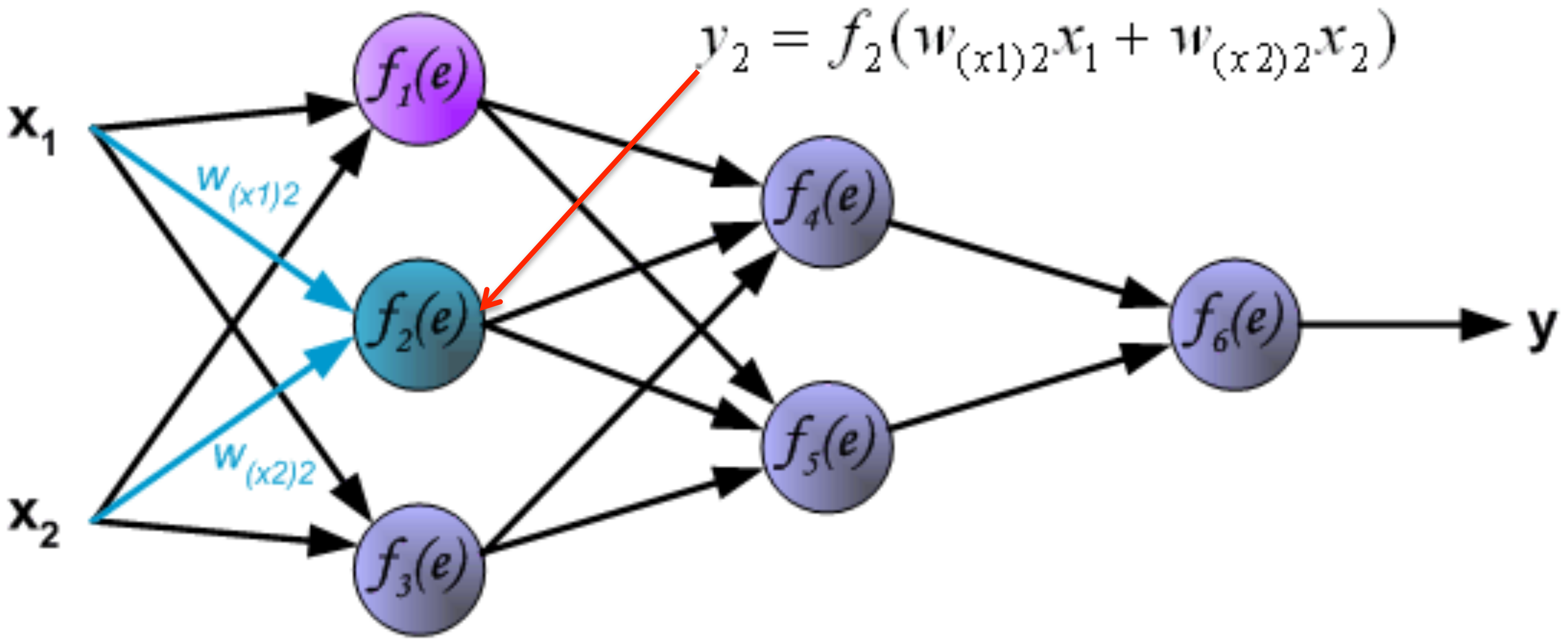


Feed Forward

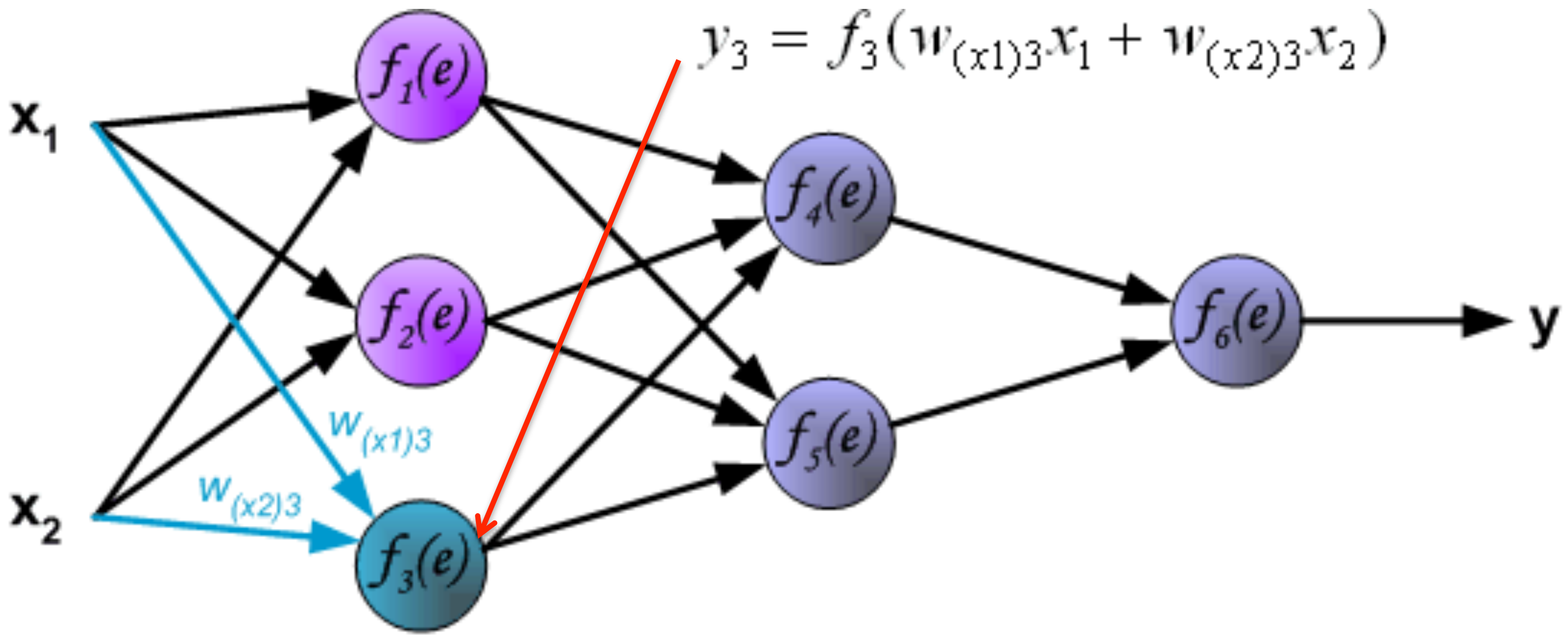
Pictures below illustrate how signal is forward-feeding through the network, Symbols $w_{(xm)n}$ represent weights of connections between network input x_m and neuron n in input layer. Symbols y_n represents output signal of neuron n.



Feed Forward



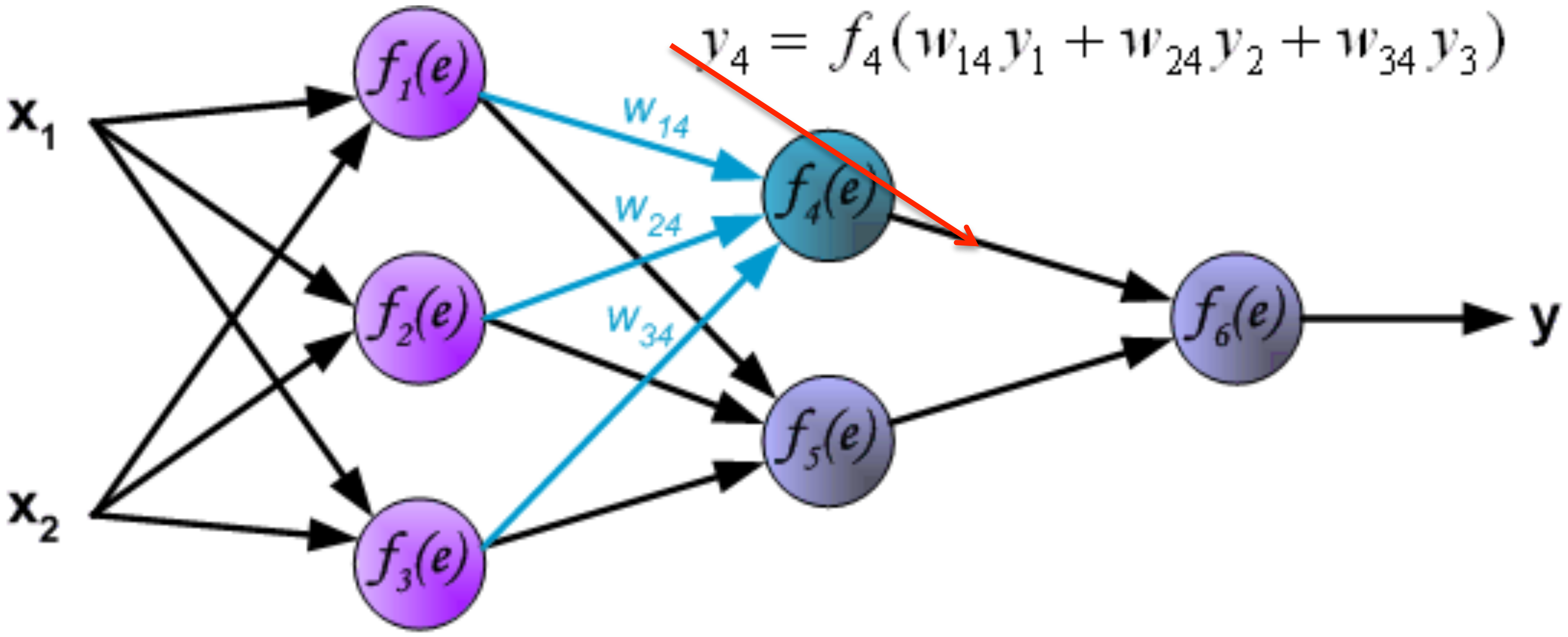
Feed Forward



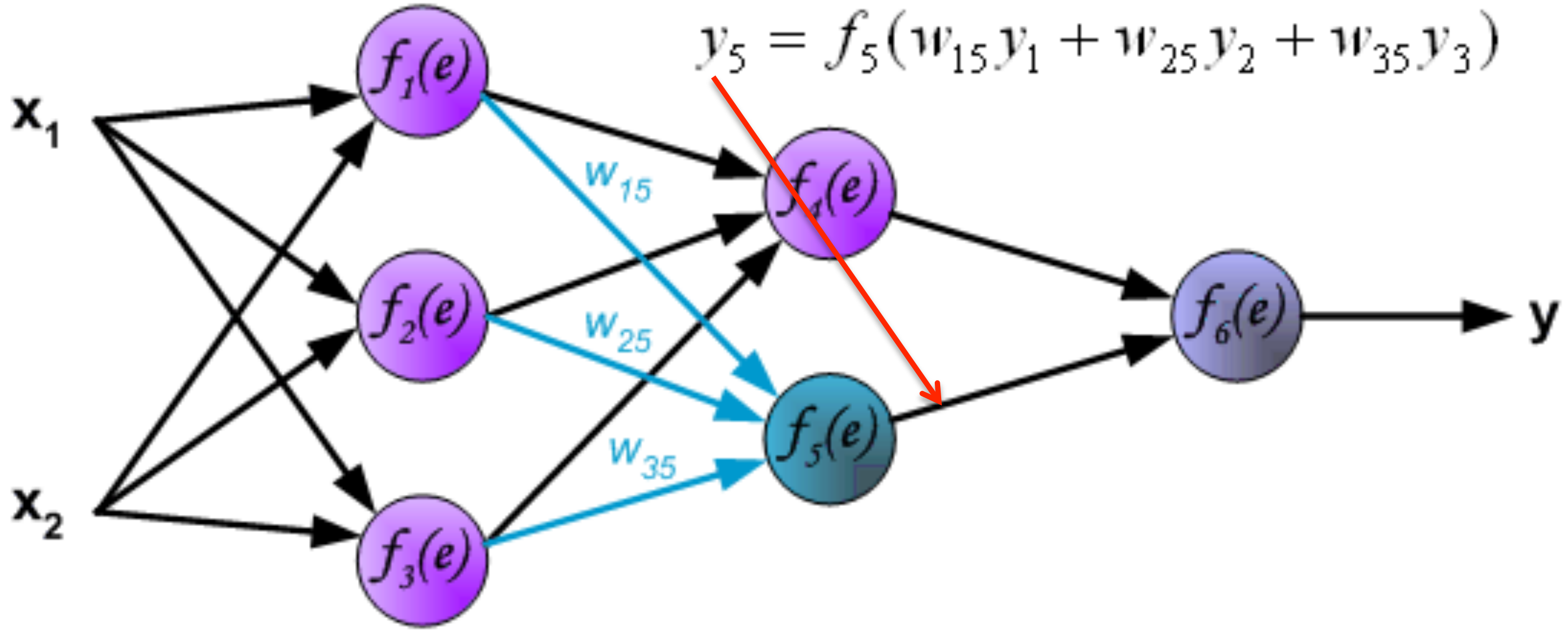
Feed Forward

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.

$$y_4 = f_4(w_{14} y_1 + w_{24} y_2 + w_{34} y_3)$$

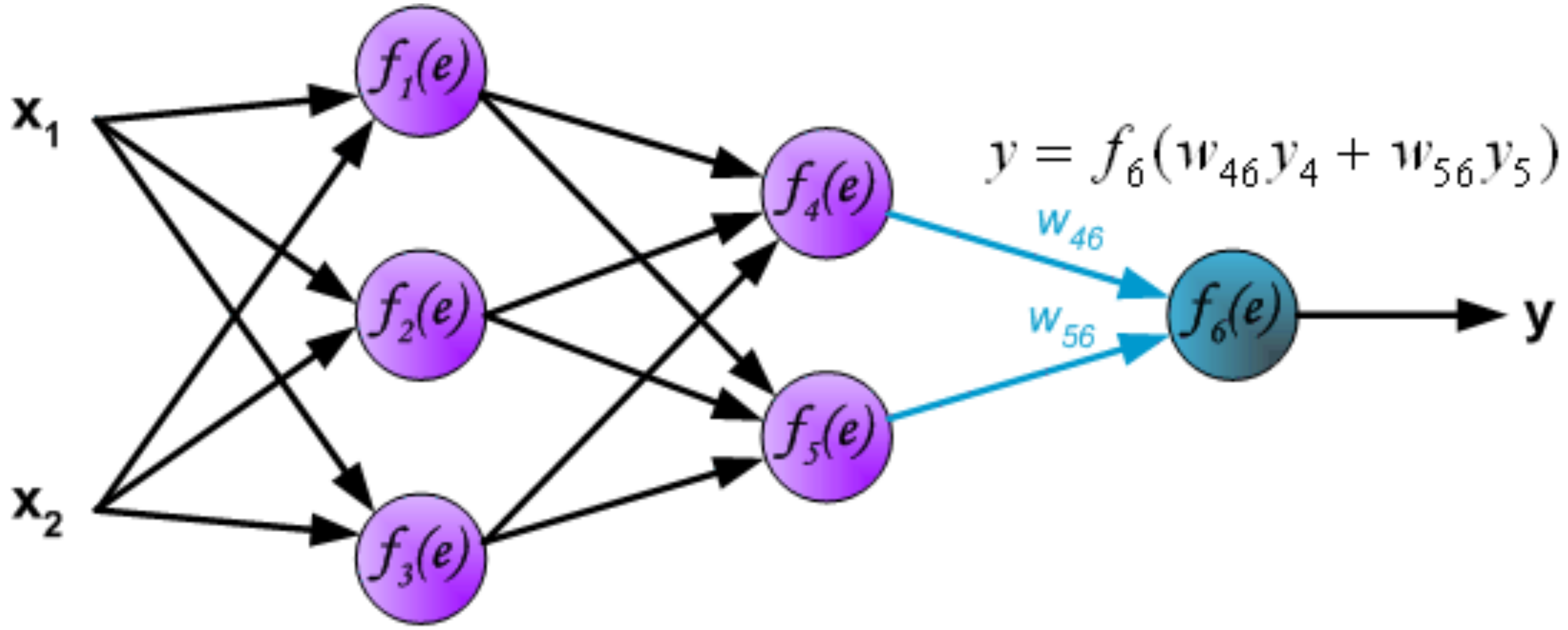


Feed Forward



Learning Algorithm: Forward Pass

Propagation of signals through the output layer.



Learning Algorithm: Backpropagation

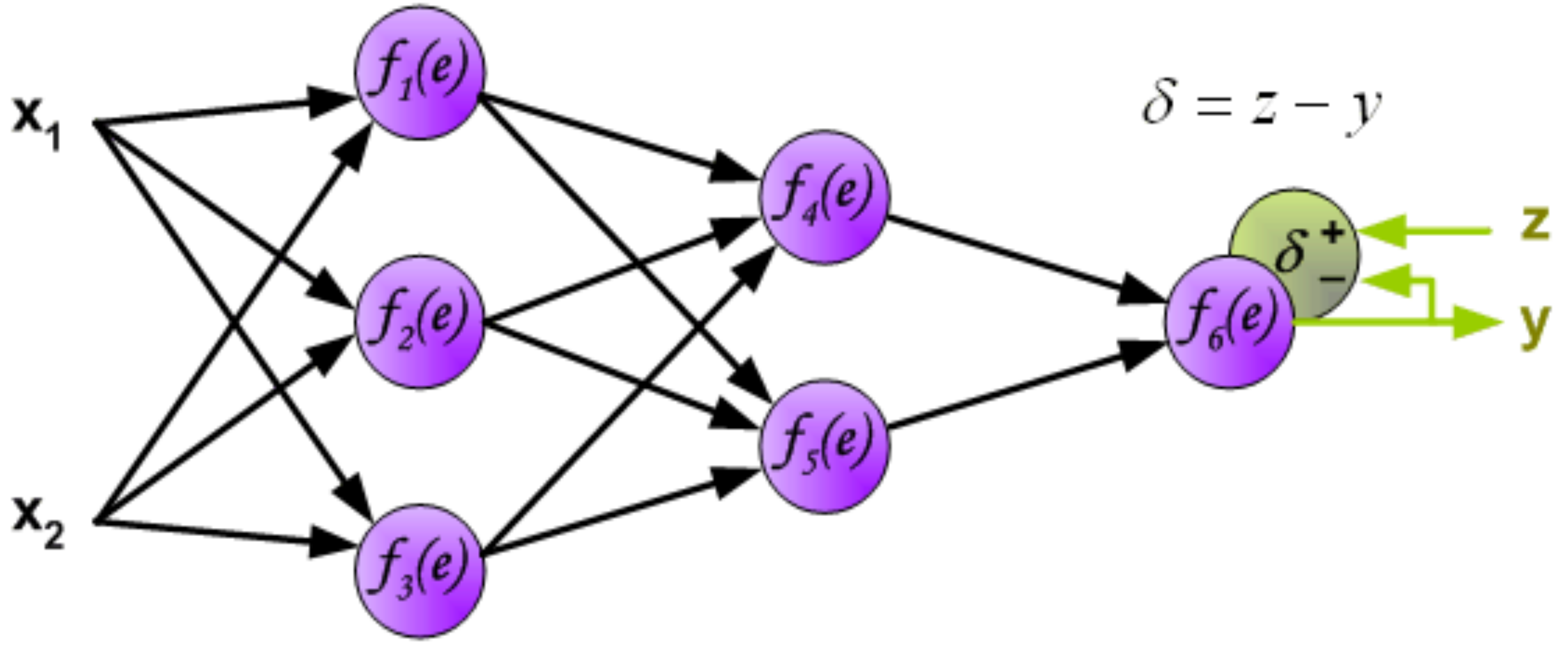
To teach the neural network we need training data set. The training data set consists of input signals (x_1 and x_2) assigned with corresponding target (desired output) z.

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

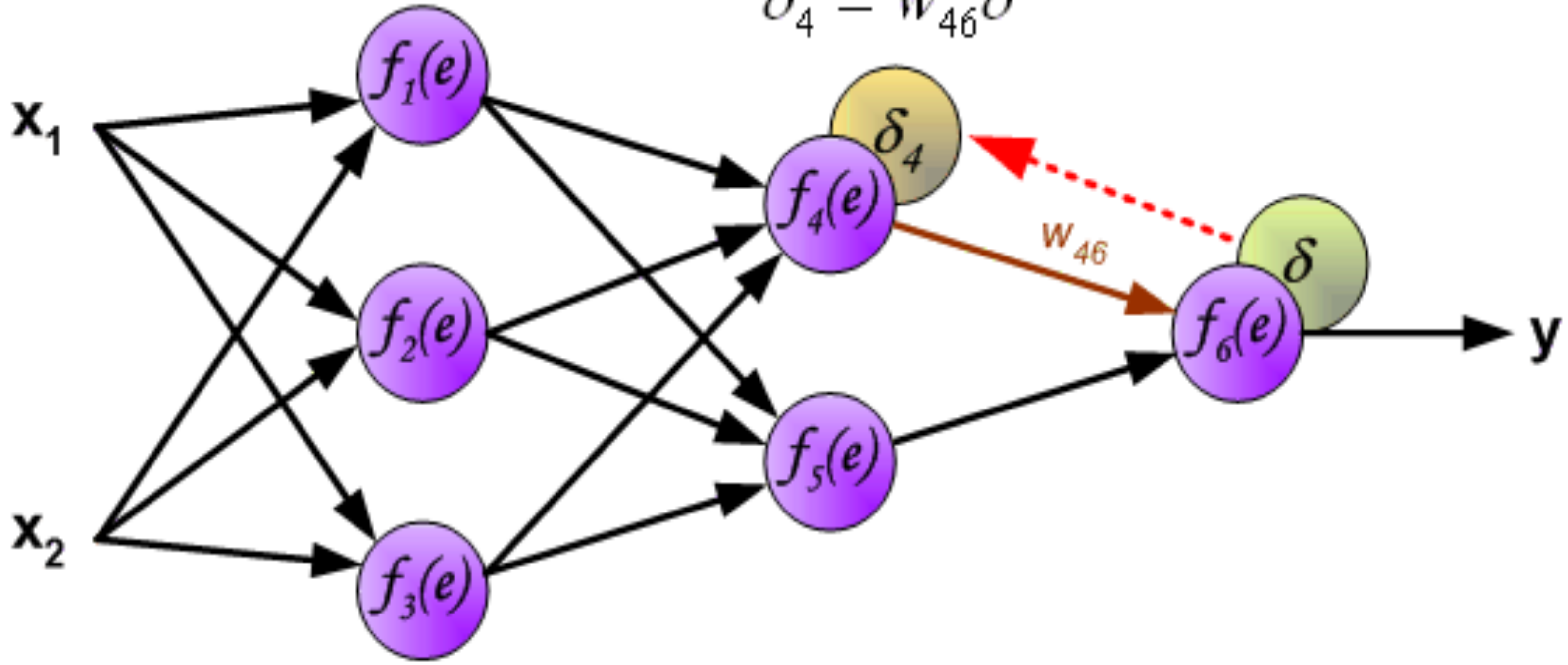
In the next algorithm step the output signal of the network *y* is compared with the desired output value (the target z), which is found in training data set. The difference is called error signal *δ* of output layer neuron



Learning Algorithm: Backpropagation

The idea is to propagate error signal δ (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron.

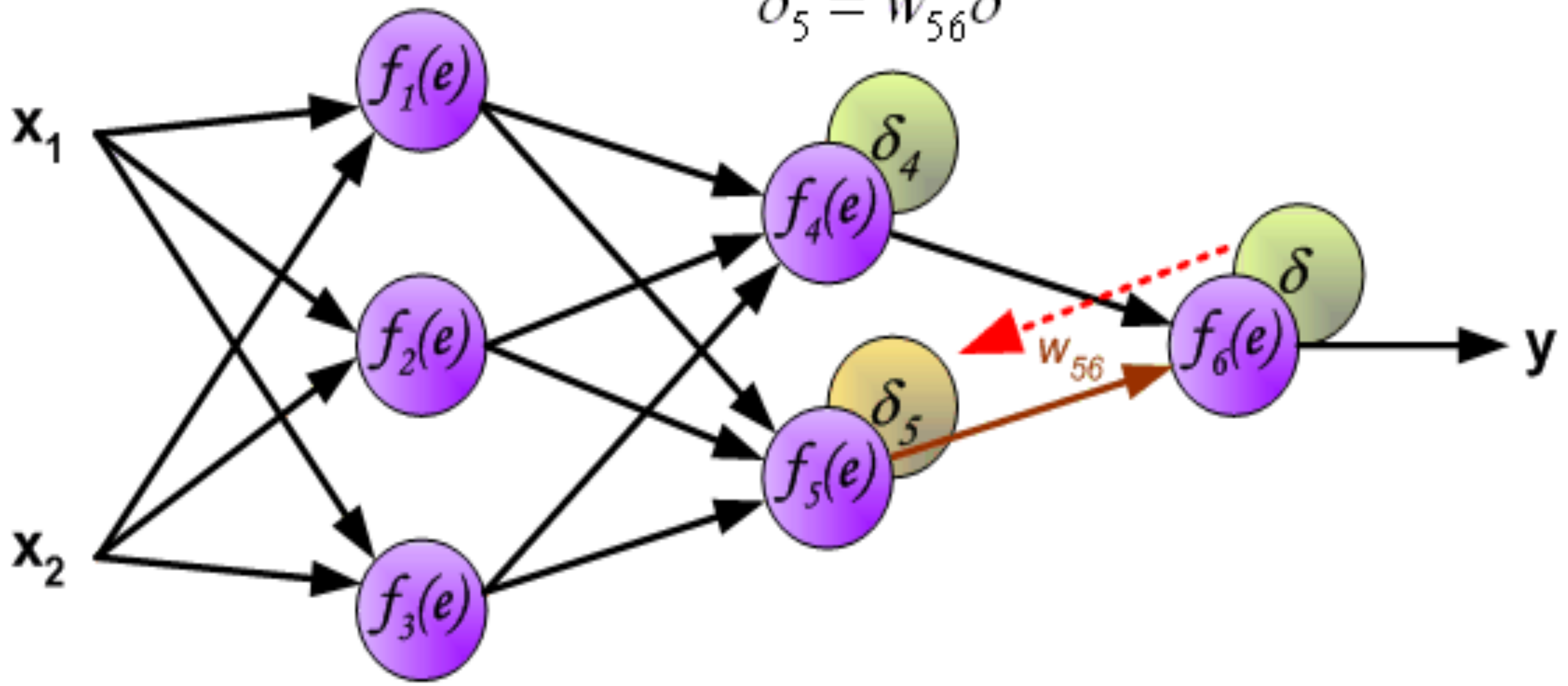
$$\delta_4 = w_{46}\delta$$



Learning Algorithm: Backpropagation

The idea is to propagate error signal δ (computed in single teaching step) back to all neurons, which output signals were input for discussed neuron

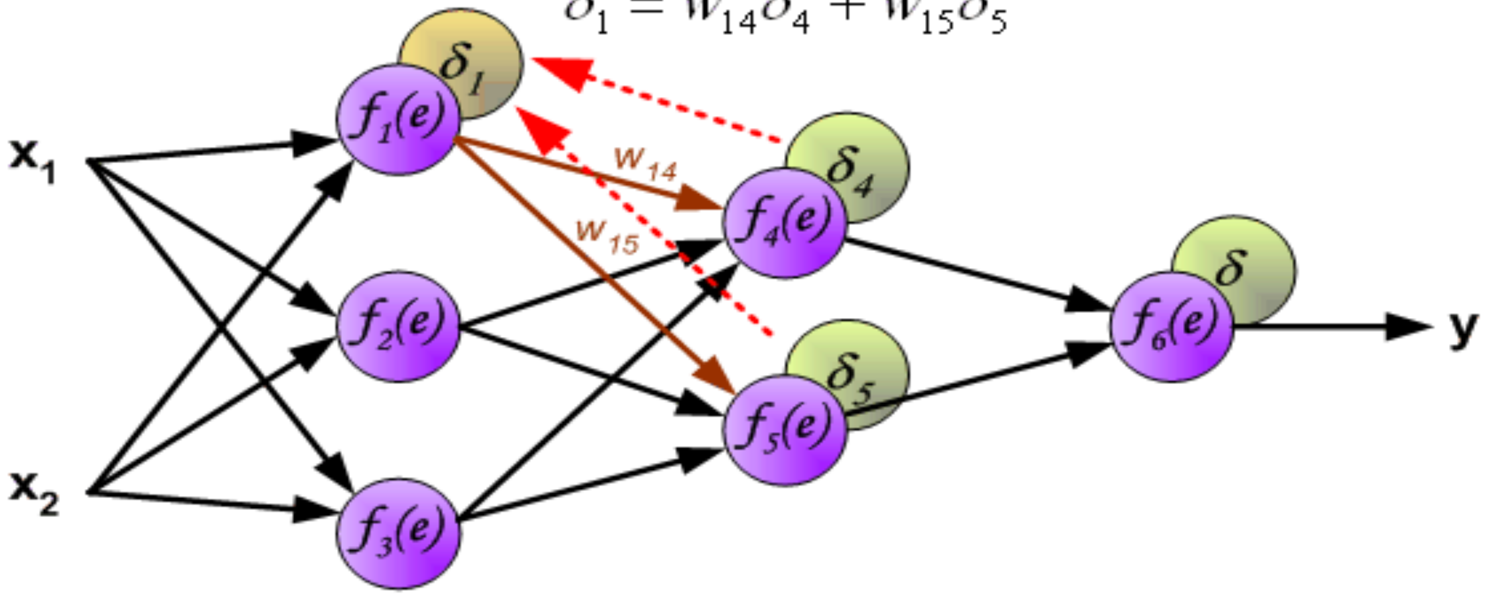
$$\delta_5 = w_{56}\delta$$



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:

$$\delta_1 = w_{14}\delta_4 + w_{15}\delta_5$$

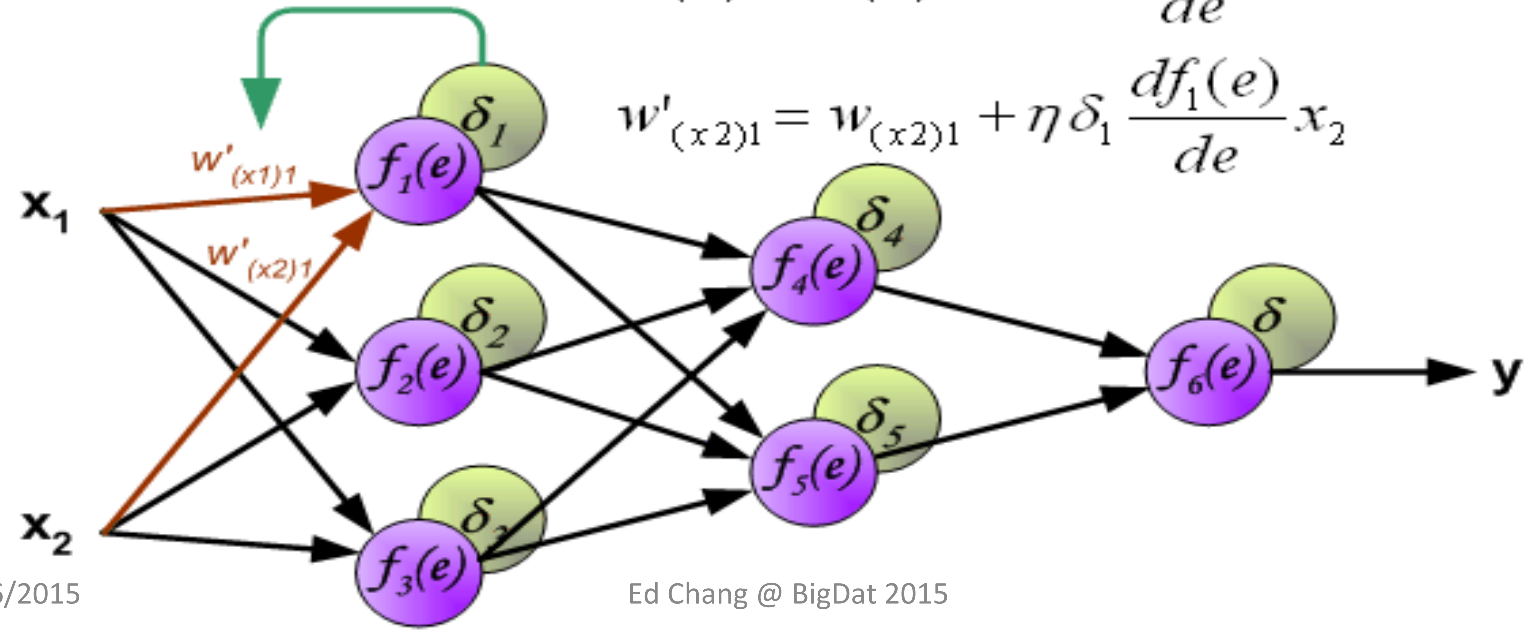


Learning Algorithm: Backpropagation

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).

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

$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$

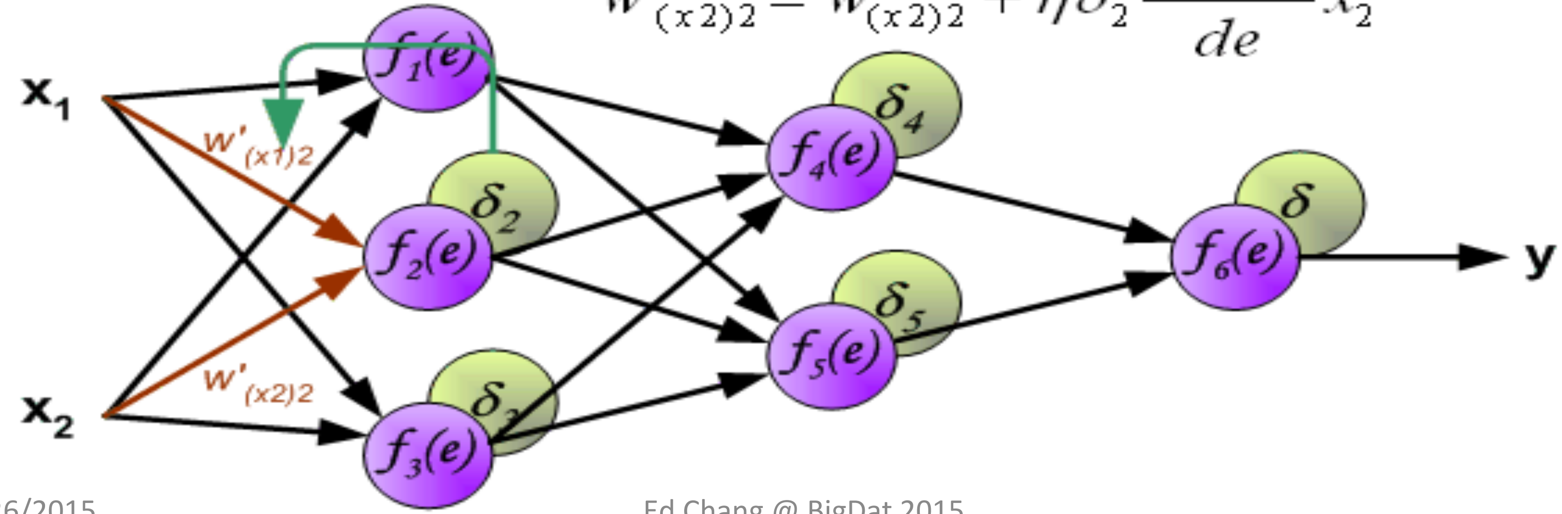


Learning Algorithm: Backpropagation

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).

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

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

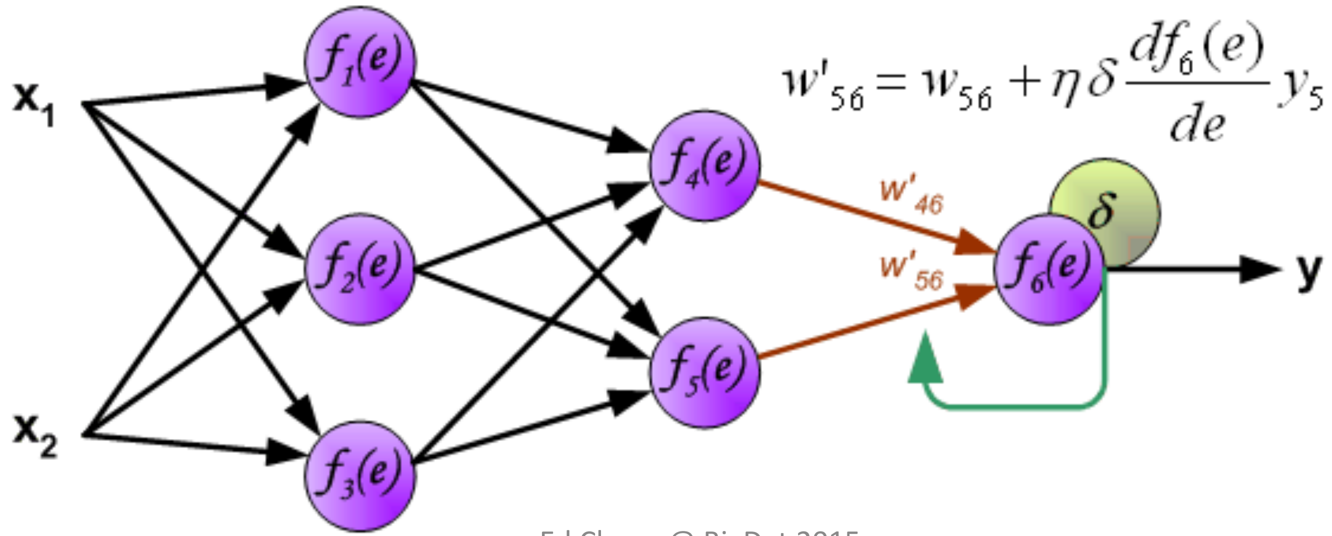


Learning Algorithm: Backpropagation

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).

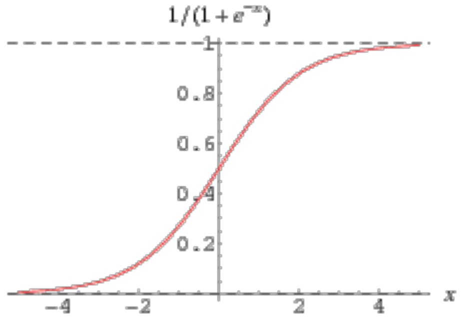
$$w'_{46} = w_{46} + \eta \delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta \delta \frac{df_6(e)}{de} y_5$$



http://mathworld.wolfram.com/SigmoidFunction.html

Sigmoid function *f(e)* and its derivative *f'(e)*



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

Hence

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

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

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

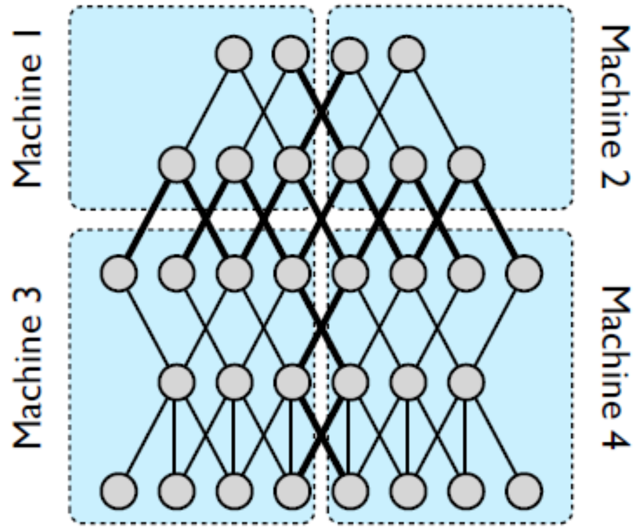
For simplicity, paramter for the slope $\beta=1$

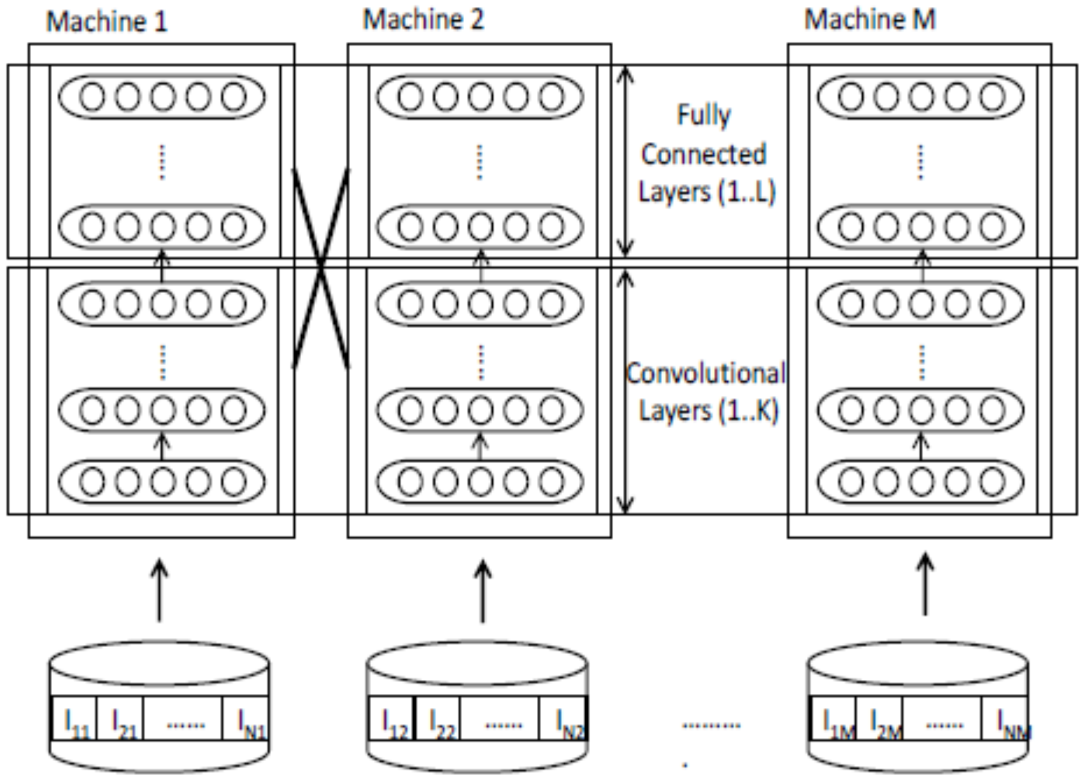
$$\boxed{f^{'}(e)=f(e)\left(1-f(e)\right)}$$

http://link.springer.com/chapter/10.1007%2F3-540-59497-3_175#page-1

Model Parallelism

[J. Dean et al, NIPS 2012]




Machine 1
Machine 2
Machine M
Fully Connected Layers (1..L)
Convolutional Layers (1..K)
I_{11} I_{21} .... I_{N1}
I_{12} I_{22} .... I_{N2}
I_{1M} I_{2M} .... I_{NM}

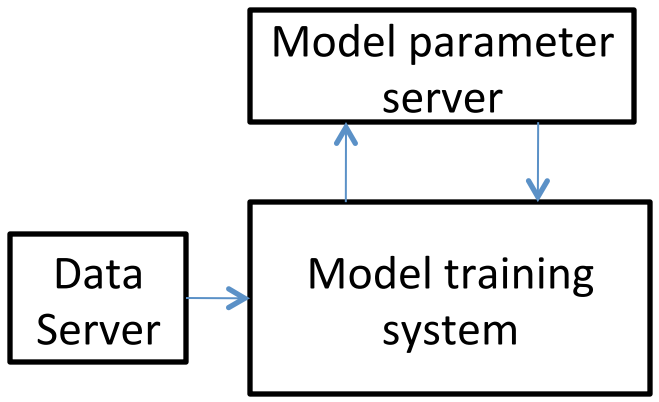
Scalable Deep Learning Platform

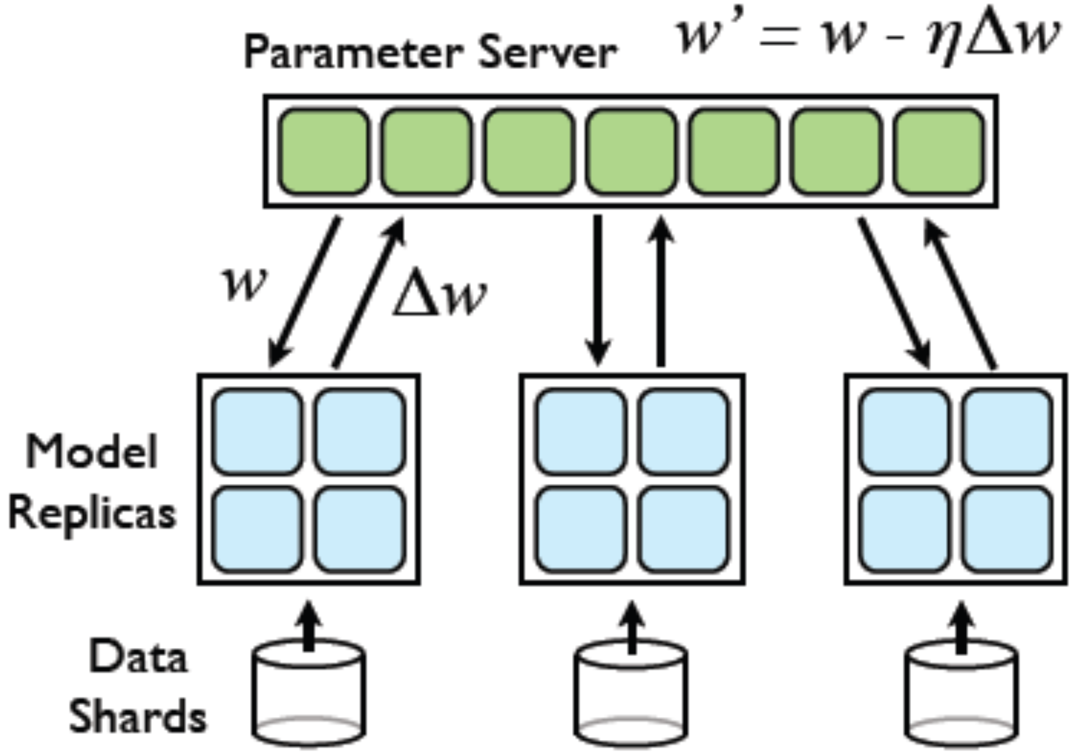
Microsoft Project ADAM

- 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

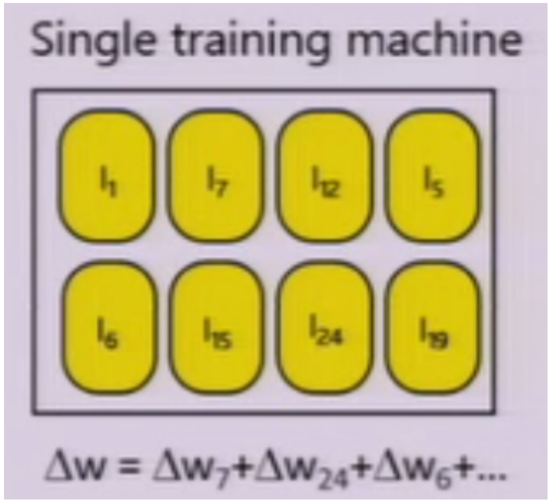




[1] Faerber, P., and Asanović, K. 1997. Parallel neural network training on Multi-Spert.

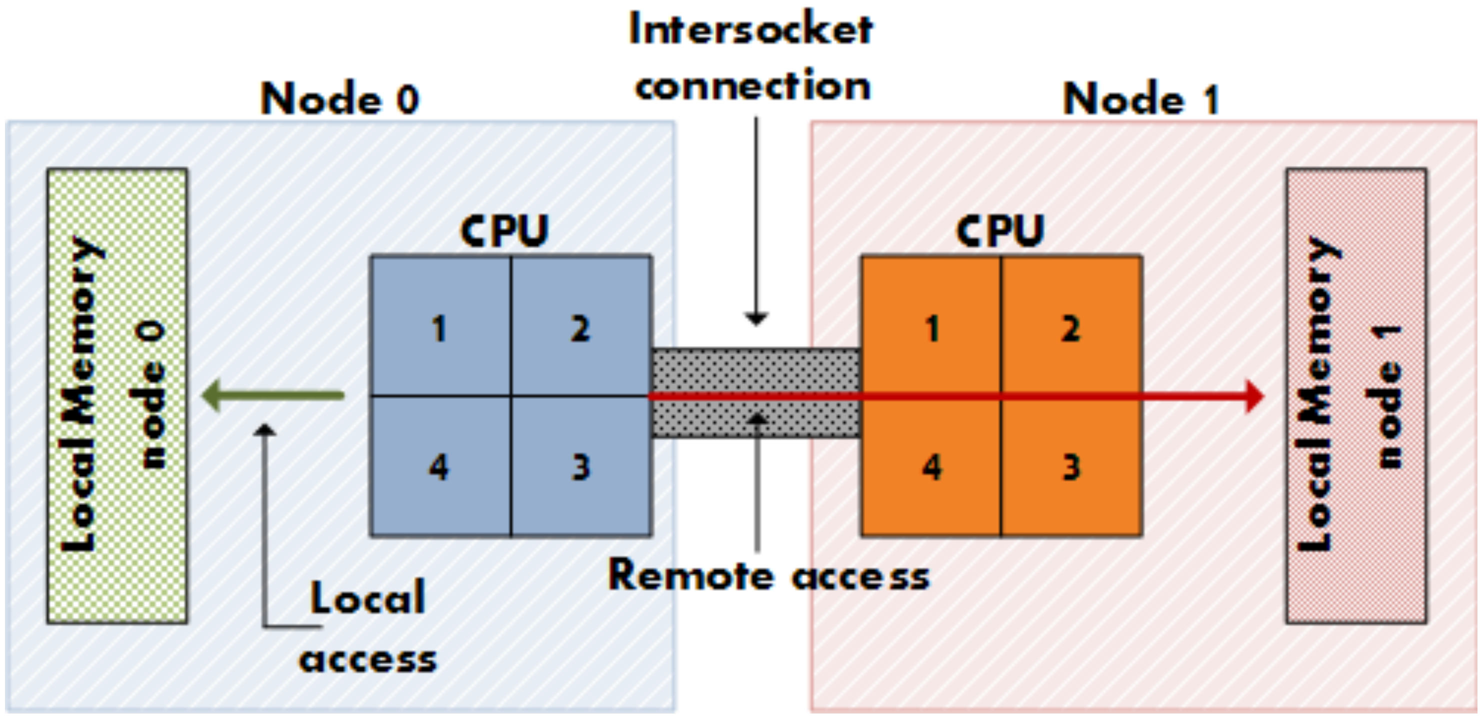
Model Training Optimizations (1/3)

- 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


Single training machine
I_1 I_7 I_{12} I_5
I_6 I_{15} I_{24} I_{19}
$\Delta w = \Delta w_7 + \Delta w_{24} + \Delta w_6 + ...$

NUMA

- Non Uniform Memory Access

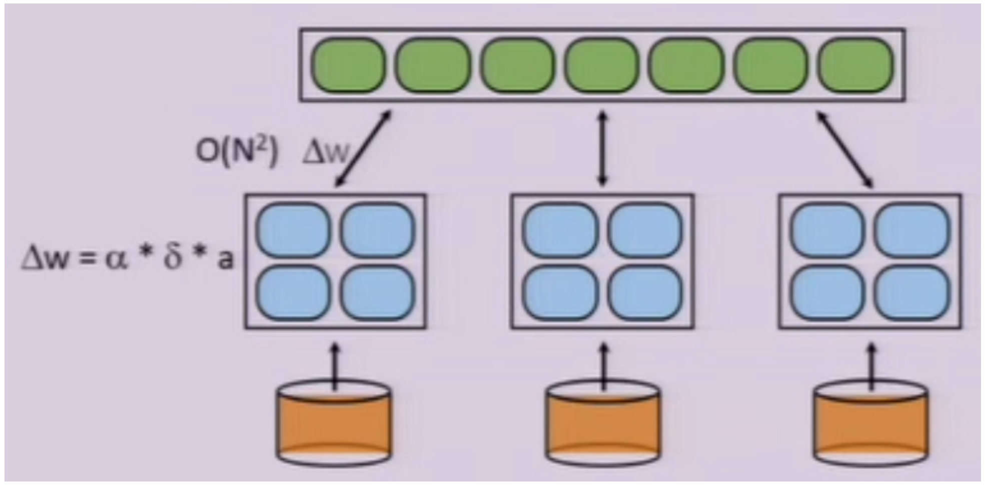


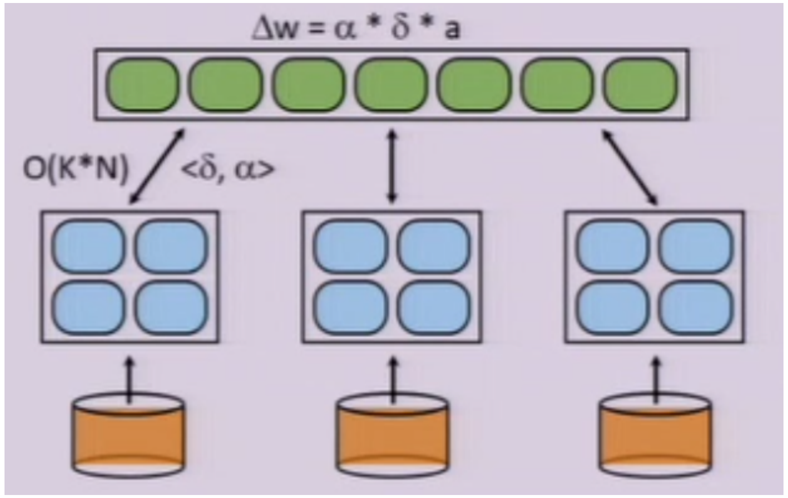
Model Training Optimizations (2/3)

- 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



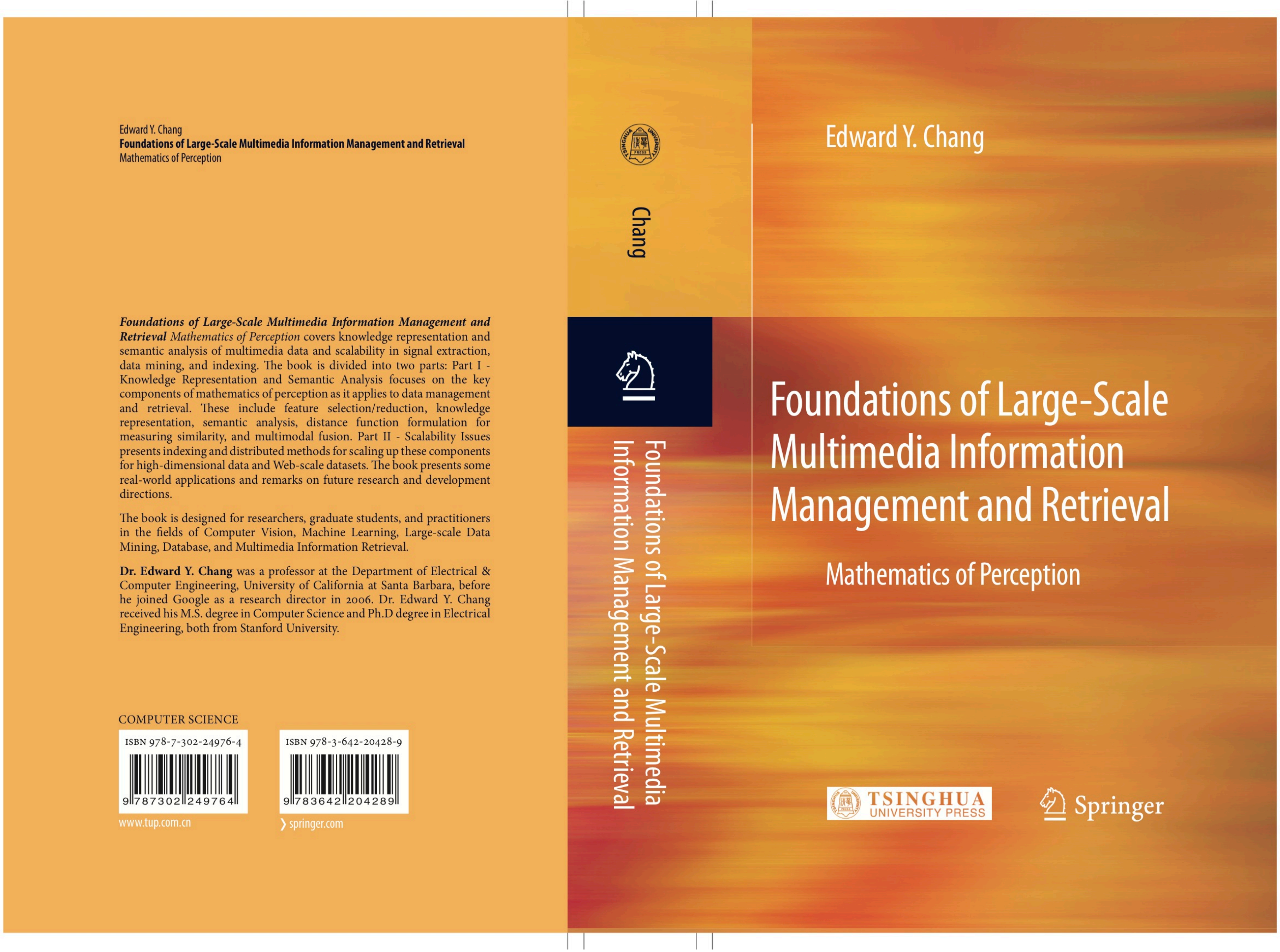


Concluding Remarks

- 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

- [ACM RS 08] PFP: Parallel FP-Growth for Query Recommendation, H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang, ACM Recommendation Systems, Lausanne, October 2008
- [TIST 10] PLDA+: Parallel Latent Dirichlet Allocation with Data Placement and Pipeline Processing, ACM Transactions on Intelligent Systems and Technology, 2011.
- [MM 01] Support Vector Machine Active Learning for Image Retrieval, S. Tong and E. Chang, ACM MM, 2001
- [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
- [NIPS 07] Parallel Support Vector Machines, E. Y. Chang, et al., NIPS 2007.
- [PAMI 10] Parallel Spectral Clustering, W.-Y. Chen, Y. Song, H. Bai, Chih-Jen Lin, and E. Y. Chang, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 2010.
- [NIPS 12] Large Scale Distributed Deep Networks. J. Dean, G.S. Corrado, R. Monga, K. Chen, M. Devin, Q.V. Le, M.Z. Mao, M.A. Ranzato, A. Senior, P. Tucker, K. Yang, A. Y. Ng, NIPS 2012.
- [OSDI 14] Project Adam: Building an Efficient and Scalable Deep Learning Training System Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman, OSDI 2014.
- [VLDB 14] Big Data, Small Footprint: The Design of a Low-Power Classifier for Detecting Transportation Modes (with Open Source dataset), M. Yu, T. Yu, C.-J. Lin, and E. Y. Chang, VLDB, August 2014.



Edward Y. Chang
Foundations of Large-Scale Multimedia Information Management and Retrieval
Mathematics of Perception

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.

COMPUTER SCIENCE

ISBN 978-7-302-24976-4
9 787302 249764

ISBN 978-3-642-20428-9
9 783642 204289

www.tup.com.cn
springer.com

Chang

Foundations of Large-Scale Multimedia Information Management and Retrieval

Edward Y. Chang

Foundations of Large-Scale Multimedia Information Management and Retrieval

Mathematics of Perception

TSINGHUA UNIVERSITY PRESS
Springer

Additional References

[1] Alexa internet. http://www.alexa.com/.

[2] D. M. Blei and M. I. Jordan. Variational methods for the dirichlet process. In Proc. of the 21st international conference on Machine learning, pages 373-380, 2004.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. Journal of Machine Learning Research, 3:993-1022, 2003.

[4] D. Cohn and H. Chang. Learning to probabilistically identify authoritative documents. In Proc. of the Seventeenth International Conference on Machine Learning, pages 167-174, 2000.

[5] D. Cohn and T. Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In Advances in Neural Information Processing Systems 13, pages 430-436, 2001.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. Journal of the American Society of Information Science, 41(6):391-407, 1990.

[7] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society. Series B (Methodological), 39(1):1-38, 1977.

[8] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern recognition and Machine Intelligence, 6:721-741, 1984.

[9] T. Hofmann. Probabilistic latent semantic indexing. In Proc. of Uncertainty in Articial Intelligence, pages 289-296, 1999.

[10] T. Hofmann. Latent semantic models for collaborative filtering. ACM Transactions on Information System, 22(1):89-115, 2004.

[11] A. McCallum, A. Corrada-Emmanuel, and X. Wang. The author-recipient-topic model for topic and role discovery in social networks: Experiments with enron and academic email. Technical report, Computer Science, University of Massachusetts Amherst, 2004.

[12] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent dirichlet allocation. In Advances in Neural Information Processing Systems 20, 2007.

[13] M. Ramoni, P. Sebastiani, and P. Cohen. Bayesian clustering by dynamics. Machine Learning, 47(1):91-121, 2002.

References (cont.)

[14] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative ltering. In Proc. Of the 24th international conference on Machine learning, pages 791-798, 2007.

[15] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In Proc. of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining, pages 678-684, 2005.

[16] M. Steyvers, P. Smyth, M. Rosen-Zvi, and T. Griths. Probabilistic author-topic models for information discovery. In Proc. of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 306-315, 2004.

[17] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. Journal on Machine Learning Research (JMLR), 3:583-617, 2002.

[18] T. Zhang and V. S. Iyengar. Recommender systems using linear classiers. Journal of Machine Learning Research, 2:313-334, 2002.

[19] S. Zhong and J. Ghosh. Generative model-based clustering of documents: a comparative study. Knowledge and Information Systems (KAIS), 8:374-384, 2005.

[20] L. Admic and E. Adar. How to search a social network. 2004

[21] T.L. Griffiths and M. Steyvers. Finding scientific topics. Proceedings of the National Academy of Sciences, pages 5228-5235, 2004.

[22] H. Kautz, B. Selman, and M. Shah. Referral Web: Combining social networks and collaborative filtering. Communitcations of the ACM, 3:63-65, 1997.

[23] R. Agrawal, T. Imielnski, A. Swami. Mining association rules between sets of items in large databses. SIGMOD Rec., 22:207-116, 1993.

[24] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the Fourteenth Conference on Uncertainty in Artifical Intelligence, 1998.

[25] M.Deshpande and G. Karypis. Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst., 22(1):143-177, 2004.

References (cont.)

[26] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International World Wide Web Conference, pages 285-295, 2001.
[27] M.Deshpande and G. Karypis. Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst., 22(1):143-177, 2004.
[28] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International World Wide Web Conference, pages 285-295, 2001.
[29] M. Brand. Fast online svd revisions for lightweight recommender systems. In Proceedings of the 3rd SIAM International Conference on Data Mining, 2003.
[30] D. Goldbberg, D. Nichols, B. Oki and D. Terry. Using collaborative filtering to weave an information tapestry. Communication of ACM 35, 12:61-70, 1992.
[31] P. Resnik, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for aollaborative filtering of netnews. In Proceedings of the ACM, Conference on Computer Supported Cooperative Work. Pages 175-186, 1994.
[32] J. Konstan, et al. Grouplens: Applying collaborative filtering to usenet news. Communication ofACM 40, 3:77-87, 1997.
[33] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating “word of mouth”. In Proceedings of ACM CHI, 1:210-217, 1995.
[34] G. Kinden, B. Smith and J. York. Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Computing, 7:76-80, 2003.
[35] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. Machine Learning Journal 42, 1:177-196, 2001.
[36] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In Proceedings of International Joint Conference in Artificial Intelligence, 1999.
[37] http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/collaborativefiltering.html