Machine Learning at Scale

James G. Shanahan [2]

Assistants: Liang Dai[1, 3]

[1]NativeX, [2]iSchool UC Berkeley, CA, [3]UC Santa Cruz

EMAIL: James_DOT_Shanahan_AT_gmail_DOT_com

Live Session #7

Feb 23, 2016

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW5, HW6, HW7 (due Thursday 8AM of Week 9)
  - MidTerm week 8
  - Bernoulli Naïve Bayes; Bernoulli Mixture Models
  - Async lecture recap plus Q&A
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

HW updates

- **Homework HW6: Extension to Saturday at 8AM for HW6 if you need it.**
- **HW7 (due Thursday 8AM of Week 9)**

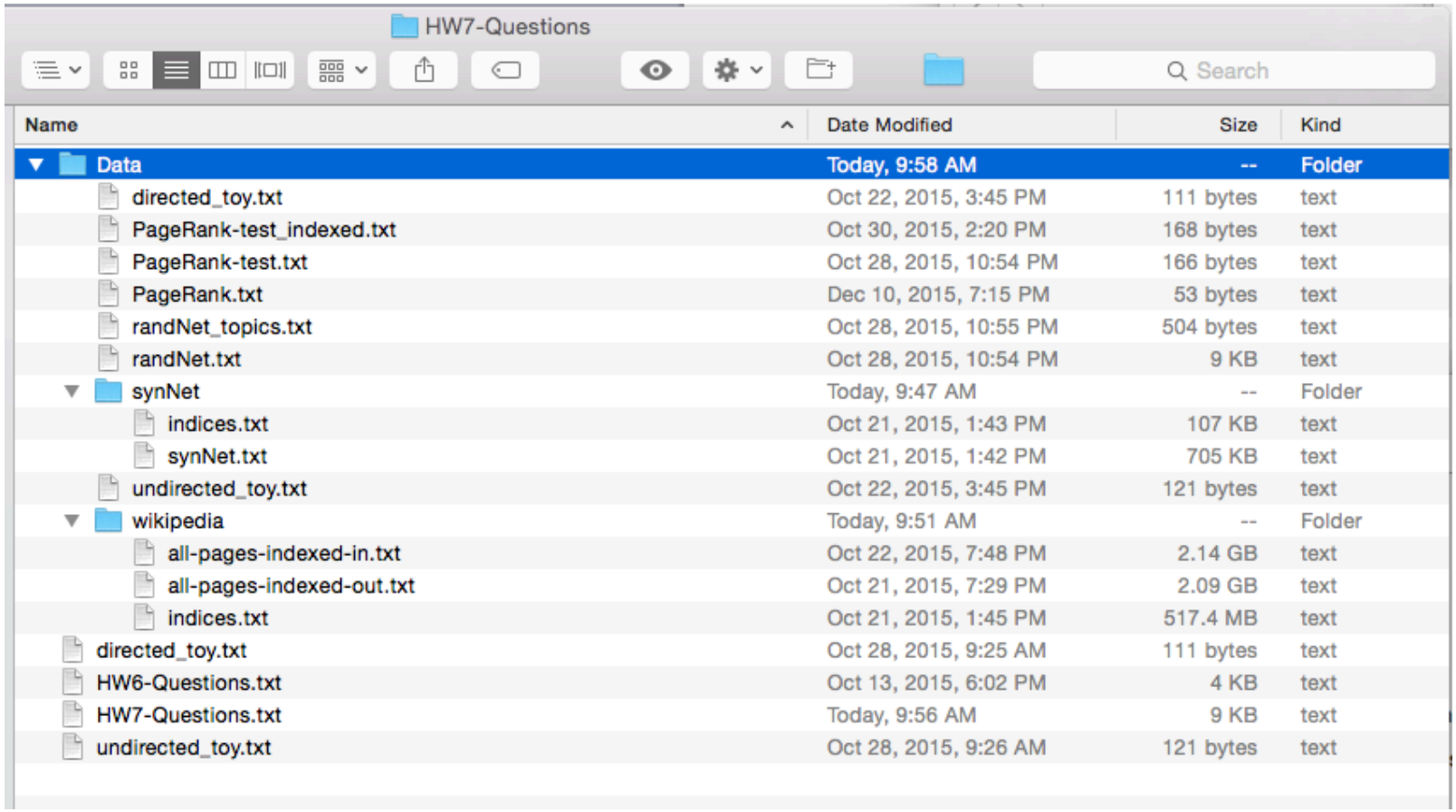
Important Links for Week 7

- **Live session Slides**

- **Instructions for Peer grading of homework HW**
  - https://www.dropbox.com/s/97m31frthj4ac28/HOMEWORK%20GRADING%20INSTRUCTIONS%20for%20MIDS%20MLS?dl=0
- **Homework HW Folder Questions + Data**
  - HW is a group oriented homework (Check out the Data sub folder)
  - https://www.dropbox.com/s/26ejqhkzqdidzwj/HW7-Questions.txt?dl=0
- **Team assignments**
  - https://docs.google.com/spreadsheets/d/1ncFQl5Tovn-16slD8mYjP_nzMTPSfiGeLLzW8v_sMjg/edit?usp=sharing
- **Please submit your homeworks (one per team) going forward via this form (and not thru the ISVC):**
  - https://docs.google.com/forms/d/1ZOr9Rnle_A06AcZDB6K1mJN4vrLeSmS2PD6Xm3eOiis/viewform?usp=send_form

HW7: Data Details

- https://www.dropbox.com/s/26ejqhkzqdidzwj/HW7-Questions.txt?dl=0



aws s3 sync s3://ucb-mids-mls-networks .

Time to complete your Survey

- https://www.surveymonkey.com/r/JXT56QT

Spring 2016 Mid Course Evaluation - DATA SCI W261.4 Shanahan

MID SEMESTER REVIEW FORM

COURSE NO: DATA SCI W261.4

INSTRUCTOR: JAMES SHANAHAN

TERM: SPRING 2016

Please consider all relevant aspects of the course, such as cases, lectures, class discussion, teaching style, and readings, and answer the following questions:

1. What do you find most valuable about this course?

2. With a realistic view as to what can be changed at this point in the semester, what one aspect should the instructor change in this course? How?

Peer Grading

- **HW5 will be peer graded as will HW6**
  - http://nbviewer.ipython.org/urls/dl.dropbox.com/.../fplz0d8nt4ioupx/JRW_hw5.ipynb

Thank you

- **Homework HW 6 Folder Questions + Data**
  - HW is a group oriented homework
  - https://www.dropbox.com/s/ctehxulla3o2sim/HW6-Questions.txt?dl=0

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW5, HW6, HW7 (due Thursday 8AM of Week 9)
  - MidTerm week 8
  - Bernoulli Naïve Bayes; Bernoulli Mixture Models
  - Async lecture recap plus Q&A
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

MidTerm Exam: Week 8

- **TIME**
  - Week 8 live sessions will be midterm exam time (no need to log into the classroom).
  - The midterm will be 1.5 to 2 (two) hours long and will held during your regular live sessions.
- **Rules**
  - No collusion/consulting with others (classmates, people who took the class previously)
  - It will be a take home exam and open book (so no need to be in the classroom).
- **Format**
  - Questions will be text or programming based
  - Exam problems will follow the same style as your lectures/homework but will be multiple choice
  - All programming exercises can be completed on your local machines using MRJob
  - Turn in your multiple choice answers plus and a notebook (with your answers)
- **Focus**
  - Focus of the mid term will be lectures 1 to 6 (not week 7).

Unit 1 | Introduction and Motivation for Machine Learning at Scale

Show Contents

Unit 2 | Parallel Computing, MapReduce, and Hadoop (Data Storage and Algorithms)

Show Contents

Unit 3 | MapReduce Algorithm Design

Show Contents

Unit 4 | MRJob, Unsupervised Learning at Scale: Clustering, Canopy-Ba Means, and Expectation Maximization

Show Contents

Unit 5 | Big Data Pipelines

Show Contents

Unit 6 | Distributed Supervised Machine Learning, Part 1

Show Contents

Focus of Mid Term

Plus live session material

- Naïve Bayes Gaussian, Multinomial, Bernoulli
- Hadoop counters
- Association rule mining (aPriori)
- Clustering: Knn; GMM; BMM
- TFIDF, similarity function Inverted Index

Unit 1 | Introduction and Motivation for Machine Learning at Scale

Hide Contents

- 1.1 Weekly Introduction (2 mins)
- 1.2 Assigned Readings
- 1.3 Big Data Definitions (8 mins)
- 1.4 Sources of Big Data: Society (12 mins)
- 1.5 Internet of Things (7 mins)
- 1.6 Data Modeling Pipeline (7 mins)
- 1.7 Data Scientist: Post This Class (9 mins)
- 1.8 Goals of This Class (9 mins)
- 1.9 Logistics and Performance Evaluation (2 mins)
- 1.10 Large-Scale Machine Learning (7 mins)
- 1.11 Bias-Variance Background (16 mins)
  - 1.11.1 Quiz: Bias-Variance
- 1.12 Poor Man's MapReduce Using Command Line (Part 1
  - 1.12.1 Poor Man's MapReduce Using Command Line (
  - 1.12.2 Quiz: Parallel Grep via a Command Line Based
  - 1.12.3 Solution: Parallel Grep via a Command Line Bas

  Restricted: 'Not available until the activity 1.12.2 Quiz: Parallel G MapReduce is marked complete.'
- 1.13 Summary (6 mins)

Higher order functions
No side effects

Unit 2 | Parallel Computing, MapReduce, and Hadoop (Data Storage and Algorithms)

Hide Contents

- 2.1 Weekly Introduction (2 mins)
- 2.2 Assigned Readings
- 2.3 Motivation for Parallel Computing (8 mins)
- 2.4 Parallel Computing Definition and Communication Synchronization Types of PC Tasks (16 mins)
  - 2.4.1 Quiz: Embarrassingly Parallel Problems
- 2.5 Architectures for Parallel Computation (11 mins)
- 2.6 Developer Frameworks for Parallel Computation (13 mins)
  - 2.6.1 Quiz: Shared Nothing
- 2.7 Hadoop Background and History (8 mins)
- 2.8 Hadoop File System (8 mins)
- 2.9 MapReduce: Functional Programming (9 mins)
- 2.10 Hadoop: MapReduce (9 mins)
- 2.11 Animated Examples (13 mins)
- 2.12 Summary (2 mins)

Unit 3 | MapReduce Algorithm Design

Hide Contents

- 3.1 Weekly Introduction (2 mins)
- 3.2 Assigned Readings
- 3.3 Background RAM vs Disk vs Bandwidth (5 mins)
- 3.4 Background Priority Queues and Merge Sort (4 mins)
- 3.5 Background: The Internals of the Hadoop Shuffle (15 mins)
- 3.6 Local Aggregation Combiners and In-Mapper Combining (14 mins)
- 3.7 Local Aggregation: Algorithmic Correctness (13 mins)
- 3.8 How Many Maps And Reduces (8 mins)
  - 3.8.1 Quiz: In-Memory Mapper: Combiner Required
- 3.9 Pairs and Stripes (20 mins)
- 3.10 Computing Relative Frequencies (10 mins)
- 3.11 Summary (3 mins)

Unit 4 | MRJob, Unsupervised Learning at Scale: Clustering, Canopy-Based K-Means, and Expectation Maximization

Hide Contents

- 4.1 Weekly Introduction (2 mins)
- 4.2 Assigned Readings
- 4.3 Background and Motivation (5 mins)
- 4.4 mrjob Installation (4 mins)
  - 4.4.1 Quiz: Install mrjob and Verify
- 4.5 mrjob Fundamentals and Concepts (7 mins)
- 4.6 Writing mrjob Code (10 mins)
  - 4.6.1 Quiz: Word Frequency Challenge
- 4.7 Log File Processing (8 mins)
  - 4.7.1 Quiz: Log File Processing Challenge
- 4.8 Thought Experiment Bad Logs (5 mins)
- 4.9 Serializable JSON (15 mins)
- 4.10 mrjob Benchmarks (7 mins)
- 4.11 Clustering Overview (7 mins)
- 4.12 K-Means Algorithm (7 mins)

Unit 5 | Big Data Pipelines

Hide Contents

- 5.1 Weekly Introduction (3 mins)
- 5.2 Assigned Readings
- 5.3 Mobile Advertising Example (19 mins)
- 5.4 Data Warehousing OLTP OLAP (11 mins)
- 5.5 Database Relational Joins (12 mins)
- 5.6 Databases Operations in MapReduce (10 mins)
- 5.7 Relational Joins (11 mins)
- 5.8 Reduce Side Join (8 mins)
- 5.9 Reduce Side Join Example (4 mins)
  - 5.9.1 Quiz: Reducer-Side Join
- 5.10 Memory Backed Map Side Join (2 mins)
- 5.11 Map Side Join Merge Join (4 mins)

GMM; BMM

Unit 6 | Distributed Supervised Machine Learning, Part 1

Hide Contents

- 6.1 Weekly Introduction (2 mins)
- 6.2 Assigned Readings
- 6.3 Supervised Machine Learning Parametric vs Non (8 mins)
- 6.4 Core Supporting Concepts Matrices Applications of Matrices (5 mins)
- 6.5 Matrices: Vector by Vector Multiplication (12 mins)
  - 6.5.1 Matrices Distributed Matrix by Vector Multiplication Part 2 (6 mins)
  - 6.5.2 Matrices: Matrix-by-Matrix Multiplication Version 1.1 (12 mins)
  - 6.5.3 Matrices: Distributed Matrix by Vector Multiplication, Version 1.1 (6 mins)
- 6.6 Distributed Matrix Summary (3 mins)
- 6.7 Core Supporting Concepts: Optimization Theory: Gradient Descent (9 mins)
  - 6.7.1 Core Supporting Concepts: Optimization Theory: Gradient Descent Part 2 (9 mins)
- 6.8 Optimization Theory: Convex Optimization (11 mins)
- 6.9 Distributed Closed Form Linear Regression (5 mins)
  - 6.9.1 Quiz: Number of mrjobs
- 6.10 Complexity Analysis of Algorithms (3 mins)
- 6.11 Distributed-Gradient-Descent Approach to Linear Regression (7 mins)
  - 6.11.1 Quiz: LR via Distributed GD: Critique This Pseudo-Code
- 6.12 Summary of Linear Regression (3 mins)

Not in exam

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW5, HW6, HW7 (due Thursday 8AM of Week 9)
  - MidTerm week 8
  - Bernoulli Naïve Bayes; Bernoulli Mixture Models
  - Async lecture recap plus Q&A
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

Bernoulli Mixture Model

- **Table 16.3 in the IRBook**

In a Bernoulli Mixture Model a document is a vector of Booleans indicating the presence of a term.

The conditional probability of a document given a set of parameters is given by:

$$P(d \mid \theta) = \sum_{k=1}^{K} \alpha_k \left(\prod_{tm \in d} q_{mk}\right)\left(\prod_{tm \notin d} (1 - q_m k)\right) \quad \text{(eqn 0)}$$

This is the sum for each class of the product of the probabilities of the terms in a document with 1 minus the probabilities of the terms not in the document.

The probability that a document from cluster ω_k containts term t_m is given by:

$$q_{mk} = P(U_m = 1 \mid \omega_k)$$

The prior α_k of cluster ω_k is the probability document d is in ω_k if we have no other information about it.

When we don't know the classifications of the documents we can use Expectation Maximization iteratively to arrive at classifications, r_{nk}.

E Step

$$r_{nk} = \frac{\alpha_k (\prod_{tm \in d_n} q_{mk})(\prod_{tm \in d_n} 1 - q_{mk})}{\sum_{k=1}^{K} \alpha_k (\prod_{tm \in d_n} q_{mk})(\prod_{tm \in d_n} 1 - q_{mk})} \quad (1)$$

The actual computation as implemented by taking the sum of the log probabilities as opposed to the products of the probabilities themselves. This is necessary because once you are dealing with many terms, multiplying a lot of small numbers results in numeric underflow. Also, in order to prevent taking the log(0), which will happen in the first iteration, a very small number, ϵ, is added to the probability before taking the log. So we end up with:

$$\frac{\alpha_k e^{\left(\sum_{tm \in d_n} log(q_{mk}+\epsilon) + \sum_{tm \notin d_n} log(1-q_{mk}+\epsilon)\right)}}{\sum_k \alpha_k e^{\left(\sum_{tm \in d_n} (log(q_{mk}+\epsilon) + \sum_{tm \notin d_n} log(1-q_{mk}+\epsilon))\right)}} \quad (1a)$$

In the code below, a single pass is made to calculate the r class soft assignments so that the terms are calculated only once so it may not be obvious what's going on.

Note: the ϵ values are important for the algorithm to converge. If you take a straight calculation of the $r_{1,1}$ term in the first iteration you will end up with a divide by zero using the original equations. By including the ϵ term this is avoided and you'll see that the result is very close to 1

M Step

$$q_{mk} = \frac{\sum_{n=1}^{N} r_{nk} I(t_m \in d_n)}{\sum_{i=1}^{N} r_{nk}} \quad (2)$$

$I(t_m \in d_n) = 1$ if term is an element of document n and 0 otherwise.

Finally, priors are updated per iteration as:

$$\alpha_k = \frac{\sum_{n=1}^{N} r_{nk}}{N} \quad (3)$$

Which of the following is correct:
A: Eqn 0, 1, 1a, 2, 3 are all correct
B: Eqn 1 is incorrect
C: equation 2 is incorrect
D: None of the above

[Thanks to Ron Cordell for leading the way here]

In a Bernoulli Mixture Model a document is a vector of Booleans indicating the presence of a term.

The conditional probability of a document given a set of parameters is given by:

Pr(Cluster|D; θ) Posterior Class Prob

$$P(D_i|C) \sim P(\mathbf{b}_i|C) = \prod_{t=1}^{|V|} [b_{it}P(w_t|C) + (1 - b_{it})(1 - P(w_t|C))] .$$

$$P(d \mid \theta) = \sum_{k=1}^{K} \alpha_k (\prod_{tm \in d} q_{mk})(\prod_{tm \notin d} (1 - q_m k)) \quad \text{(eqn 0)}$$

Which of the following is correct:
A: Eqn 0, 1, 1a, 2, 3 are all correct
B: Eqn 1 is incorrect
C: equation 2 is incorrect
D: None of the above

This is the sum for each class of the product of the probabilities of the terms in a document with 1 minus the probabilities of the terms not in the document.

The probability that a document from cluster ω_k containts term t_m is given by:

$$q_{mk} = P(U_m = 1 \mid \omega_k)$$

Word class or cluster conditional

The prior α_k of cluster ω_k is the probability document d is in ω_k if we have no other information about it.

When we don't know the classifications of the documents we can use Expectation Maximization iteratively to arrive at classifications, r_{nk}.

E Step

$$r_{nk} = \frac{\alpha_k (\prod_{tm \in d_n} q_{mk})(\prod_{tm \in d_n} 1 - q_{mk})}{\sum_{k=1}^{K} \alpha_k (\prod_{tm \in d_n} q_{mk})(\prod_{tm \in d_n} 1 - q_{mk})} \quad (1)$$

Pr(Cluster$_k$|D$_n$; θ)

The actual computation as implemented by taking the sum of the log probabilities as opposed to the products of the probabilities themselves. This is necessary because once you are dealing with many terms, multiplying a lot of small numbers results in numeric underflow. Also, in order to prevent taking the log(0), which will happen in the first iteration, a very small number, ϵ, is added to the probability before taking the log. So we end up with:

$$\frac{\alpha_k e^{(\sum_{tm \in d_n} log(q_{mk}+\epsilon) + \sum_{tm \notin d_n} log(1-q_{mk}+\epsilon))}}{\sum_k \alpha_k e^{(\sum_{tm \in d_n} (log(q_{mk}+\epsilon) + \sum_{tm \notin d_n} log(1-q_{mk}+\epsilon)))}} \quad (1a)$$

Pr(Cluster$_k$|D$_n$; θ) with modifications
Incorrect. Smoothing needs to be added to eqn 2.

In the code below, a single pass is made to calculate the n class soft assignments so that the terms are c
going on.

Note: the ϵ values are important for the algorithm to converge. If you take a straight calculation of the $r_{1,1}$
by zero using the original equations. By including the ϵ term this is avoided and you'll see that the result is

M Step

$$q_{mk} = \frac{\sum_{n=1}^{N} r_{nk} I(t_m \in d_n)}{\sum_{i=1}^{N} r_{nk}} \quad (2)$$

Eqn 2 P(term$_m$|Cluster$_k$) with no smoothing. Please add smoothing here. (not the +1 laplace smoothing but the +0.0001 smoothing (for r$_{nk}$)

$I(t_m \in d_n) = 1$ if term is an element of document n and 0 otherwise.

Finally, priors are updated per iteration as:

$$\alpha_k = \frac{\sum_{n=1}^{N} r_{nk}}{N} \quad (3)$$

Class or cluster prior

Learning a Bernoulli Naïve Bayes Model

Example 13.2: Applying the Bernoulli model to the example in Table 13.1, we have the same estimates for the priors as before: $\hat{P}(c) = 3/4$, $\hat{P}(\overline{c}) = 1/4$. The conditional probabilities are:

Japan does NOT occur class C
So gets a default prob of 1/5

$$\hat{P}(\text{Chinese}|c) = (3+1)/(3+2) = 4/5$$
$$\hat{P}(\text{Japan}|c) = \hat{P}(\text{Tokyo}|c) = (0+1)/(3+2) = 1/5$$
$$\hat{P}(\text{Beijing}|c) = \hat{P}(\text{Macao}|c) = \hat{P}(\text{Shanghai}|c) = (1+1)/(3+2) = 2/5$$
$$\hat{P}(\text{Chinese}|\overline{c}) = (1+1)/(1+2) = 2/3$$
$$\hat{P}(\text{Japan}|\overline{c}) = \hat{P}(\text{Tokyo}|\overline{c}) = (1+1)/(1+2) = 2/3$$
$$\hat{P}(\text{Beijing}|\overline{c}) = \hat{P}(\text{Macao}|\overline{c}) = \hat{P}(\text{Shanghai}|\overline{c}) = (0+1)/(1+2) = 1/3$$

Bernoulli NB: Smoothing is based on the number of classes

The denominators are $(3+2)$ and $(1+2)$ because there are three documents in c and one document in $\overline{c}$ and because the constant B in Equation (13.7) is 2 – there are two cases to consider for each term, occurrence and nonoccurrence.

The scores of the test document for the two classes are

$$\begin{aligned}\hat{P}(c|d_5) &\propto \hat{P}(c) \cdot \hat{P}(\text{Chinese}|c) \cdot \hat{P}(\text{Japan}|c) \cdot \hat{P}(\text{Tokyo}|c) \\ &\quad \cdot (1-\hat{P}(\text{Beijing}|c)) \cdot (1-\hat{P}(\text{Shanghai}|c)) \cdot (1-\hat{P}(\text{Macao}|c)) \\ &= 3/4 \cdot 4/5 \cdot 1/5 \cdot 1/5 \cdot (1-2/5) \cdot (1-2/5) \cdot (1-2/5) \\ &\approx 0.005\end{aligned}$$

- 6 terms in Vocabulary 6 terms in calculation of P(C|X)
- Count single occurrence of term, .e.g., Chinese is just counted once per document

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

and, analogously,

$$\begin{aligned}\hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot 2/3 \cdot 2/3 \cdot 2/3 \cdot (1-1/3) \cdot (1-1/3) \cdot (1-1/3) \\ &\approx 0.022\end{aligned}$$

Thus, the classifier assigns the test document to $\overline{c}$ = *not-China*. When looking only at binary occurrence and not at term frequency, Japan and Tokyo are indicators for $\overline{c}$ ($2/3 > 1/5$) and the conditional probabilities of Chinese for c and $\overline{c}$ are not different enough (4/5 vs. 2/3) to affect the classification decision.

Decision Rule for Bernoulli Naïve Bayes Model

Use all vocabulary for classification (regardless if the word occurred or not in the test example)

► **Table 13.3** Multinomial versus Bernoulli model.

	multinomial model	Bernoulli model
event model	generation of token	generation of document
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
document representation	$d = \langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle, t_k \in V$	$d = \langle e_1, \ldots, e_i, \ldots, e_M \rangle$, $e_i \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t\|c)$	$\hat{P}(U_i = e\|c)$
decision rule: maximize	$\hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(X = t_k\|c)$	$\hat{P}(c) \prod_{t_i \in V} \hat{P}(U_i = e_i\|c)$
multiple occurrences	taken into account	ignored
length of docs	can handle longer docs	works best for short docs
# features	can handle more	works best with fewer
estimate for term the	$\hat{P}(X = \text{the}\|c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1\|c) \approx 1.0$

Bernoulli Naïve Bayes model: Smoothing to Avoid Overfitting and Zero probabilities

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

# of values of X_i

Laplace smoothing for Bernoulli model
k is number of classes here

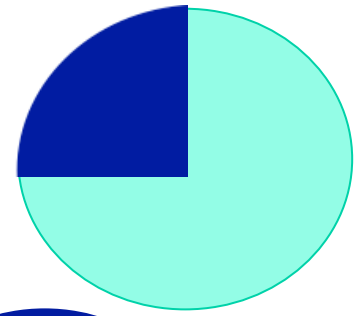
Multinomial NB model: Smoothing to Avoid Overfitting and Zero probabilities

1000 words in the Hadoop vocab and 10^9 in English
Background model dominates the foreground model. NOT good!

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

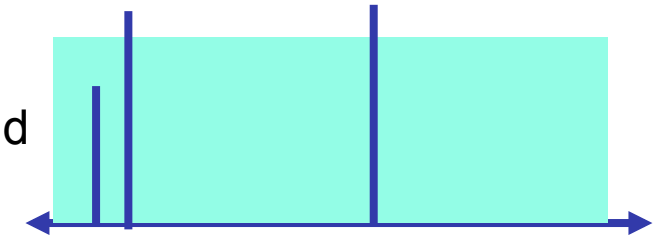
+0.0001

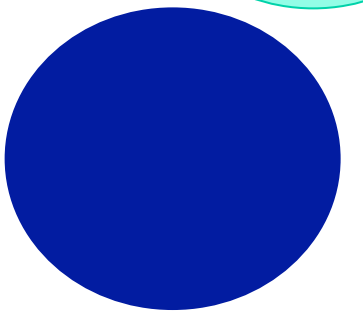
# of values of X_i



Laplace smoothing for Multinomial model
k is vocabulary size

Cyan:Background
Blue: foreground





Text Classification using Naive Bayes

Hiroshi Shimodaira*

10 February 2015

Text classification is the task of classifying documents by their content: that is, by the words of which they are comprised. Perhaps the best-known current text classification problem is email *spam filtering*: classifying email messages into spam and non-spam (ham).

1 Document models

Text classifiers often don't use any kind of deep representation about language: often a document is represented as a *bag of words*. (A bag is like a set that allows repeating elements.) This is an extremely simple representation: it only knows which words are included in the document (and how many times each word occurs), and throws away the word order!

Consider a document D, whose class is given by C. In the case of email spam filtering there are two classes $C = S$ (spam) and $C = H$ (ham). We classify D as the class which has the highest posterior probability $P(C|D)$, which can be re-expressed using Bayes' Theorem:

$$P(C|D) = \frac{P(D|C)\,P(C)}{P(D)} \propto P(D|C)\,P(C)\,. \tag{1}$$

We shall look at two probabilistic models of documents, both of which represent documents as a bag of words, using the Naive Bayes assumption. Both models represent documents using feature vectors whose components correspond to word types. If we have a vocabulary V, containing $|V|$ word types, then the feature vector dimension $d = |V|$.

Bernoulli document model: a document is represented by a feature vector with binary elements taking value 1 if the corresponding word is present in the document and 0 if the word is not present.

Multinomial document model: a document is represented by a feature vector with integer elements whose value is the frequency of that word in the document.

Example: Consider the vocabulary:

$$V = \{blue, red, dog, cat, biscuit, apple\}\,.$$

In this case $|V| = d = 6$. Now consider the (short) document "the blue dog ate a blue biscuit". If $\mathbf{d}^B$ is the Bernoulli feature vector for this document, and $\mathbf{d}^M$ is the multinomial feature vector, then we would have:

$$\mathbf{d}^B = (1,0,1,0,1,0)^T$$
$$\mathbf{d}^M = (2,0,1,0,1,0)^T$$

To classify a document we use equation (1), which requires estimating the likelihoods of the document given the class, $P(D|C)$ and the class prior probabilities $P(C)$. To estimate the likelihood, $P(D|C)$, we use the Naive Bayes assumption applied to whichever of the two document models we are using.

*Heavily based on notes inherited from Steve Renals and Iain Murray.

2 The Bernoulli document model

As mentioned above, in the Bernoulli model a document is represented by a binary vector, which represents a point in the space of words. If we have a vocabulary V containing a set of $|V|$ words, then the t th dimension of a document vector corresponds to word w_t in the vocabulary. Let $\mathbf{b}_i$ be the feature vector for the i th document D_i; then the t th element of $\mathbf{b}_i$, written b_{it}, is either 0 or 1 representing the absence or presence of word w_t in the i th document.

Let $P(w_t|C)$ be the probability of word w_t occurring in a document of class C; the probability of w_t not occurring in a document of this class is given by $(1 - P(w_t|C))$. If we make the naive Bayes assumption, that the probability of each word occurring in the document is independent of the occurrences of the other words, then we can write the document likelihood $P(D_i|C)$ in terms of the individual word likelihoods $P(w_t|C)$:

$$P(D_i|C) \sim P(\mathbf{b}_i|C) = \prod_{t=1}^{|V|} [b_{it}P(w_t|C) + (1 - b_{it})(1 - P(w_t|C))]\,. \tag{2}$$

This product goes over all words in the vocabulary. If word w_t is present, then $b_{it} = 1$ and the required probability is $P(w_t|C)$; if word w_t is not present, then $b_{it} = 0$ and the required probability is $1 - P(w_t|C)$. We can imagine this as a model for generating document feature vectors of class C, in which the document feature vector is modelled as a collection of $|V|$ weighted coin tosses, the t th having a probability of success equal to $P(w_t|C)$.

The *parameters* of the likelihoods are the probabilities of each word given the document class $P(w_t|C)$; the model is also parameterised by the prior probabilities, $P(C)$. We can learn (estimate) these parameters from a training set of documents labelled with class $C = k$. Let $n_k(w_t)$ be the number of documents of class $C = k$ in which w_t is observed; and let N_k be the total number of documents of that class. Then we can estimate the parameters of the word likelihoods as,

$$\hat{P}(w_t \mid C = k) = \frac{n_k(w_t)}{N_k}\,, \tag{3}$$

the relative frequency of documents of class $C = k$ that contain word w_t. If there are N documents in total in the training set, then the prior probability of class $C = k$ may be estimated as the relative frequency of documents of class $C = k$:

$$\hat{P}(C = k) = \frac{N_k}{N}\,. \tag{4}$$

Thus given a training set of documents (each labelled with a class), and a set of K classes, we can estimate a Bernoulli text classification model as follows:

1. Define the vocabulary V; the number of words in the vocabulary defines the dimension of the feature vectors
2. Count the following in the training set:
   - N the total number of documents
   - N_k the number of documents labelled with class $C=k$, for $k=1,\ldots,K$
   - $n_k(w_t)$ the number of documents of class $C=k$ containing word w_t for every class and for each word in the vocabulary
3. Estimate the likelihoods $P(w_t \mid C=k)$ using equation (3)
4. Estimate the priors $P(C=k)$ using equation (4)

To classify an unlabelled document D_j, we estimate the posterior probability for each class combining equations (1) and (2):

$$\begin{aligned} P(C|D_j) &= P(C|\mathbf{b}_j) \\ &\propto P(\mathbf{b}_j|C)\,P(C) \\ &\propto P(C)\prod_{t=1}^{|V|}\left[b_{jt}P(w_t|C)+(1-b_{jt})(1-P(w_t|C))\right]. \end{aligned} \tag{5}$$

Example

Consider a set of documents, each of which is related either to *Sports* (S) or to *Informatics* (I). Given a training set of 11 documents, we would like to estimate a Naive Bayes classifier, using the Bernoulli document model, to classify unlabelled documents as S or I.

We define a vocabulary of eight words:

$$V = \begin{bmatrix} w_1 = \text{goal}, \\ w_2 = \text{tutor}, \\ w_3 = \text{variance}, \\ w_4 = \text{speed}, \\ w_5 = \text{drink}, \\ w_6 = \text{defence}, \\ w_7 = \text{performance}, \\ w_8 = \text{field} \end{bmatrix}$$

Thus each document is represented as an 8-dimensional binary vector.

The training data is presented below as a matrix for each class, in which each row represents an 8-dimensional document vector

$$\mathbf{B}^{\text{Sport}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\mathbf{B}^{\text{Inf}} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Classify the following into Sports or Informatics using a Naive Bayes classifier.

1. $\mathbf{b}_1 = (1,0,0,1,1,1,0,1)^T$
2. $\mathbf{b}_2 = (0,1,1,0,1,0,1,0)^T$

Solution:

The total number of documents in the training set $N=11$; $N_S=6$, $N_I=5$

Using (4), we can estimate the prior probabilities from the training data as:

$$P(S) = \frac{6}{11}; \qquad P(I) = \frac{5}{11}$$

The word counts in the training data are:

$$\begin{aligned} n_S(w_1) &= 3 & n_S(w_2) &= 1 \\ n_S(w_3) &= 2 & n_S(w_4) &= 3 \\ n_S(w_5) &= 3 & n_S(w_6) &= 4 \\ n_S(w_7) &= 4 & n_S(w_8) &= 4 \end{aligned}$$

$$\begin{aligned} n_I(w_1) &= 1 & n_I(w_2) &= 3 \\ n_I(w_3) &= 3 & n_I(w_4) &= 1 \\ n_I(w_5) &= 1 & n_S(w_6) &= 1 \\ n_S(w_7) &= 3 & n_S(w_8) &= 1 \end{aligned}$$

The we can estimate the word likelihoods using (3)

$$\begin{aligned} P(w_1|S) &= \frac{1}{2} & P(w_2|S) &= \frac{1}{6} \\ P(w_3|S) &= \frac{1}{3} & P(w_4|S) &= \frac{1}{2} \\ P(w_5|S) &= \frac{1}{2} & P(w_6|S) &= \frac{2}{3} \\ P(w_7|S) &= \frac{2}{3} & P(w_8|S) &= \frac{2}{3} \end{aligned}$$

And for class I:

$$P(w_1|I) = \frac{1}{5} \qquad P(w_2|I) = \frac{3}{5}$$
$$P(w_3|I) = \frac{3}{5} \qquad P(w_4|I) = \frac{1}{5}$$
$$P(w_5|I) = \frac{1}{5} \qquad P(w_6|I) = \frac{1}{5}$$
$$P(w_7|I) = \frac{3}{5} \qquad P(w_8|I) = \frac{1}{5}$$

We use (5) to compute the posterior probabilities of the two test vectors and hence classify them.

1. $\mathbf{b}_1 = (1,0,0,1,1,1,0,1)^T$

$$P(S|\mathbf{b}_1) \propto P(S)\prod_{t=1}^{8}[b_{1t}P(w_t|S) + (1-b_{1t})(1-P(w_t|S))]$$
$$\propto \frac{6}{11}\left(\frac{1}{2}\times\frac{5}{6}\times\frac{2}{3}\times\frac{1}{2}\times\frac{1}{2}\times\frac{2}{3}\times\frac{1}{3}\times\frac{2}{3}\right) = \frac{5}{891} = 5.6\times10^{-3}$$
$$P(I|\mathbf{b}_1) \propto P(I)\prod_{t=1}^{8}[b_{1t}P(w_t|I) + (1-b_{1t})(1-P(w_t|I))]$$
$$\propto \frac{5}{11}\left(\frac{1}{5}\times\frac{2}{5}\times\frac{2}{5}\times\frac{1}{5}\times\frac{1}{5}\times\frac{1}{5}\times\frac{2}{5}\times\frac{1}{5}\right) = \frac{8}{859375} = 9.3\times10^{-6}$$

   Classify this document as S.

2. $\mathbf{b}_2 = (0,1,1,0,1,0,1,0)^T$

$$P(S|\mathbf{b}_2) \propto P(S)\prod_{t=1}^{8}[b_{2t}P(w_t|S) + (1-b_{2t})(1-P(w_t|S))]$$
$$\propto \frac{6}{11}\left(\frac{1}{2}\times\frac{1}{6}\times\frac{1}{3}\times\frac{1}{2}\times\frac{1}{2}\times\frac{1}{3}\times\frac{2}{3}\times\frac{1}{3}\right) = \frac{12}{14256} = 8.4\times10^{-4}$$

$$P(I|\mathbf{b}_1) \propto P(I)\prod_{t=1}^{8}[b_{1t}P(w_t|I) + (1-b_{1t})(1-P(w_t|I))]$$
$$\propto \frac{5}{11}\left(\frac{4}{5}\times\frac{3}{5}\times\frac{3}{5}\times\frac{4}{5}\times\frac{1}{5}\times\frac{4}{5}\times\frac{3}{5}\times\frac{4}{5}\right) = \frac{34560}{4296875} = 8.0\times10^{-3}$$

   Classify as I.

3 The multinomial distribution

Before discussing the multinomial document model, it is important to be familiar with the multinomial distribution.

We first need to be able to count the number of distinct arrangements of a set of items, when some of the items are *indistinguishable*. For example: Using all the letters, how many distinct sequences

can you make from the word "Mississippi"? There are 11 letters to permute, but "i" and "s" occur four times and "p" twice. If these letters were distinct (e.g., if they were labelled i_1, i_2, etc.) then there would be 11! permutations. However of these permutations there are 4! that are the same if the subscripts are removed from the "i"s. This means that we can reduce the size of the total sample space by a factor of 4! to take account of the four occurrences of "i". Likewise there is a factor of 4! for "s" and a factor of 2! for "p" (and a factor of 1! for "m"). This gives the total number distinct permutations as:

$$\frac{11!}{4!\,4!\,2!\,1!} = 34650$$

Generally if we have n items of d types, with n_1 of type 1, n_2 of type 2 and n_d of type d (such that $n_1 + n_2 + \ldots + n_d = n$), then the number of distinct permutations is given by:

$$\frac{n!}{n_1!\,n_2!\ldots n_d!}$$

These numbers are called the *multinomial coefficients*.

Now suppose a population contains items of $d \geq 2$ different types and that the proportion of items that are of type t is p_t ($t = 1,\ldots,d$), with

$$\sum_{t=1}^{d} p_t = 1 \qquad p_t > 0, \text{ for all } t.$$

Suppose n items are drawn at random (with replacement) and let x_t denote the number of items of type t. The vector $\mathbf{x} = (x_1,\ldots,x_d)^T$ has a *multinomial distribution* with parameters n and $p_1,\ldots,p_d$, defined by:

$$P(\mathbf{x}) = \frac{n!}{x_1!\,x_2!\ldots x_d!}\,p_1^{x_1}p_2^{x_2}\ldots p_d^{x_d}$$
$$= \frac{n!}{\prod_{t=1}^{d} x_t!}\prod_{t=1}^{d} p_t^{x_t} \qquad (6)$$

The $\prod_{t=1}^{d} p_t^{x_t}$ product gives the probability of one sequence of outcomes with counts $\mathbf{x}$. The multinomial coefficient, counts the number of such sequences that there are.

4 The multinomial document model

In the multinomial document model, the document feature vectors capture the frequency of words, not just their presence or absence. Let $\mathbf{x}_i$ be the multinomial model feature vector for the ith document D_i. The tth element of $\mathbf{x}_i$, written x_{it}, is the count of the number of times word w_t occurs in document D_i. Let $n_i = \sum_t x_{it}$ be the total number of words in document D_i.

Let $P(w_t|C)$ again be the probability of word w_t occurring in class C, this time estimated using the word frequency information from the document feature vectors. We again make the naive Bayes assumption, that the probability of each word occurring in the document is independent of the occurrences of the other words. We can then write the document likelihood $P(D_i|C)$ as a multinomial distribution (equation 6), where the number of draws corresponds to the length of the document, and the proportion

of drawing item t is the probability of word type t occurring in a document of class C, $P(w_t|C)$.

$$P(D_i|C) \sim P(\mathbf{x}_i|C) = \frac{n_i!}{\prod_{t=1}^{|V|} x_{it}!} \prod_{t=1}^{|V|} P(w_t|C)^{x_{it}}$$
$$\propto \prod_{t=1}^{|V|} P(w_t|C)^{x_{it}}. \quad (7)$$

We often often won't need the normalisation term $(n_i!/\prod_t x_{it}!)$, because it does not depend on the class, C. The numerator of the right hand side of this expression can be interpreted as the product of word likelihoods for each word in the document, with repeated words taking part for each repetition.

As for the Bernoulli model, the parameters of the likelihood are the probabilities of each word given the document class $P(w_t|C)$, and the model parameters also include the prior probabilities $P(C)$. To estimate these parameters from a training set of documents labelled with class $C=k$, let z_{ik} be an indicator variable which equals 1 when D_i has class $C=k$, and equals 0 otherwise. If N is again the total number of documents, then we have:

$$\hat{P}(w_t \mid C=k) = \frac{\sum_{i=1}^{N} x_{it} z_{ik}}{\sum_{s=1}^{|V|} \sum_{i=1}^{N} x_{is} z_{ik}}, \quad (8)$$

an estimate of the probability $P(w_t \mid C=k)$ as the relative frequency of w_t in documents of class $C=k$ with respect to the total number of words in documents of that class.

The prior probability of class $C=k$ is estimated as before (equation 4).

Thus given a training set of documents (each labelled with a class) and a set of K classes, we can estimate a multinomial text classification model as follows:

1. Define the vocabulary V; the number of words in the vocabulary defines the dimension of the feature vectors.
2. Count the following in the training set:
   - N the total number of documents,
   - N_k the number of documents labelled with class $C=k$, for each class $k=1,\ldots,K$,
   - x_{it} the frequency of word w_t in document D_i, computed for every word w_t in V.
3. Estimate the likelihoods $P(w_t \mid C=k)$ using (8).
4. Estimate the priors $P(C=k)$ using (4).

To classify an unlabelled document D_j, we estimate the posterior probability for each class combining (1) and (7):

$$P(C|D_j) = P(C|\mathbf{x}_j)$$
$$\propto P(\mathbf{x}_j|C)\,P(C)$$
$$\propto P(C) \prod_{t=1}^{|V|} P(w_t|C)^{x_{it}}. \quad (9)$$

Unlike the Bernoulli model, words that do not occur in the document (i.e., for which $x_{it}=0$) do not affect the probability (since $p^0=1$). Thus we can write the posterior probability in terms of words u which occur in the document:

$$P(C|D_j) \propto P(C) \prod_{h=1}^{len(D_j)} P(u_h|C)$$

Where u_h is the hth word in document D_i.

5 The Zero Probability Problem

A drawback of relative frequency estimates—equation (8) for the multinomial model—is that zero counts result in estimates of zero probability. This is a bad thing because the Naive Bayes equation for the likelihood (7) involves taking a product of probabilities: if any one of the terms of the product is zero, then the whole product is zero. This means that the probability of the document belonging to the class in question is zero—which means it is impossible.

Just because a word does not occur in a document class in the training data does not mean that it cannot occur in any document of that class.

The problem is that equation (8) *underestimates* the likelihoods of words that do not occur in the data. Even if word w is not observed for class $C=k$ in the training set, we would still like $P(w \mid C=k) > 0$. Since probabilities must sum to 1, if unobserved words have underestimated probabilities, then those words that are observed must have overestimated probabilities. Therefore, one way to alleviate the problem is to remove a small amount of probability allocated to observed events and distribute this across the unobserved events. A simple way to do this, sometimes called *Laplace's law of succession* or *add one smoothing*, adds a count of one to each word type. If there are W word types in total, then equation (8) may be replaced with:

$$P_{\text{Lap}}(w_t \mid C=k) = \frac{1 + \sum_{i=1}^{N} x_{it} z_{ik}}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{N} x_{is} z_{ik}} \quad (10)$$

The denominator was increased to take account of the $|V|$ extra "observations" arising from the "add 1" term, ensuring that the probabilities are still normalised.

Question: The Bernoulli document model can also suffer from the zero probability model. How would you apply add one smoothing in this case?

6 Comparing the two models

The Bernoulli and the multinomial document models are both based on a bag of words. However there are a number of differences, which we summarise here:

1. **Underlying model of text:**
   Bernoulli: a document can be thought of as being generated from a multidimensional Bernoulli distribution: the probability of a word being present can be thought of as a (weighted) coin flip with probability $P(w_t|C)$.
   Multinomial: a document is formed by drawing words from a multinomial distribution: you can think of obtaining the next word in the document by rolling a (weighted) $|V|$-sided dice with probabilities $P(w_t|C)$.

2. **Document representation:**
   Bernoulli: binary vector, elements indicating presence or absence of a word.
   Multinomial: integer vector, elements indicating frequency of occurrence of a word.

3. **Multiple occurrences of words:**
   Bernoulli: ignored.
   Multinomial: taken into account.

4. **Behaviour with document length:**
   Bernoulli: best for short documents.
   Multinomial: longer documents are OK.

5. **Behaviour with "the":**
   Bernoulli: since "the" is present in almost every document, $P(\text{"the"}|C) \sim 1.0$.
   Multinomial: since probabilities are based on relative frequencies of word occurrence in a class, $P(\text{"the"}|C) \sim 0.05$.

7 Conclusion

In this chapter we have shown how the Naive Bayes approximation can be used for document classification, by constructing distributions over words. The classifiers require a *document model* to estimate $P(\text{document} \mid \text{class})$. We looked at two document models that we can use with the Naive Bayes approximation:

- **Bernoulli document model:** a document is represented by a binary feature vector, whose elements indicate absence or presence of corresponding word in the document.
- **Multinomial document model:** a document is represented by an integer feature vector, whose elements indicate frequency of corresponding word in the document.

Laplace Smoothing

- Suppose we estimate a probability P(z) and we have n_0 examples where $z = 0$ and n_1 examples where $z = 1$
  MLE estimate is $$P(z = 1) = \frac{n_1}{n_0 + n_1}$$

- Laplace Smoothing. Add 1 to the numerator and 2 to the denominator $$P(z = 1) = \frac{n_1 + 1}{n_0 + n_1 + 2}$$

  If we don't observe any examples, we expect P(z=1) = 0.5, but our belief is weak (equivalent to seeing one example of each outcome).

Naïve Bayes Reading : Bernoulli versus multinomial

- http://classes.engr.oregonstate.edu/eecs/spring2012/cs534/notes/Naivebayes-6.pdf
- http://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note07-2up.pdf
- **A Comparison of Event Models for Naive Bayes Text Classication by McCallum and Nigam**
  - https://www.dropbox.com/s/p1m360p21v55v7p/NaiveBayes-Comparison-Nigam.pdf?dl=0

- **Naïve Bayes**

- https://www.dropbox.com/s/6jp8vngx8zje83u/NaiveBayes-Bernoulli-Notes.pdf?dl=0
- **http://www.inf.ed.ac.uk/teaching/courses/inf2b/learnnotes/inf2b-learn-note07-2up.pdf**

- ..

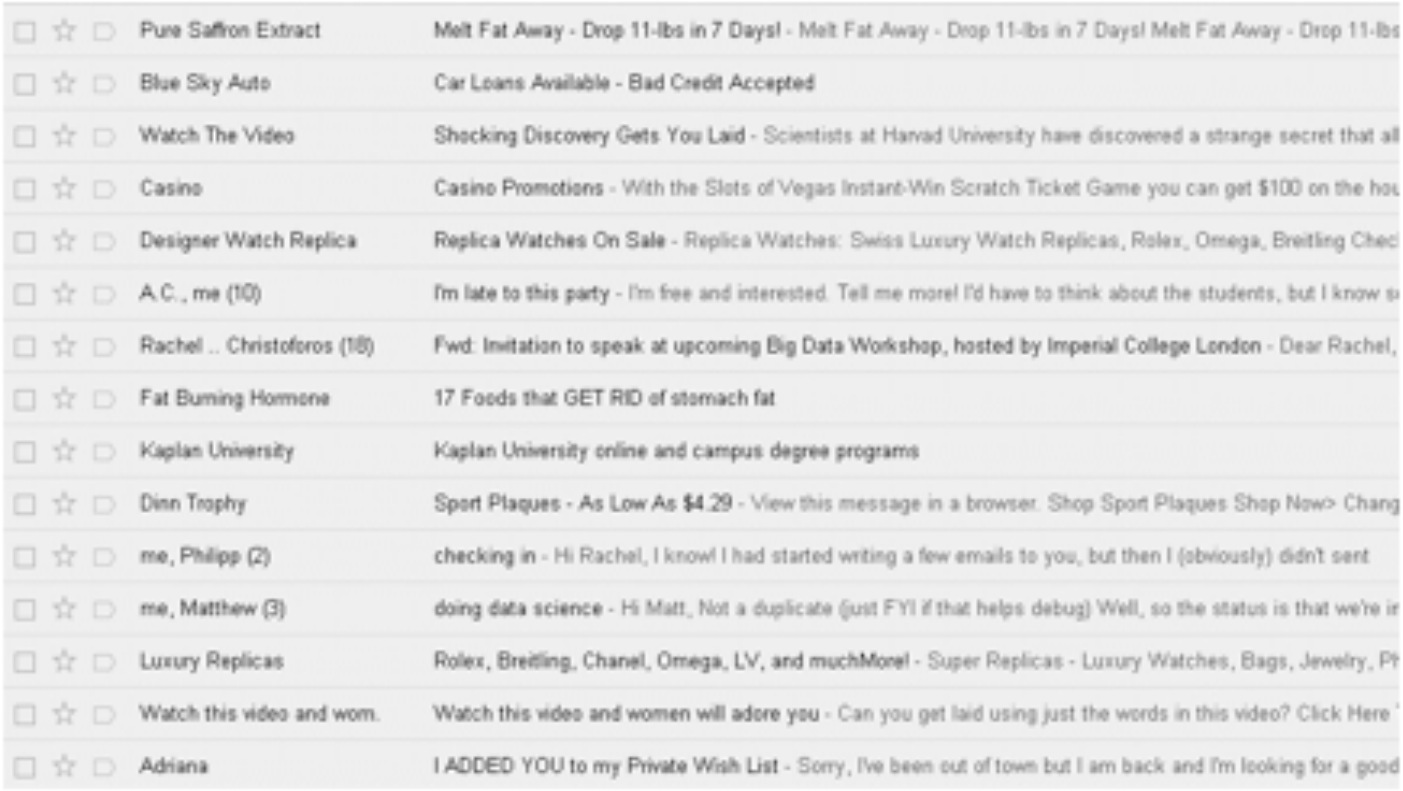
- ..

Thought Experiment: Learning by Example

Let's start by looking at a bunch of text shown in Figure 4-1, whose rows seem to contain the subject and first line of an email in an inbox.

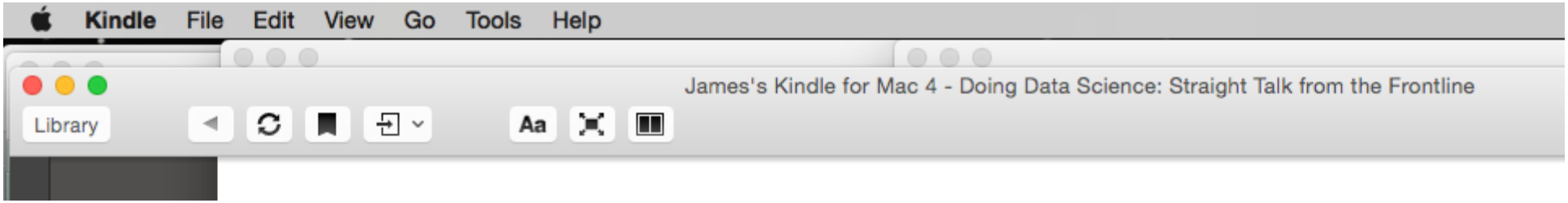
You may notice that several of the rows of text look like spam.

How did you figure this out? Can you write code to automate the spam filter that your brain represents?



Pure Saffron Extract	Melt Fat Away - Drop 11-lbs in 7 Days! - Melt Fat Away - Drop 11-lbs in 7 Days! Melt Fat Away - Drop 11-lbs
Blue Sky Auto	Car Loans Available - Bad Credit Accepted
Watch The Video	Shocking Discovery Gets You Laid - Scientists at Harvad University have discovered a strange secret that all
Casino	Casino Promotions - With the Slots of Vegas Instant-Win Scratch Ticket Game you can get $100 on the hou
Designer Watch Replica	Replica Watches On Sale - Replica Watches: Swiss Luxury Watch Replicas, Rolex, Omega, Breitling Chec
A.C., me (10)	I'm late to this party - I'm free and interested. Tell me more! I'd have to think about the students, but I know s
Rachel .. Christoforos (18)	Fwd: Invitation to speak at upcoming Big Data Workshop, hosted by Imperial College London - Dear Rachel,
Fat Burning Hormone	17 Foods that GET RID of stomach fat
Kaplan University	Kaplan University online and campus degree programs
Dinn Trophy	Sport Plaques - As Low As $4.29 - View this message in a browser. Shop Sport Plaques Shop Now> Chang
me, Philipp (2)	checking in - Hi Rachel, I know! I had started writing a few emails to you, but then I (obviously) didn't sent
me, Matthew (3)	doing data science - Hi Matt, Not a duplicate (just FYI if that helps debug) Well, so the status is that we're ir
Luxury Replicas	Rolex, Breitling, Chanel, Omega, LV, and muchMore! - Super Replicas - Luxury Watches, Bags, Jewelry, Pr
Watch this video and wom.	Watch this video and women will adore you - Can you get laid using just the words in this video? Click Here '
Adriana	I ADDED YOU to my Private Wish List - Sorry, I've been out of town but I am back and I'm looking for a good

Figure 4-1. Suspiciously spammy



This similarly answers the question: given the data I saw, which parameter θ is the most likely?

Here we will apply the spirit of Bayes's Law to transform θ_{MAP} to get something that is, up to a constant, equivalent to $p(D \mid \theta) \cdot p(\theta)$. The term $p(\theta)$ is referred to as the "prior," and we have to make an assumption about its form to make this useful. If we make the assumption that the probability distribution of θ is of the form $\theta^{\alpha}(1-\theta)^{\beta}$, for some α and β, then we recover the Laplace Smoothed result.

IS THAT A REASONABLE ASSUMPTION?

Recall that θ is the chance that a word is in spam if that word is in some email. On the one hand, as long as both $\alpha > 0$ and $\beta > 0$, this distribution vanishes at both 0 and 1. This is reasonable: you want very few words to be expected to *never* appear in spam or to *always* appear in spam.

On the other hand, when α and β are large, the shape of the distribution is bunched in the middle, which reflects the prior that most words are equally likely to appear in spam or outside spam. That doesn't seem true either.

A compromise would have α and β be positive but small, like 1/5. That would keep your spam filter from being too overzealous without having the wrong idea. Of course, you could relax this prior as you have more and better data; in general, strong priors are only needed when you don't have sufficient data.

Comparing Naive Bayes to k-NN

Sometimes α and β are called "pseudocounts." Another common name is "hyperparameters." They're fancy but also simple. It's up to you, the data scientist, to set the values of these two hyperparameters in the numerator and denominator for smoothing, and it gives you two knobs to tune. By contrast, k-NN has one knob, namely k, the number of neighbors. Naive Bayes is a linear classifier, while k-NN is not. The curse of dimensionality and large feature sets are a problem for k-NN, while Naive Bayes performs well. k-NN requires no training (just load in the dataset), whereas Naive Bayes does. Both are examples of supervised learning (the data comes labeled).

Sample Code in bash

```
#!/bin/bash
#
# file: enron_naive_bayes.sh
#
# description: trains a simple one-word naive bayes spam
# filter using enron email data
#
# usage: ./enron_naive_bayes.sh <word>
#
# requirements:
fi

# if the file doesn't exist, download from the web
if ! [ -e enron1.tar.gz ]
    then
    wget 'http://www.aueb.gr/users/ion/data/
    enron-spam/preprocessed/enron1.tar.gz'
fi

# if the directory doesn't exist, uncompress the .tar.gz
if ! [ -d enron1 ]
    then
    tar zxvf enron1.tar.gz
fi

# change into enron1
cd enron1

# get counts of total spam, ham, and overall msgs
Nspam=`ls -1 spam/*.txt | wc -l`
Nham=`ls -1 ham/*.txt | wc -l`
Ntot=$Nspam+$Nham

echo $Nspam spam examples
echo $Nham ham examples

# get counts containing word in spam and ham classes
Nword_spam=`grep -il $word spam/*.txt | wc -l`
Nword_ham=`grep -il $word ham/*.txt | wc -l`

echo $Nword_spam "spam examples containing $word"
echo $Nword_ham "ham examples containing $word"

# calculate probabilities using bash calculator "bc"
Pspam=`echo "scale=4; $Nspam / ($Nspam+$Nham)" | bc`
Pham=`echo "scale=4; 1-$Pspam" | bc`
echo
echo "estimated P(spam) =" $Pspam
echo "estimated P(ham) =" $Pham

Pword_spam=`echo "scale=4; $Nword_spam / $Nspam" | bc`
Pword_ham=`echo "scale=4; $Nword_ham / $Nham" | bc`
echo "estimated P($word|spam) =" $Pword_spam
echo "estimated P($word|ham) =" $Pword_ham

Pspam_word=`echo "scale=4; $Pword_spam*$Pspam" | bc`
Pham_word=`echo "scale=4; $Pword_ham*$Pham" | bc`
Pword=`echo "scale=4; $Pspam_word+$Pham_word" | bc`
Pspam_word=`echo "scale=4; $Pspam_word / $Pword" | bc`
echo
echo "P(spam|$word) =" $Pspam_word

# return original directory
cd ..
```

Naïve Bayes Classifier for Text

$$P(Y = y_k | X_1, X_2, ..., X_N) = \frac{P(Y=y_k) P(X_1, X_2, .., X_N | Y=y_k)}{\sum_j P(Y=y_j) P(X_1, X_2, .., X_N | Y=y_j)}$$

$$= \frac{P(Y=y_k) \Pi_i P(X_i | Y=y_k)}{\sum_j P(Y=y_j) \Pi_i P(X_i | Y=y_j)}$$

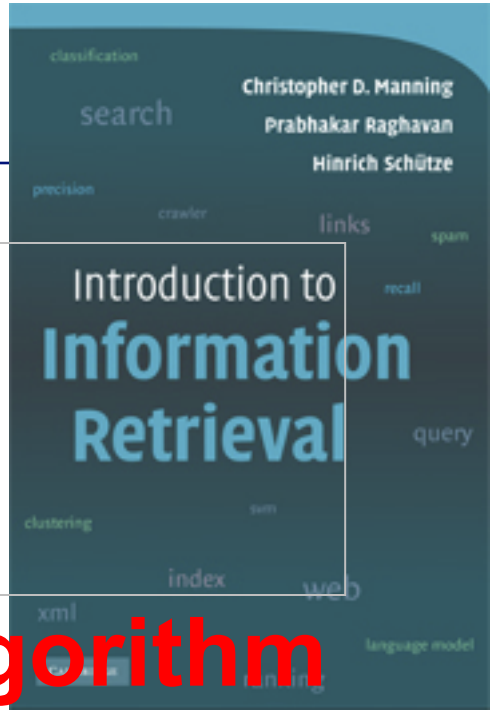
$$Y \leftarrow argmax_{y_k} P(Y = y_k) \Pi_i P(X_i | Y = y_k)$$

Naïve Bayes

- A generative, parametric model
- Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using Bayes rule.
- The standard naive Bayes classifier (at least the R implementation) assumes independence of the predictor variables, and Gaussian distribution (given the target class) of metric/continuous predictors.
- For attributes with missing values, the corresponding table entries are omitted for prediction.

Naïve Bayes and Conditional Independence

- **Make a conditional independence assumption; this leads to a Naïve Bayes classifier**
  - Reduces the number of parameters from $2^{n-1} * 2$ parameters to 2n
- ***Definition: Given random variables X;Y and Z, we say X is conditionally* independent of *Y given Z, if and only if the probability* distribution governing *X is independent of the value of Y given Z;***
  - $(\forall i; j; k)\ P(X = x_i \mid Y = y_j, Z = z_k) = P(X = x_i \mid Z = z_k)$



The Naïve Bayes algorithm

Some slides Adapted from Lectures by

Prabhakar Raghavan (Yahoo and Stanford) and Christopher Manning (Stanford)

Reading material

- **PDF and HTML versions of Chapter 13 are available here**
  - [Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008.] http://nlp.stanford.edu/IR-book/
  - http://www.cs.cmu.edu/~epxing/Class/10701-10s/Lecture/lecture5.pdf
  - http://www.cs.cmu.edu/~tom/10701_sp11/lectures.shtml

Spam filtering: Another text classification task

From: "" <takworlld@hotmail.com>
Subject: real estate is the only way... gem oalvgkay

Anyone can buy real estate with no money down

Stop paying rent TODAY !

There is no need to spend hundreds or even thousands for similar courses

I am 22 years old and I have already purchased 6 properties using the methods outlined in this truly INCREDIBLE ebook.

Change your life NOW !

==========
Click Below to order:
http://www.wholesaledaily.com/sales/nmd.htm
==========

Text classification: Naïve Bayes Text Classification

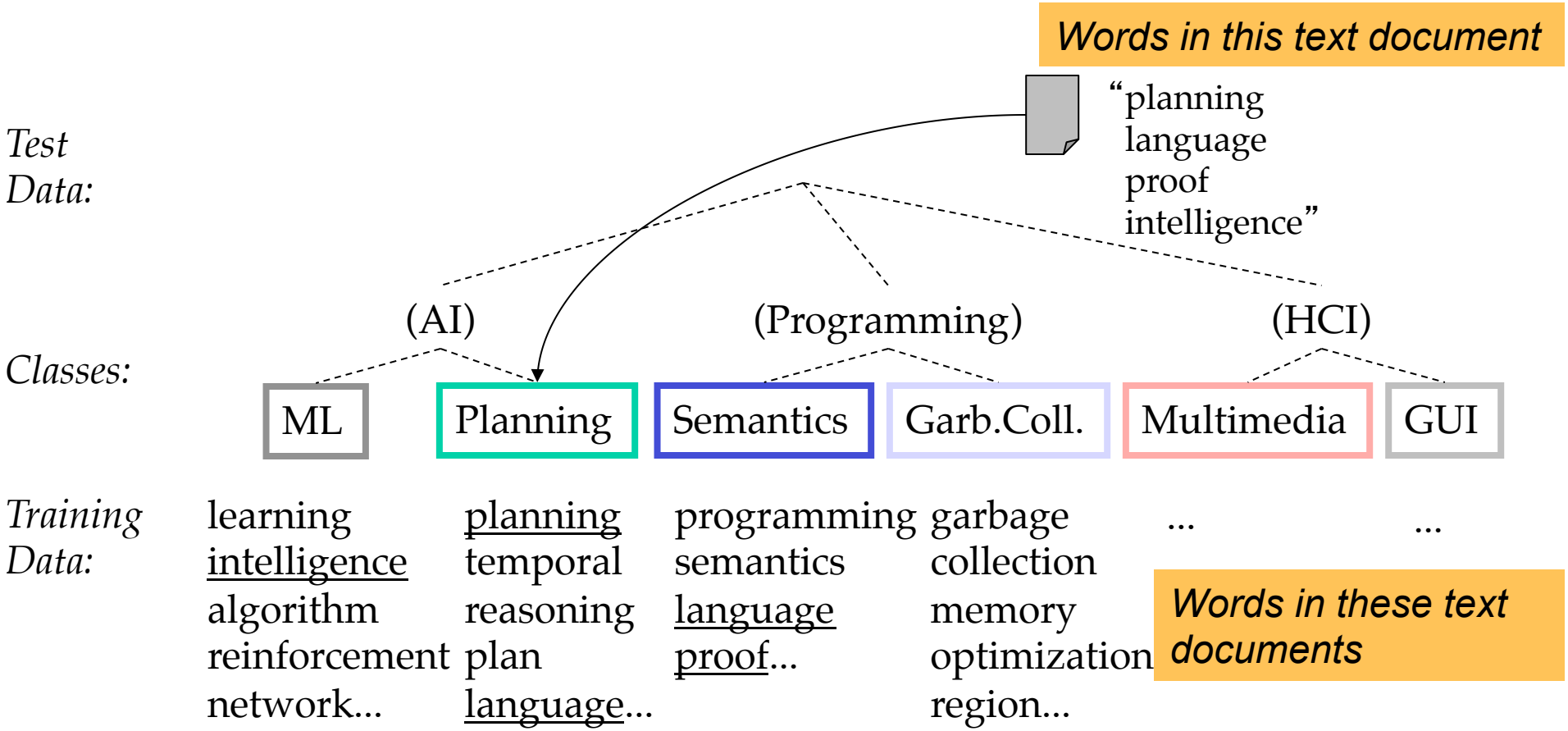
- **Today:**
  - Introduction to Text Classification
    - Also widely known as "text categorization".
  - Probabilistic Language Models
  - Naïve Bayes text classification
    - Multinomial
    - Bernoulli
  - Feature Selection

Categorization/Classification

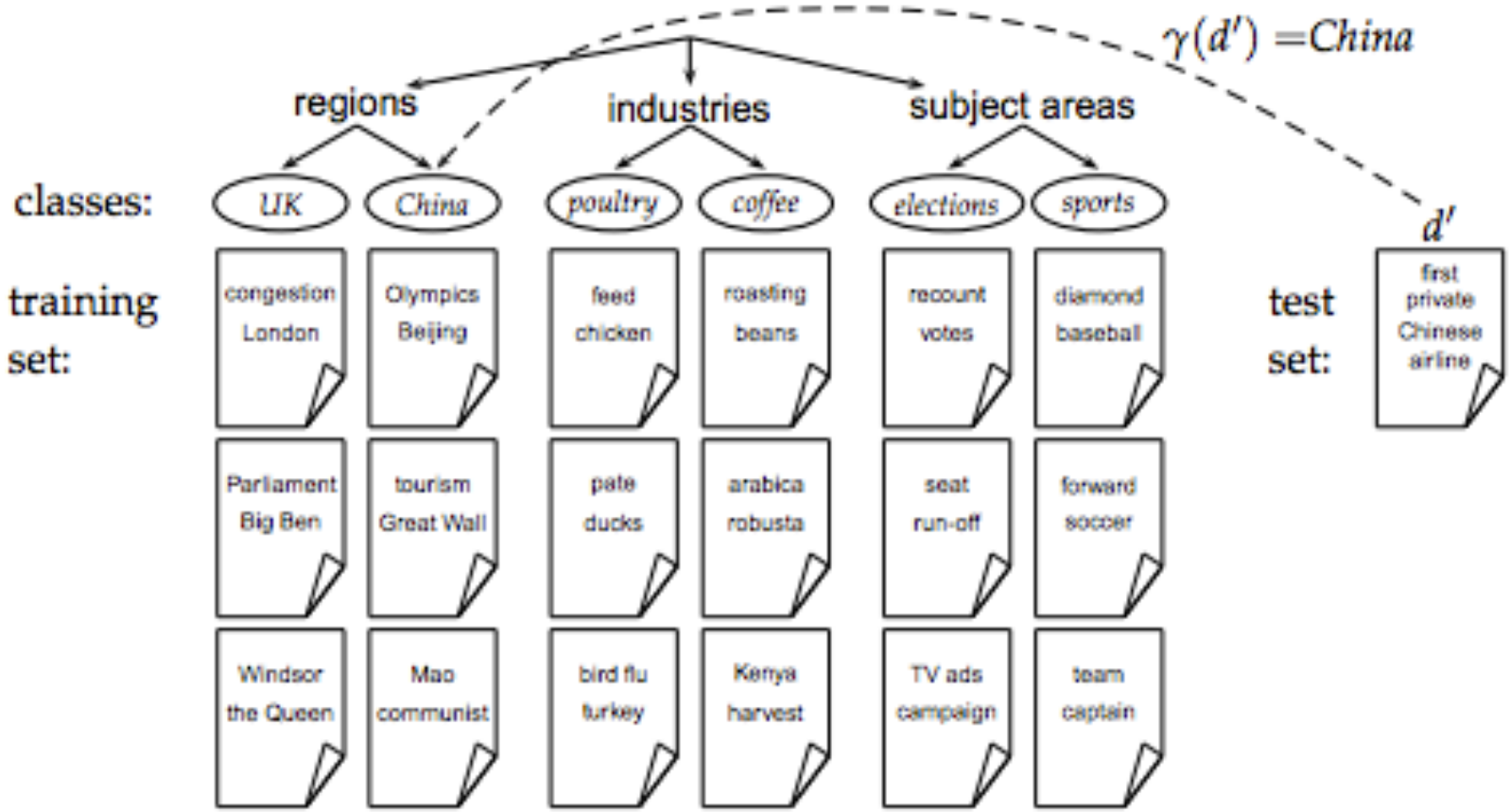
- **Given:**
  - A description of an instance, $x \in X$, where X is the *instance language* or *instance space*.
    - *Issue*: how to represent text documents.
  - A fixed set of classes:

    $C = \{c_1, c_2, \ldots, c_J\}$
- **Determine:**
  - The category of x: $c(x) \in C$, where $c(x)$ is a *classification function* whose domain is X and whose range is C.
    - We want to know how to build classification functions ("classifiers").

Document Classification



(Note: in real life there is often a hierarchy, not present in the above problem statement; and also, you get papers on ML approaches to Garb. Coll.)



► **Figure 13.1** Classes, training set, and test set in text classification .

More Text Classification Examples:

Many search engine functionalities use classification

Assign labels to each document or web-page:

- **Labels are most often topics such as Yahoo-categories**
  e.g., "finance," "sports," "news>world>asia>business"
- **Labels may be genres**
  e.g., "editorials" "movie-reviews" "news"
- **Labels may be opinion on a person/product**
  e.g., "like", "hate", "neutral"
- **Labels may be domain-specific**
  e.g., "interesting-to-me" : "not-interesting-to-me"
  e.g., "contains adult language" : "doesn't"
  e.g., language identification: English, French, Chinese, …
  e.g., search vertical: about Linux versus not
  e.g., "link spam" : "not link spam"

Classification Methods (1)

- **Manual classification**
  - Used by Yahoo! (originally; now downplayed), Looksmart, about.com, ODP, PubMed
  - Very *accurate* when job is done by experts
  - *Consistent* when the problem size and team is small
  - *Difficult and expensive to scale*
    - Means we need automatic classification methods for big problems

Classification Methods (2)

- **Automatic document classification**
  - Hand-coded rule-based systems
    - Used by CS dept's spam filter, Reuters, CIA, etc.
    - Companies (Verity) provide "IDE" for writing such rules
      - E.g., assign category if document contains a given boolean combination of words
    - Standing queries: Commercial systems have complex query languages (everything in IR query languages + accumulators)
      - Accuracy is often very high if a rule has been carefully refined over time by a subject expert
    - Building and maintaining these rules is expensive

Classification Methods (3)

- **Supervised learning of a document-label assignment function**
  - Many systems partly rely on machine learning (Autonomy, MSN, Verity, Enkata, Yahoo!, …)
    - k-Nearest Neighbors (simple, powerful)
    - Naive Bayes (simple, common method)
    - Support-vector machines (new, more powerful)
    - … plus many other methods
    - No free lunch: requires hand-classified training data
    - But data can be built up (and refined) by amateurs
- **Note that many commercial systems use a mixture of methods**

Recall a few probability basics

- **For events *a* and *b*:**
- **Bayes' Rule**

Pr(Spade) = ¼
Pr(King) = 1/13
Pr(Spade) and Pr(King) = 1/52
Pr(king|spade) = 1/13
Pr(king|spade) = Pr(Spade) and Pr(King) / Pr(Spac
Pr(spade|King = Pr(Spade) and Pr(King) / Pr(King

$$p(a,b) = p(a \cap b) = p(a \mid b)p(b) = p(b \mid a)p(a)$$

$$p(\bar{a} \mid b)p(b) = p(b \mid \bar{a})p(\bar{a})$$

$$p(a \mid b) = \frac{p(b \mid a)p(a)}{p(b)} = \frac{p(b \mid a)p(a)}{\sum_{x=a,\bar{a}} p(b \mid x)p(x)}$$

Prior

Posterior

- **Odds:**

$$O(a) = \frac{p(a)}{p(\bar{a})} = \frac{p(a)}{1 - p(a)}$$

Bayesian Methods

- **Learning and classification methods based on probability theory.**
  - Bayes theorem plays a critical role in probabilistic learning and classification.
- **Build a *generative model* that approximates how data is produced.**
- **Uses *prior* probability of each category given no information about an item.**
- **Categorization produces a *posterior* probability distribution over the possible categories given a description of an item (and prior probabilities).**

Bayes' Rule

Pr(Spade) = ¼
Pr(King) = 1/13
Pr(Spade) and Pr(King) = 1/52
Pr(king|spade) = 1/13
Pr(king|spade) = Pr(Spade) and Pr(King) / Pr(Spac
Pr(spade|King = Pr(Spade) and Pr(King) / Pr(King

Theorem of total probabilites
P(D) =P(D|C=TRUE) + P(D|C=FALSE)

$$P(C,D) = P(C \mid D)P(D) = P(D \mid C)P(C)$$

$$P(C \mid D) = \frac{P(D \mid C)P(C)}{P(D)}$$

TASK P(class=china|{T, C}) vs P(class=not china|{T, C})

Naive Bayes Classifiers

***Task*: Classify a new instance *D* based on a tuple of attribute values $D = \langle x_1, x_2, \ldots, x_n \rangle$ into one of the classes $c_j \in C$**

$$c_{MAP} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j \mid x_1, x_2, \ldots, x_n)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} \frac{P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)}{P(x_1, x_2, \ldots, x_n)}$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c_j) P(c_j)$$

MAP = Maximum Aposteriori Probability

Naïve Bayes Classifier: Naïve Bayes Assumption

- $P(c_j)$
  - Can be estimated from the frequency of classes in the training examples.
- $P(x_1,x_2,\ldots,x_n|c_j)$
  - $O(|X|^n \bullet |C|)$ parameters
  - Could only be estimated if a very, very large number of training examples was available.

Naïve Bayes Conditional Independence Assumption:

- **Assume that the probability of observing the conjunction of attributes is equal to the product of the individual probabilities $P(x_i|c_j)$.**

Naïve Bayes

- **Two different ways to set up a Naïve Bayes classifier**
- **Different type of input variables (X_i):**
  - Multivariate Bernoulli or Bernoulli Naïve Bayes Model

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$\hat{P}(\text{Chinese}|c) = (3+1)/(3+2) = 4/5$$

Count 3 documents with Chinese for YES class

  - Multinomial Naïve Bayes

Parameter estimation for Naïve Bayes

- **Multivariate Bernoulli model:** X_w = t if word present document

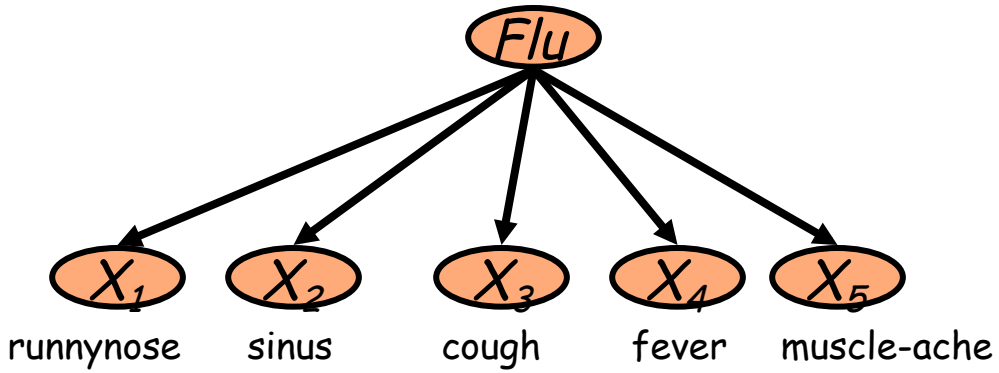
$$\hat{P}(X_w = t \mid c_j) = \text{fraction of documents of topic } c_j \text{ in which word } w \text{ appears}$$

- **Multinomial model:**

$$\hat{P}(X_i = w \mid c_j) = \text{fraction of times in which word } w \text{ appears across all documents of topic } c_j$$

  - Can create a mega-document for topic *j* by concatenating all documents on this topic
  - Use frequency of *w* in mega-document

The Naïve Bayes Classifier

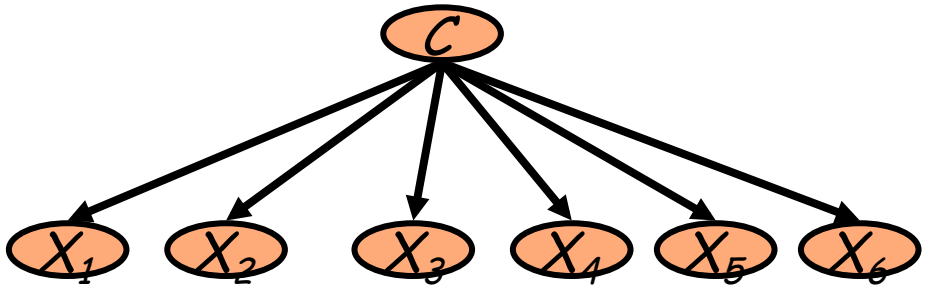


- **Conditional Independence Assumption: Features (term presence) are *independent* of each other given the class:**

$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- **This model is appropriate for binary variables**
  - Multivariate Bernoulli model

Learning the Model

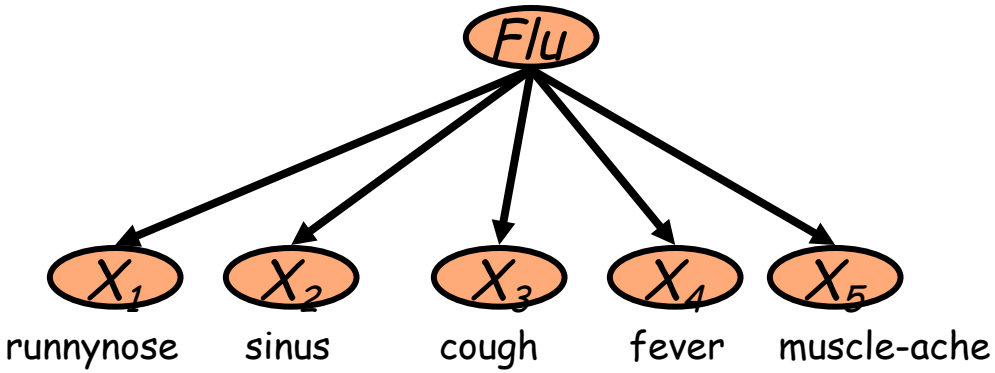


- ***First attempt*: maximum likelihood estimates**
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{N(C = c_j)}{N}$$

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j)}{N(C = c_j)}$$

Problem with Max Likelihood



$$P(X_1,\ldots,X_5 \mid C) = P(X_1 \mid C) \bullet P(X_2 \mid C) \bullet \cdots \bullet P(X_5 \mid C)$$

- **What if we have seen no training cases where patient had no flu and muscle aches?**

$$\hat{P}(X_5 = t \mid C = nf) = \frac{N(X_5 = t, C = nf)}{N(C = nf)} = 0$$

- **Zero probabilities cannot be conditioned away, no matter the other evidence!**

$$\ell = \arg\max_c \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Bernoulli model: Smoothing to Avoid Overfitting and Zero probabilities

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

# of values of X_i

Laplace smoothing for Bernoulli model
k is number of classes here

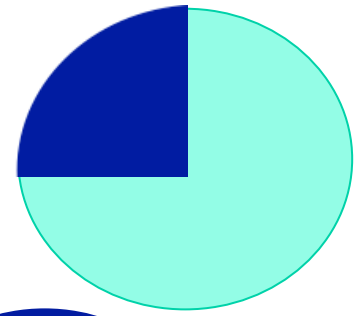
Multinomial model: Smoothing to Avoid Overfitting and Zero probabilities

1000 words in the Hadoop vocab and 10^9 in English

$$\hat{P}(x_i \mid c_j) = \frac{N(X_i = x_i, C = c_j) + 1}{N(C = c_j) + k}$$

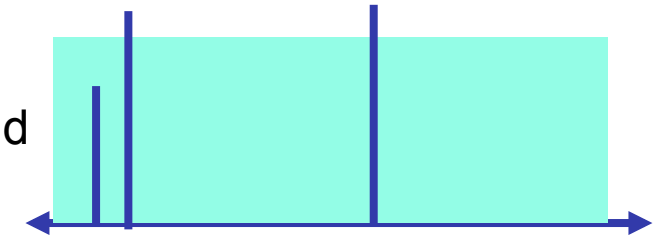
+0.0001

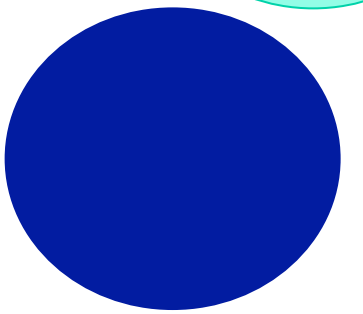
# of values of X_i



Laplace smoothing for Bernoulli model
k is vocabulary size

Cyan:Background
Blue: foreground





Classification

- **Multinomial vs Multivariate Bernoulli?**
- **Multinomial model is almost always more effective in text applications!**
- **See *IR Book* sections 13.2 and 13.3 for worked examples with each model**

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Vocabulary = {Chinese, Beijing, Shanghai, Macao, Tokyo}

Example 13.2: Applying the Bernoulli model to the example in Table 13.1, we have the same estimates for the priors as before: $\hat{P}(c) = 3/4$, $\hat{P}(\overline{c}) = 1/4$. The conditional probabilities are:

Japan does NOT occur class C

$$\begin{aligned}
\hat{P}(\text{Chinese}|c) &= (3+1)/(3+2) = 4/5\\
\hat{P}(\text{Japan}|c) = \hat{P}(\text{Tokyo}|c) &= (0+1)/(3+2) = 1/5\\
\hat{P}(\text{Beijing}|c) = \hat{P}(\text{Macao}|c) = \hat{P}(\text{Shanghai}|c) &= (1+1)/(3+2) = 2/5\\
\hat{P}(\text{Chinese}|\overline{c}) &= (1+1)/(1+2) = 2/3\\
\hat{P}(\text{Japan}|\overline{c}) = \hat{P}(\text{Tokyo}|\overline{c}) &= (1+1)/(1+2) = 2/3\\
\hat{P}(\text{Beijing}|\overline{c}) = \hat{P}(\text{Macao}|\overline{c}) = \hat{P}(\text{Shanghai}|\overline{c}) &= (0+1)/(1+2) = 1/3
\end{aligned}$$

The denominators are $(3+2)$ and $(1+2)$ because there are three documents in c and one document in $\overline{c}$ and because the constant B in Equation (13.7) is 2 – there are two cases to consider for each term, occurrence and nonoccurrence.

The scores of the test document for the two classes are

$$\begin{aligned}
\hat{P}(c|d_5) &\propto \hat{P}(c) \cdot \hat{P}(\text{Chinese}|c) \cdot \hat{P}(\text{Japan}|c) \cdot \hat{P}(\text{Tokyo}|c)\\
&\quad \cdot (1-\hat{P}(\text{Beijing}|c)) \cdot (1-\hat{P}(\text{Shanghai}|c)) \cdot (1-\hat{P}(\text{Macao}|c))\\
&= 3/4 \cdot 4/5 \cdot 1/5 \cdot 1/5 \cdot (1-2/5) \cdot (1-2/5) \cdot (1-2/5)\\
&\approx 0.005
\end{aligned}$$

and, analogously,

$$\begin{aligned}
\hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot 2/3 \cdot 2/3 \cdot 2/3 \cdot (1-1/3) \cdot (1-1/3) \cdot (1-1/3)\\
&\approx 0.022
\end{aligned}$$

Thus, the classifier assigns the test document to $\overline{c}$ = *not-China*. When looking only at binary occurrence and not at term frequency, Japan and Tokyo are indicators for $\overline{c}$ ($2/3 > 1/5$) and the conditional probabilities of Chinese for c and $\overline{c}$ are not different enough (4/5 vs. 2/3) to affect the classification decision.

Bernoulli NB: Smoothing is based on the number of classes

- 6 terms in Vocabulary 6 terms in calculation of P(C|X)
- Count single occurrence of term, .e.g., Chinese is just counted once

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Decision Rule for Bernoulli Model

Use all vocabulary for classification (regardless if the word occurred or not in the test example)

► **Table 13.3** Multinomial versus Bernoulli model.

	multinomial model	Bernoulli model
event model	generation of token	generation of document
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
document representation	$d = \langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle, t_k \in V$	$d = \langle e_1, \ldots, e_i, \ldots, e_M \rangle,$ $e_i \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t\|c)$	$\hat{P}(U_i = e\|c)$
decision rule: maximize	$\hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(X = t_k\|c)$	$\hat{P}(c) \prod_{t_i \in V} \hat{P}(U_i = e_i\|c)$
multiple occurrences	taken into account	ignored
length of docs	can handle longer docs	works best for short docs
# features	can handle more	works best with fewer
estimate for term the	$\hat{P}(X = \text{the}\|c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1\|c) \approx 1.0$

- **Multinomial Naïve Bayes**

Stochastic Language Models

- **Models *probability* of generating strings (each word in turn) in the language (commonly all strings over ∑). E.g., unigram model**

Model M

0.2	the
0.1	a
0.01	man
0.01	woman
0.03	said
0.02	likes
...	

the	man	likes	the	woman
0.2	0.01	0.02	0.2	0.01

multiply

P(s | M) = 0.00000008

Stochastic Language Models

- **Model *probability* of generating any string**

Model M1		Model M2	
0.2	the	0.2	the
0.01	class	0.0001	class
0.0001	sayst	0.03	sayst
0.0001	pleaseth	0.02	pleaseth
0.0001	yon	0.1	yon
0.0005	maiden	0.01	maiden
0.01	woman	0.0001	woman

the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$$P(s|M2) > P(s|M1)$$

Unigram and higher-order models

- P (● ● ● ●)

  = P (●) P (● | ●) P (● | ● ●) P (● | ● ● ●)

- **Unigram Language Models**

  P (●) P (●) P (●) P (●)

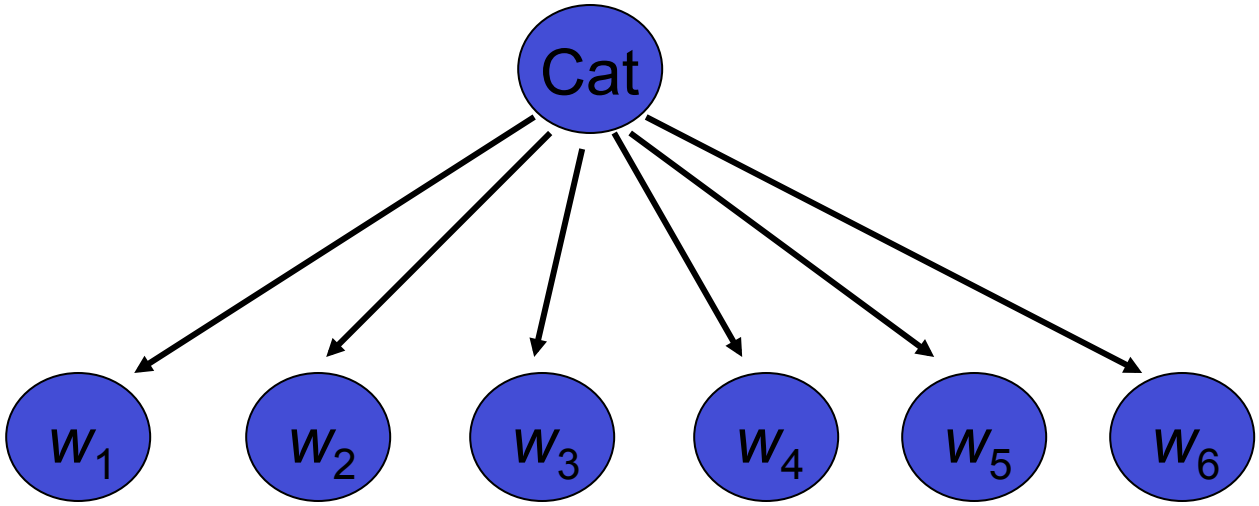
  Easy. Effective!

- **Bigram (generally, *n*-gram) Language Models**

  P (●) P (● | ●) P (● | ●) P (● | ●)

- **Other Language Models**
  - Grammar-based models (PCFGs), etc.
    - Probably not the first thing to try in IR

Naïve Bayes via a class conditional language model = multinomial NB



- **Effectively, the probability of each class is done as a class-specific unigram language model**

Using Multinomial Naive Bayes Classifiers to Classify Text: Basic method

- **Attributes are text positions, values are words.**

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_i P(x_i \mid c_j)$$

$$= \underset{c_j \in C}{\operatorname{argmax}} P(c_j) P(x_1 = \text{"our"} \mid c_j) \cdots P(x_n = \text{"text"} \mid c_j)$$

- Still too many possibilities
- Assume that classification is *independent* of the positions of the words
  - Use same parameters for each position
  - Result is bag of words model (over tokens not types)

Naïve Bayes: Learning Algorithm

- **From training corpus, extract *Vocabulary***
- **Calculate required $P(c_j)$ and $P(x_k \mid c_j)$ terms**
  - For each c_j in C do
    - $docs_j \leftarrow$ subset of documents for which the target class is c_j
    - $$P(c_j) \leftarrow \frac{|docs_j|}{|\text{total \# documents}|}$$

- $Text_j \leftarrow$ single document containing all $docs_j$
- for each word x_k in *Vocabulary*
  - $n_k \leftarrow$ number of occurrences of x_k in $Text_j$
  - $$P(x_k \mid c_j) \leftarrow \frac{n_k + \alpha}{n + \alpha |Vocabulary|}$$

n = total number of tokens
(alpha is "tuning/smoothing" factor that can be set to 1)
Multinomial Conditional Independence Naïve Bayes: Learning from Training Set

Naïve Bayes: Classifying

- positions ← all word positions in current document which contain tokens found in *Vocabulary*
- Return c_{NB}, where

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Naïve Bayes: Classifying

- **positions ← all word positions in current document which contain tokens found in *Vocabulary***
- **Return c_{NB}, where**

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

Classify a NEW document D with 100 words (not seen in the class=TRUE)
P(W|Class=TRUE) = $1/(10^6 + 10^7)$ ##default probability of a word given *Class=TRUE*

Assume a Vocabulary of 10^6 words; number of words in *class=TRUE* is 10^7
THEN
Pr(Class = TRUE | D) ~ $(1/(10^6 + 10^7))^{100}$ X Prior

Underflow Prevention: log space

- **Multiplying lots of probabilities, which are between 0 and 1, can result in floating-point underflow.**
- **Since log(*xy*) = log(*x*) + log(*y*), it is better to perform all computations *by summing logs of probabilities rather than multiplying probabilities*.**
- **Class with highest final un-normalized log probability score is still the most probable.**

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j) \quad \#\#\text{Log}(\frac{a}{b}) = \log(a) - \log(b); Note: \log(1) = 0$$

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

- **Note that model is now just max of sum of weights…**

Log Rules Refresher

$$\log_a(bc) = \log_a(b) + \log_a(c)$$

$$\log_a(b^c) = c\log_a(b)$$

$$\log_a(1/b) = -\log_a(b)$$

$$\log_a(1) = 0$$

$$\log_a(a) = 1$$

$$\log_a(a^r) = r$$

$$\log_{1/a}(b) = -\log_a(b)$$

$$\log_a(b)\log_b(c) = \log_a(c)$$

$$\log_b(a) = \frac{1}{\log_a(b)}$$

$$\log_{a^m}(a^n) = \frac{n}{m}, \qquad m \neq 0$$

http://grockit.com/blog/act-logarithms-explained/

Naïve Bayes: Classifying: Extreme example!

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

- **positions ← all word positions in current document which contain tokens found in *Vocabulary***
- **Return c_{NB}, where**

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j) \quad \#\#\text{Log}(\frac{a}{b}) = \log(a) - \log(b); Note: \log(1) = 0$$

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

In R
> log (1/(10^6+ 10^7)) *100
[1] -1621.341
> exp(-1621.341)
[1] 0 #super small number; we underflow

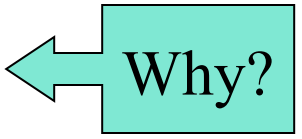
Classify a NEW document D with 100 words (all 100 words are not seen in the class=TRUE)
P(W|Class=TRUE) = 1/(10^6+ 10^7) ##default probability of a word given *Class=TRUE*

Assume a Vocabulary of 10^6 words; number of words in *class=TRUE* is 10^7
THEN
Pr(Class = TRUE | D) ~ (1/(10^6+ 10^7))^{100} X Prior

Approximate with log base 10 (assume (10^6+ 10^7) =10^7)
P(W|Class=TRUE) =~ (0 – log(10^7, 10)) X 100 = ~ -7
P(W|Class=TRUE) =~ -7 x 100 #in log space

Naive Bayes: Time Complexity

- **Training Time: $O(|D|L_d + |C||V|))$ where L_d is the average length of a document in D.**
  - Assumes V and all D_i, n_i, and n_{ij} pre-computed in $O(|D|L_d)$ time during one pass through all of the data.
  - Generally just $O(|D|L_d)$ since usually $|C||V| < |D|L_d$



- **Test Time: $O(|C|\ L_t)$ where L_t is the average length of a test document.**
  - Very efficient overall, linearly proportional to the time needed to just read in all the data.
  - Plus, robust in practice

- **Exercise**

Exercise: Multinomial Naive Bayes

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$P(C \mid D) = \frac{P(D \mid C)P(C)}{P(D)}$$

- Estimate parameters of Multinomial Naive Bayes classifier
- Classify test document

Exercise: Multinomial Naive Bayes

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Priors Pr(Class=China) = ¾ Pr(Class=Not China) = ¼
Class conditional probabilities; likelihoods
Pr(Chinese | Class=China) = 5/8
Pr(Chinese | Class=NOT China) = 1/3
What is the class of this document {Tokyo,chinese}
Pr(Class = China|{Tokyo,chinese}) = ¾ X 0/8 X 5/8 = 0
Pr(Class = Not China|{Tokyo,chinese}) = ¼ X 1/3 X 1/3 = 1/36
Pr(Class = China|{Tokyo,chinese})=0/ 0+ 1/36 = 0
Pr(Class = Not China|{Tokyo,chinese}) = 1/36/ 0+ 1/36 = 1

$$P(C \mid D) = \frac{P(D \mid C)P(C)}{P(D)}$$

Pr(Class = China|{Tokyo,chinese}) = ¾ X 0+ 1/8+6 X 5+1/8+6 = 0.023
Pr(Class = Not China|{Tokyo,chinese}) = ¼ X 1+1/3+6 X 1+1/3+6 = 0.012
Pr(Class = China|{Tokyo,chinese})=0.023 / (0.023 + 0.012) = 0.65
Pr(Class = Not China|{Tokyo,chinese}) = 1 – 0.65

Exercise: Multinomial Naive Bayes

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Pr(Class=China) = ¾ Pr(Class = not China) = ¼

$$P(C \mid D) = \frac{P(D \mid C)P(C)}{P(D)}$$

Pr(Chinese|Class = China) = 5 / 8, 1/8, 1/8, 1/8 =1
TEST = {Japan, Chinese}
Pr(class =China| {Japan, Chinese}) = Prior X Likelihood = ¾ X (5/14 X 1/14) = 0.022; PC|D) = 0.022/(0.022+0.0123)= 0.64
Pr(class = NOT China| {Japan, Chinese}) = Prior X Likelihood = 1/4 X (**1+1/(3+6)** X 1+1/(3+6) = 0.0123 = ; P(C|D) = 1 - 0.64

6 unique
Pr(Chinese|Class = China) = 5+1/ (8+6)

- Estimate parameters of Multinomial Naive Bayes classifier
- Classify test document

Example: Parameter estimates

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ Conditional probabilities:

$$\begin{aligned}
\hat{P}(\text{CHINESE}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{TOKYO}|c) = \hat{P}(\text{JAPAN}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{CHINESE}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{TOKYO}|\overline{c}) = \hat{P}(\text{JAPAN}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

The denominators are (8 + 6) and (3 + 6) because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B is 6 as the vocabulary consists of six terms.

Example: Classification of d_5

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

$$\hat{P}(c|d_5) \propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003$$

$$\hat{P}(\overline{c}|d_5) \propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001$$

Thus, the classifier assigns the test document to c = *China*. The reason for this classification decision is that the three occurrences of the positive indicator CHINESE in d_5 outweigh the occurrences of the two negative indicators JAPAN and TOKYO.

Note: Two Models: Multivariate Bernoulli

- ***Model 1*: Multivariate Bernoulli**
  - One feature X_w for each word in dictionary
  - X_w = true in document d if w appears in d
  - Naive Bayes assumption:
    - Given the document's topic, appearance of one word in the document tells us nothing about chances that another word appears
- **This is the model used in the binary independence model in classic probabilistic relevance feedback in hand-classified data**

Two Models: Multinomial NB

- ***Model 2*: Multinomial = Class conditional unigram**
  - One feature X_i for each word pos in document
    - feature's values are all words in dictionary
  - Value of X_i is the word in position *i*
  - Naïve Bayes assumption:
    - Given the document's topic, word in one position in the document tells us nothing about words in other positions
  - Second assumption:
    - Word appearance does not depend on position

$$P(X_i = w \mid c) = P(X_j = w \mid c)$$

for all positions *i,j*, word *w*, and class *c*

- **Puzzle**

Example: Parameter estimates

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

taiwan

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ Conditional probabilities:

$$\begin{aligned}
\hat{P}(\text{CHINESE}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\
\hat{P}(\text{TOKYO}|c) = \hat{P}(\text{JAPAN}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\text{CHINESE}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\text{TOKYO}|\overline{c}) = \hat{P}(\text{JAPAN}|\overline{c}) &= (1+1)/(3+6) = 2/9
\end{aligned}$$

The denominators are (8 + 6) and (3 + 6) because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B is 6 as the vocabulary consists of six terms.

Example: Parameter estimates

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ Condition[illegible]ties:

$$\hat{P}(\text{CHINESE}|c) = [illegible]/(8+6) = 6/14 = 3/7$$

$$\hat{P}(\text{TOKYO}|c) = \hat{P}(\text{JAP}[illegible]) = (0+1)/(8+6) = 1/14$$

$$\hat{P}([illegible]|c) = (1+1)/(3+6) = 2/9$$

$$\hat{P}(\text{TO}[illegible]\ (\text{JAPAN}|\overline{c}) = (1+1)/(3+6) = 2/9$$

Which type of Naïve Bayes model is this?

The denominators are (8 + 6) and (3 + 6) because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B is 6 as the vocabulary consists of six terms.

Example: Parameter estimates

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Priors: $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ Conditional probabilities:

$$\hat{P}(\text{CHINESE}|c) = (5+1)/(8+6) = 6/14 = 3/7$$

Which type of Naïve Bayes model is this?

Which type of Naïve Bayes model is this?
Multinomial….look at *Pr(Chinese|c)* is made up of 5 occurences of the word in the class c (versus 3 in the Bernoulli model for the same class conditional, *Pr(Chinese|c)*)

Underflow Prevention: log space

- **Multiplying lots of probabilities, which are between 0 and 1, can result in floating-point underflow.**
- **Since log(*xy*) = log(*x*) + log(*y*), it is better to perform all computations *by summing logs of probabilities rather than multiplying probabilities*.**
- **Class with highest final un-normalized log probability score is still the most probable.**

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} \log P(c_j) + \sum_{i \in positions} \log P(x_i \mid c_j)$$

- **Note that model is now just max of sum of weights…**

Multinomial vs Bernoulli Naïve Bayes

► **Table 13.3** Multinomial versus Bernoulli model.

	multinomial model	Bernoulli model
event model	generation of token	generation of document
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
document representation	$d = \langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle, t_k \in V$	$d = \langle e_1, \ldots, e_i, \ldots, e_M \rangle,$ $e_i \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t\|c)$	$\hat{P}(U_i = e\|c)$
decision rule: maximize	$\hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(X = t_k\|c)$	$\hat{P}(c) \prod_{t_i \in V} \hat{P}(U_i = e_i\|c)$
multiple occurrences	taken into account	ignored
length of docs	can handle longer docs	works best for short docs
# features	can handle more	works best with fewer
estimate for term the	$\hat{P}(X = \text{the}\|c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1\|c) \approx 1.0$

Multinomial Naïve Bayes for text

- ..

$$P(Y = y_k | X_1, X_2, ..., X_N) = \frac{P(Y=y_k) P(X_1, X_2, .., X_N | Y=y_k)}{\sum_j P(Y=y_j) P(X_1, X_2, .., X_N | Y=y_j)}$$

$$= \frac{P(Y=y_k) \Pi_i P(X_i | Y=y_k)}{\sum_j P(Y=y_j) \Pi_i P(X_i | Y=y_j)}$$

$$Y \leftarrow argmax_{y_k} P(Y = y_k) \Pi_i P(X_i | Y = y_k)$$

Naïve Bayes Classifier for Text

- Given the training data what are the parameters to be estimated?

$P(Y)$ $P(X|Y_1)$ $P(X|Y_2)$

Diabetes : 0.8
Hepatitis : 0.2

the: 0.001
diabetic : 0.02
blood : 0.0015
sugar : 0.02
weight : 0.018
...

the: 0.001
diabetic : 0.0001
water : 0.0118
fever : 0.01
weight : 0.008
...

13.2 Naive Bayes text classification

MULTINOMIAL NAIVE BAYES

The first supervised learning method we introduce is the *multinomial Naive Bayes* or *multinomial NB* model, a probabilistic learning method. The probability of a document d being in class c is computed as

$$P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c) \tag{13.2}$$

where $P(t_k|c)$ is the conditional probability of term t_k occurring in a document of class c.[1] We interpret $P(t_k|c)$ as a measure of how much evidence t_k contributes that c is the correct class. $P(c)$ is the prior probability of a document occurring in class c. If a document's terms do not provide clear evidence for one class versus another, we choose the one that has a higher prior probability. $\langle t_1, t_2, \ldots, t_{n_d} \rangle$ are the tokens in d that are part of the vocabulary we use for classification and n_d is the number of such tokens in d. For example, $\langle t_1, t_2, \ldots, t_{n_d} \rangle$ for the one-sentence document *Beijing and Taipei join the WTO* might be ⟨Beijing, Taipei, join, WTO⟩, with $n_d = 4$, if we treat the terms and and the as stop words.

In text classification, our goal is to find the *best* class for the document. The best class in NB classification is the most likely or *maximum a posteriori* (MAP) class c_{map}:

MAXIMUM A POSTERIORI CLASS

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(t_k|c). \tag{13.3}$$

http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf

We write $\hat{P}$ for P because we do not know the true values of the parameters $P(c)$ and $P(t_k|c)$, but estimate them from the training set as we will see in a moment.

In Equation (13.3), many conditional probabilities are multiplied, one for each position $1 \le k \le n_d$. This can result in a floating point underflow. It is therefore better to perform the computation by adding logarithms of probabilities instead of multiplying probabilities. The class with the highest log probability score is still the most probable; $\log(xy) = \log(x) + \log(y)$ and the logarithm function is monotonic. Hence, the maximization that is

- ..

ADD-ONE SMOOTHING

To eliminate zeros, we use *add-one* or *Laplace smoothing*, which simply adds one to each count (cf. Section 11.3.2):

$$\hat{P}(t|c) = \frac{T_{ct}+1}{\sum_{t' \in V}(T_{ct'}+1)} = \frac{T_{ct}+1}{(\sum_{t' \in V} T_{ct'}) + B'} \tag{13.7}$$

where $B = |V|$ is the number of terms in the vocabulary. Add-one smoothing can be interpreted as a uniform prior (each term occurs once for each class) that is then updated as evidence from the training data comes in. Note that this is a prior probability for the occurrence of a *term* as opposed to the prior probability of a *class* which we estimate in Equation (13.5) on the document level.

NB example

► **Table 13.1** Data for parameter estimation examples.

	docID	words in document	in $c =$ *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

► **Table 13.2** Training and test times for NB.

mode	time complexity
training	$\Theta(\|\mathbb{D}\|L_{ave} + \|\mathbb{C}\||V|)$
testing	$\Theta(L_a + \|\mathbb{C}\|M_a) = \Theta(\|\mathbb{C}\|M_a)$

We have now introduced all the elements we need for training and applying an NB classifier. The complete algorithm is described in Figure 13.2.

Example 13.1: For the example in Table 13.1, the multinomial parameters we need to classify the test document are the priors $\hat{P}(c) = 3/4$ and $\hat{P}(\overline{c}) = 1/4$ and the following conditional probabilities:

$$\begin{aligned} \hat{P}(\text{Chinese}|c) &= (5+1)/(8+6) = 6/14 = 3/7 \\ \hat{P}(\text{Tokyo}|c) = \hat{P}(\text{Japan}|c) &= (0+1)/(8+6) = 1/14 \\ \hat{P}(\text{Chinese}|\overline{c}) &= (1+1)/(3+6) = 2/9 \\ \hat{P}(\text{Tokyo}|\overline{c}) = \hat{P}(\text{Japan}|\overline{c}) &= (1+1)/(3+6) = 2/9 \end{aligned}$$

The denominators are $(8+6)$ and $(3+6)$ because the lengths of $text_c$ and $text_{\overline{c}}$ are 8 and 3, respectively, and because the constant B in Equation (13.7) is 6 as the vocabulary consists of six terms.

We then get:

$$\begin{aligned} \hat{P}(c|d_5) &\propto 3/4 \cdot (3/7)^3 \cdot 1/14 \cdot 1/14 \approx 0.0003. \\ \hat{P}(\overline{c}|d_5) &\propto 1/4 \cdot (2/9)^3 \cdot 2/9 \cdot 2/9 \approx 0.0001. \end{aligned}$$

Thus, the classifier assigns the test document to $c =$ *China*. The reason for this classification decision is that the three occurrences of the positive indicator Chinese in d_5 outweigh the occurrences of the two negative indicators Japan and Tokyo.

http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW5, HW6, HW7 (due Thursday 8AM of Week 9)
  - MidTerm week 8
  - Bernoulli Naïve Bayes; Bernoulli Mixture Models
  - Async lecture recap plus Q&A
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

- **Week 7**

Unit 7 | Introduction to Graph Algorithms at Scale: Single Shortest Path Algorithm

Hide Contents

- 7.1 Weekly Introduction (2 mins)
- 7.2 Assigned Readings
- 7.3 Networks Introduction and Motivation (5 mins)
- 7.4 Applications Graphs (5 mins)
- 7.5 Graph Definitions (5 mins)
- 7.6 Shortest Path Introduction (4 mins)
- 7.7 Single Source Shortest Path Unweighted Graphs (15 mins)
- 7.8 BFS for Unweighted Graphs Directed and Animations (6 mins)
- 7.9 BFS for Unweighted Graphs Undirected and Animations (7 mins)
- 7.10 SSSP for Weighted Graphs BFS (2 mins)
  - 7.10.1 Quiz: BFS on Weighted Graphs Challenge
- 7.11 SSSP for Weighted Graphs Dijkstras Algorithm (8 mins)
- 7.12 Distributed SSSP Unweighted Graph (10 mins)
- 7.13 Distributed SSSP Unweighted Graph Algorithm (12 mins)
- 7.14 Distributed SSSP Weighted Graph (7 mins)
  - 7.14.1 Quiz: Disktra Vs Depth First

Data Mining & Networks

- Data mining has rich history and methods for analyzing …
  - … tabular data
  - … textual data
  - … time series & streams
  - … market baskets

Bag of features

- What about relations and dependencies?

Networks are useful

- Networks allow for modeling dependencies
- Networks are a general language for describing realworld systems
- Only recently they have become a first class citizen of the machine learning world

Motivation

- Graphs are ubiquitous in modern society: examples encountered by almost everyone on a daily basis include:
- the hyperlink structure of the web (simply known as the web graph)
- social networks (manifest in the flow of email, phone call patterns, connections on social networking sites, etc.),
- and transportation networks (roads, bus routes, flights, etc.).

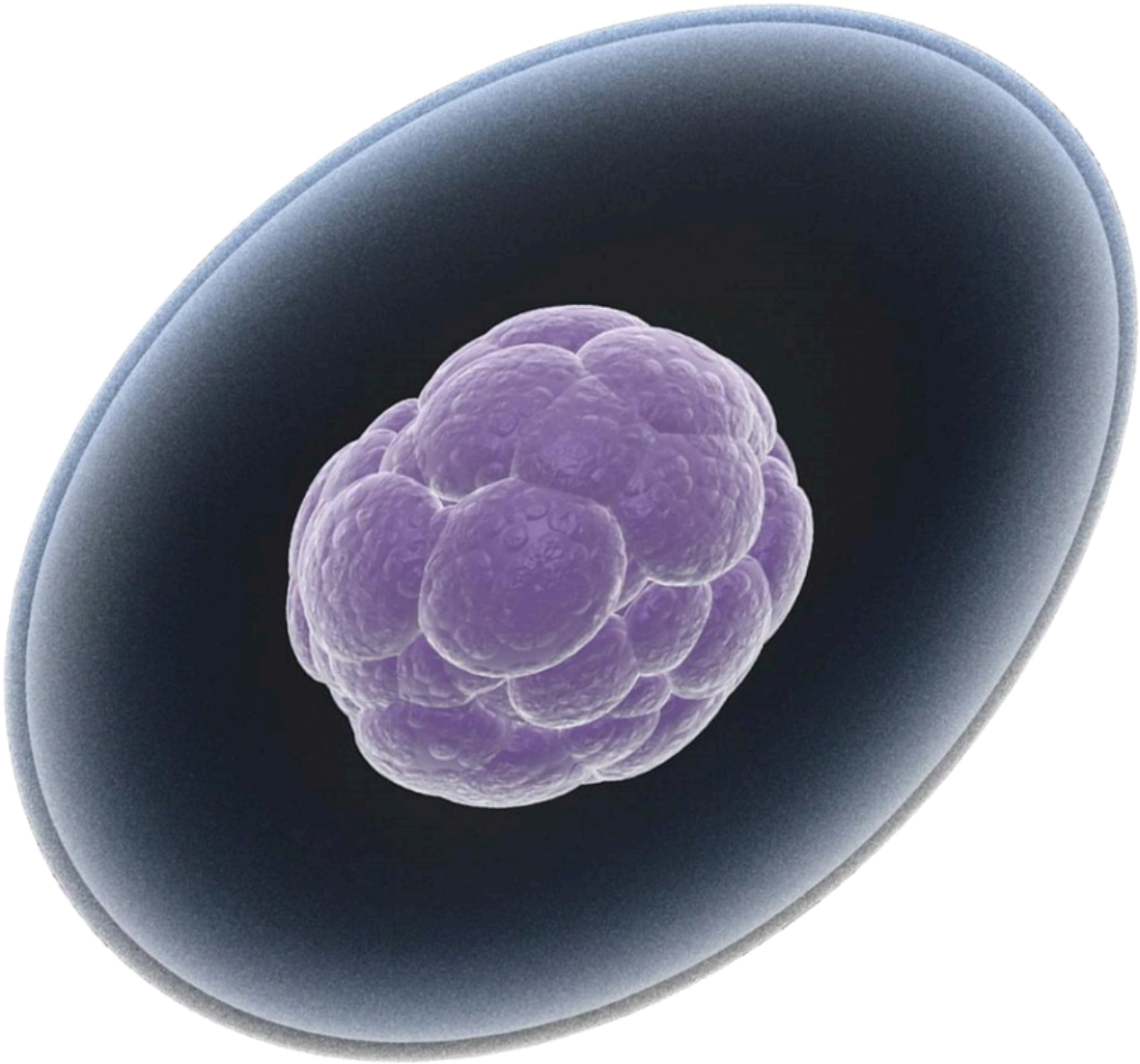
- Our very own existence is dependent on an intricate metabolic and regulatory network, which can be characterized as a large, complex graph involving interactions between genes, proteins, and other cellular products.

- This Lecture focuses on graph algorithms in MapReduce.

Infrastructure


NASDAQ
KB Home
-2.2%
-1.16
50.96
+1.5%
+0.44
30.89
+2.0%
+0.35
17.84
+4.2%
+1.1%
+0.35
32.06
OPLK
BUY ORDERS
19.12
140%
+1.10
19.09
+6.1%
+1.2%
23.46
+0.27
+1.0%
17.15
+0.17
30.89
+0.1%
12.38
+0.01
+3.2%
23.98
+0.74
+2.1%
30.45
+0.64
63.04
+7.9%
+4.62
NASDAQ
NDTV PROFIT

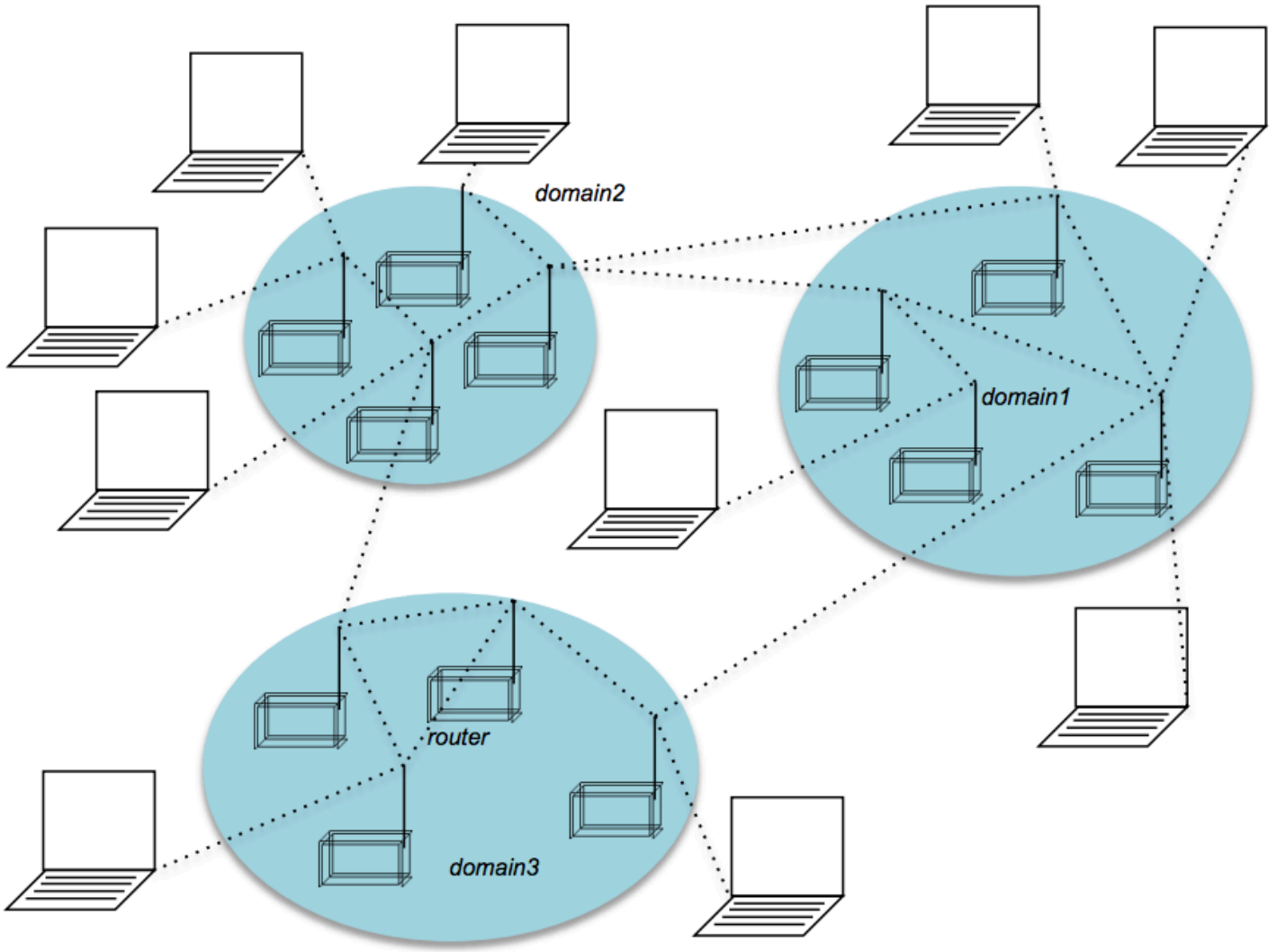
Economy



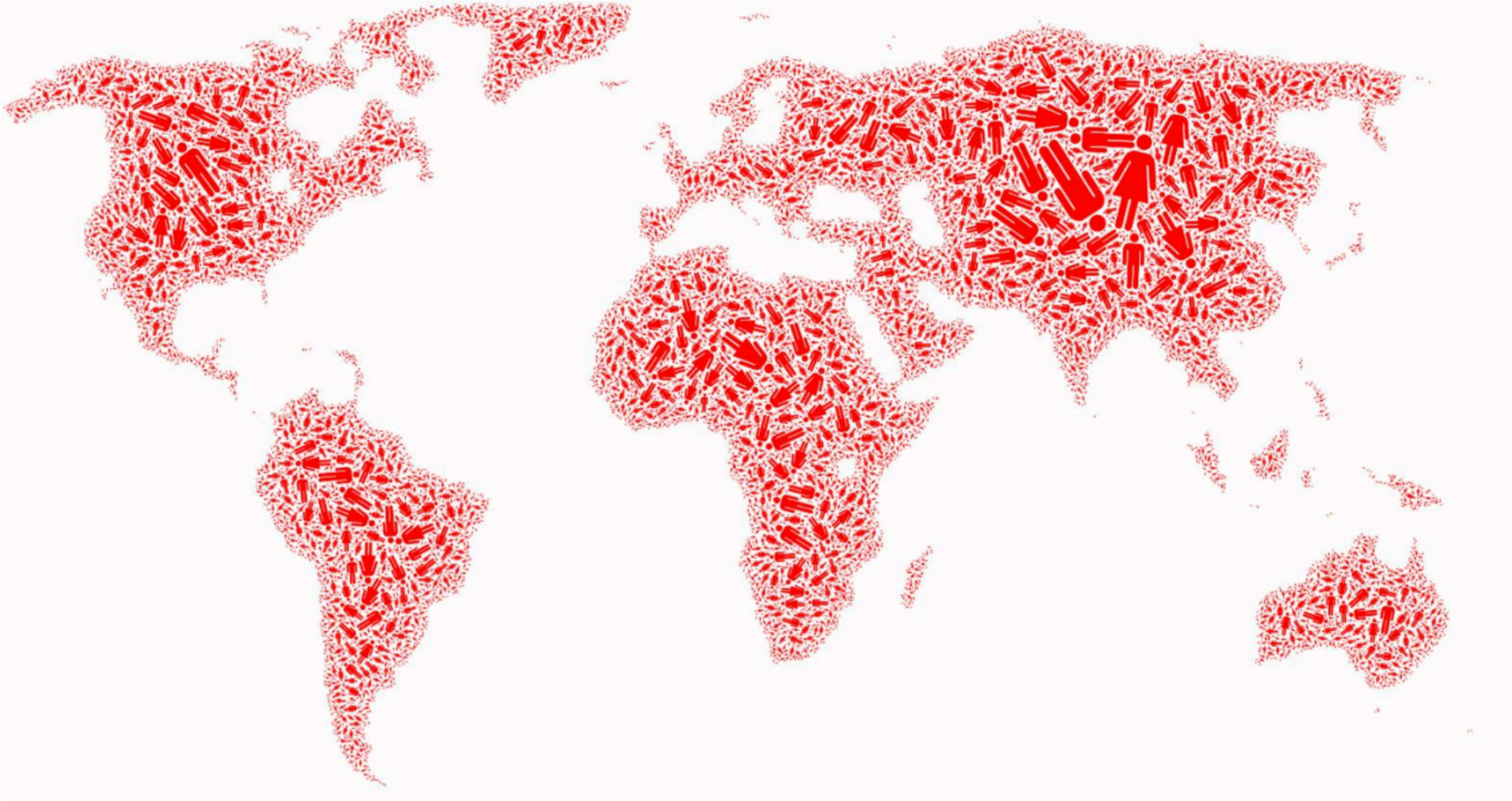
Human cell

Brain

Friends & Family

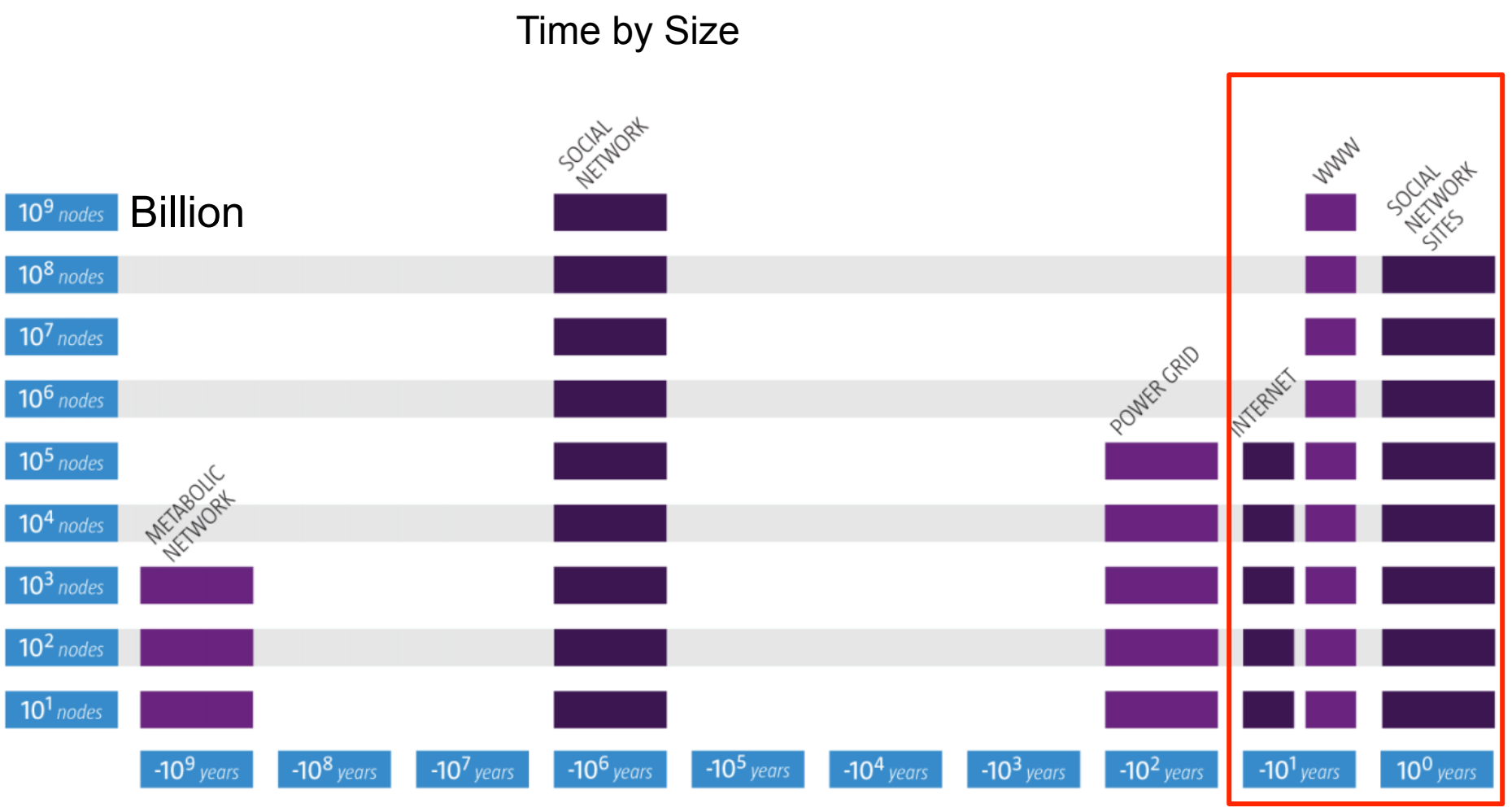


Internet



Society

The Life of Networks



Networks, why now?

Why are networks suddenly important?

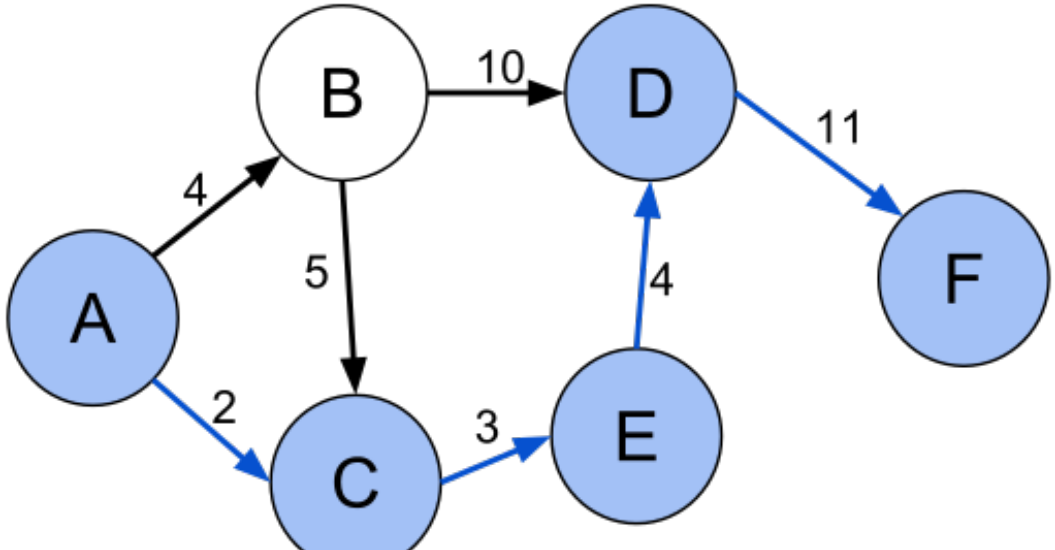
Last 10-20 years

LIVE QUESTIONS Graphs and MapReduce

- **Graph algorithms typically involve:**
  - Performing computation at each node
  - Processing node-specific data, edge-specific data, and link structure
  - Traversing the graph in some manner
- **Key questions:**
  - How do you represent graph data in MapReduce?
  - How do you process a graph in stateless MapReduce?

LIVE QUESTIONS

For maximum parallelism, you need the Maps and Reduces to be stateless, to not depend on any data generated in the same MapReduce job.
You cannot control the order in which the maps run, or the reductions.



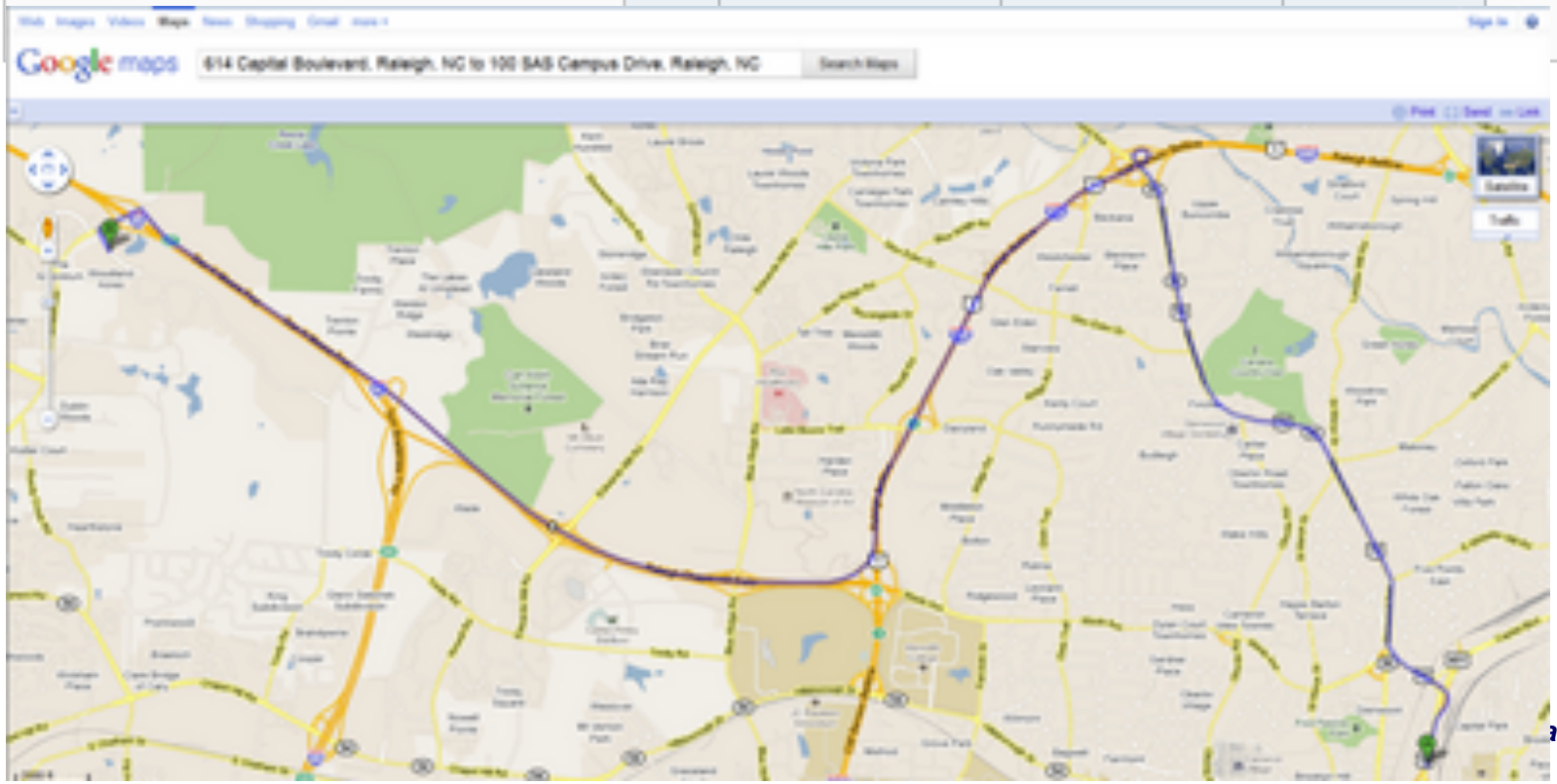
Some Graph Problems

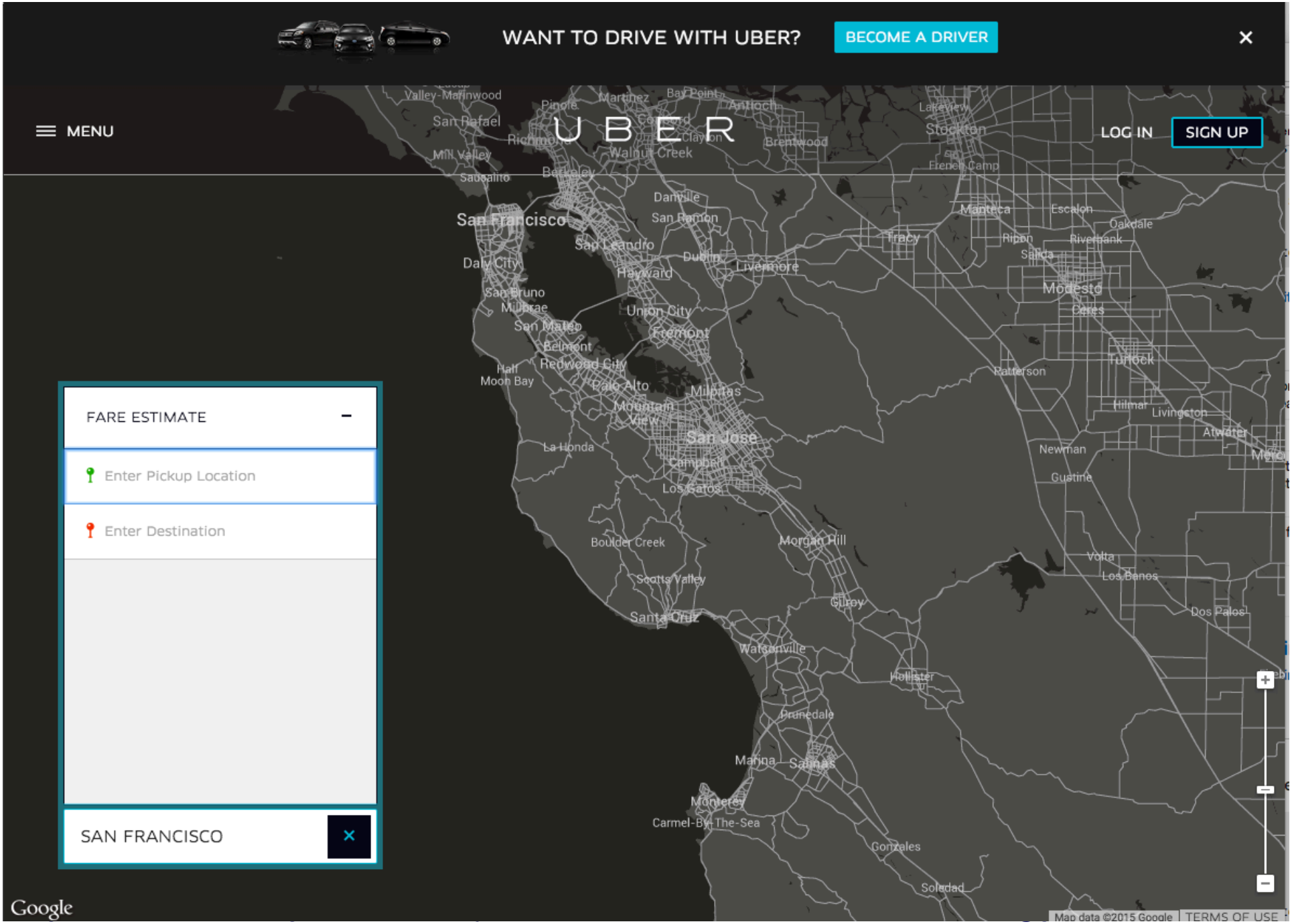
- **Graph search and path planning**
  - Routing Internet traffic and UPS delivery trucks
  - Finding experts, friend recommendations
- **Finding minimum spanning trees**
  - Telco laying down fiber
- **Finding Max Flow**
  - Airline scheduling
- **Identify "special" nodes and communities**
  - Breaking up terrorist cells, spread of avian flu
- **Bipartite matching**
  - Monster.com (employer-Applicant), Match.com (person-person), Netflix(viewer-movie), recommender systems, finding restuarants
- **Text summarization, concept extraction (TextRank)**
- **Find popular webpages: PageRank, Hits**

Figure 2.3: Shortest Path for Road Network at 5:00 P.M.

Seasonality
- Rush hour
- Game night

order	start_inter	end_inter	time_to_travel
1	SASCampusDrive	US40W/HarrisonAve	1.2000
2	US40W/HarrisonAve	RaleighExpy/US40W	1.4182
3	RaleighExpy/US40W	US440W/RaleighExpy	3.0000
4	US440W/RaleighExpy	US70W/US440W	2.7000
5	US70W/US440W	Capital/US70W	3.2000
6	Capital/US70W	614CapitalBlvd	1.4400
			12.9582




WANT TO DRIVE WITH UBER?
BECOME A DRIVER
MENU
UBER
LOG IN
SIGN UP
FARE ESTIMATE
Enter Pickup Location
Enter Destination
SAN FRANCISCO
San Francisco
Oakland
San Jose
Fremont
Hayward
Modesto
Stockton
Tracy
Livermore
Santa Cruz
Monterey
Salinas
Gilroy
Morgan Hill
Watsonville
Google
Map data ©2015 Google
TERMS OF USE

Single source: smallest number of edges that must be traversed in order to get to every vertex

- **Challenge 1: Shortest path to all nodes from a source**
  - Some times we want to reach all nodes in a graph from a single starting node…how can we do this in the shortest way possible?
- **Challenge 2: Shortest path to a single destination**
  - Sometimes we just have one target destination in mind
- **Do we know an algorithm for determining this?**

!! Yes: breadth-first search.

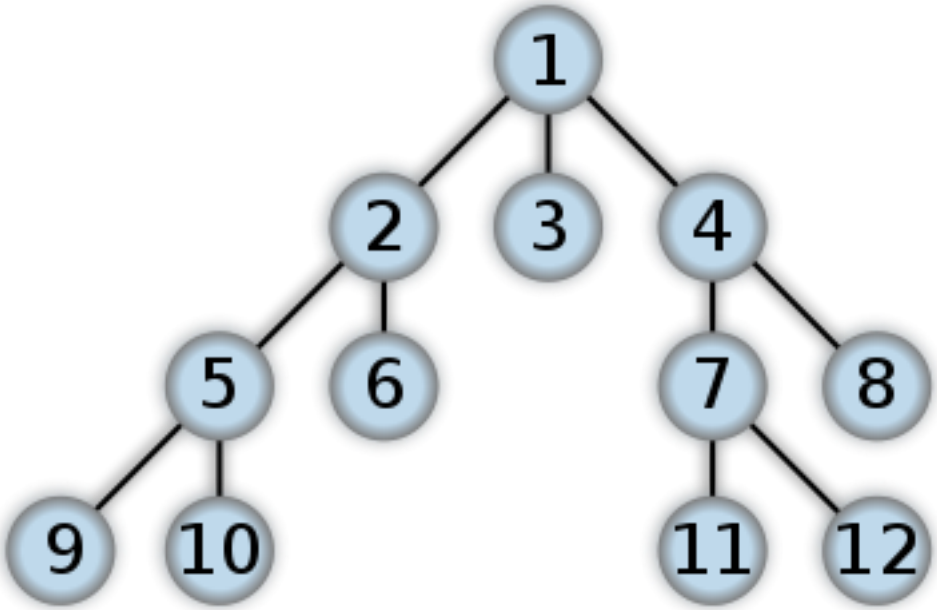
Single source: smallest number of edges that must be traversed in order to get to every vertex

- **Shortest is relative**
  - Unweighted
  - Weighted (sum of edge weights on path from source node to target node)

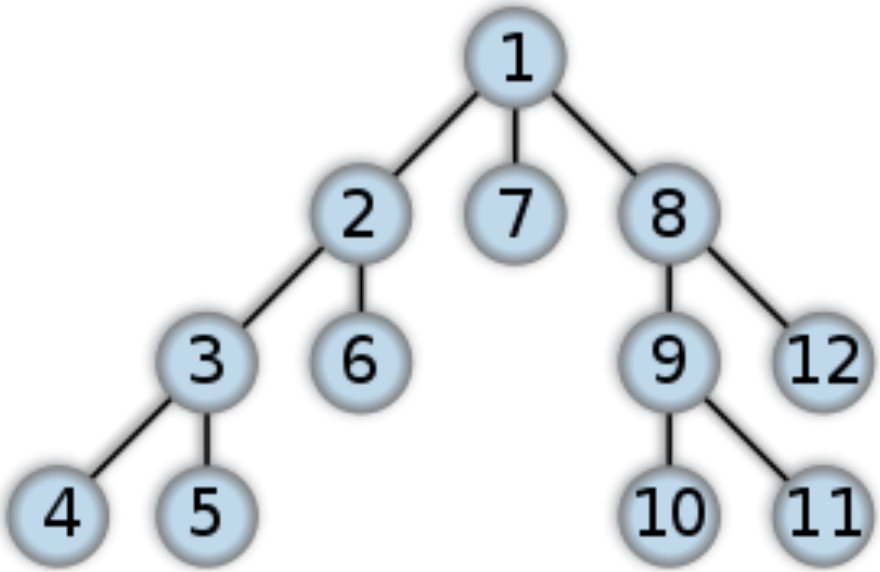
- **Graph Traversal**
  - Unweighted single-source shortest path
  - BFS, DFS
- **Dijkstra algorithm**
  - Weighted single-source shortest path
  - Modified BFS (with weights+priority queue)

Single Source Short path Algorithm

- **Breadth first traversal (FIFO Queue)**
- **Breadth first traversal + Priority Queue == Dijkstra algorithm**



Breadth first traversal



Depth first traversal

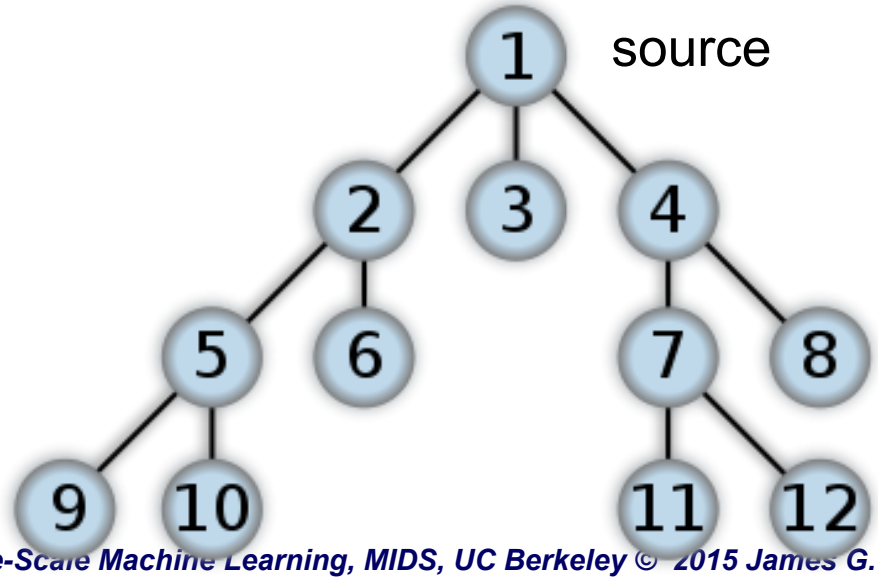
BFS is an O(V+E)

- The running time of that algorithm is O(V+E) where V is the number of vertices and E is the number of edges,
- because it pushes each reachable vertex onto the queue and considers each outgoing edge from it once.
- There can't be any faster algorithm for solving this problem, because in general the algorithm must at least look at the entire graph, which has size O(V+E).

Breadth-First Traversal

Breadth-first traversal of a graph:

- **Is roughly analogous to level-by-level traversal of an ordered tree**
- **Start the traversal from an arbitrary vertex;**
- **Visit all of its adjacent vertices;**
- **Then, visit all unvisited adjacent vertices of those visited vertices in last level;**
- **Continue this process, until all vertices have been visited.**



Distance from source is incremented by each iteration; if node has already been visited then we have found its shortest path already (so discard the new path)

BFS (FIFO Q): create a tree like structure with the source at the root: Visit each node

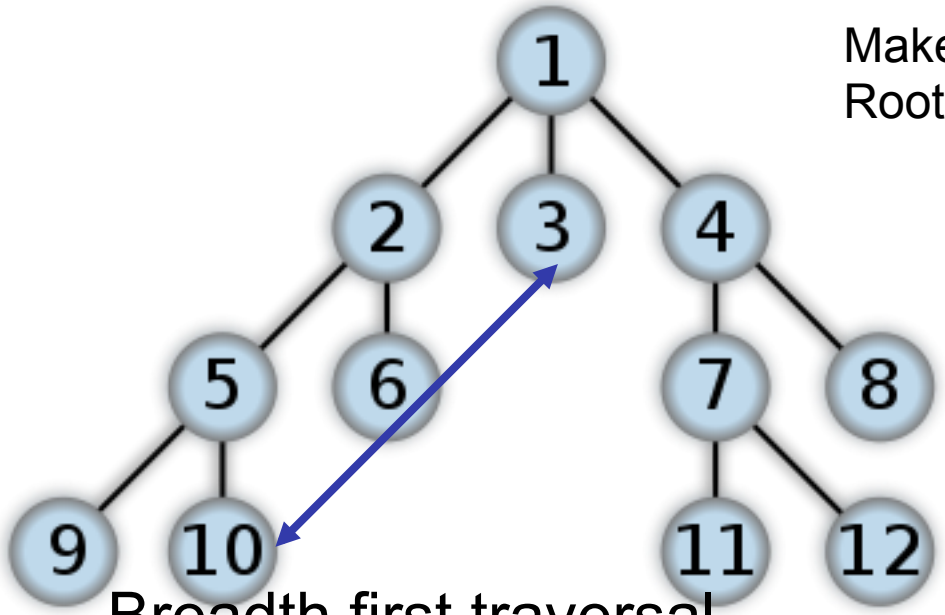
Step	Visited	FIFO_Q	unvisited
1	1	1	2-12
1 Expand(1)→2,3,4	1:0,{2,3,4}:1	2,3,4	5-12
1 Expand(2)→5,6	1:0,{2,3,4}:1 {5,6}:2	3,4,5,6	7-12

Level 0; dist 0

Level 1; dist 1

Level 2; dist 2

Level 3; dist 3



Make source node
Root of tree

Breadth first traversal

BFS (FIFO Q): create a tree like structure with the source at the root: Visit each node

Step	Visited	FIFO_Q	unvisited
1	1	A	2-12
1 Expand(1)→2,3,4	A0,{2,3,4}:1	2,3,4	5-12
1 Expand(2)→5,6	A0,{2,3,4}:1 {5,6}:2	3,4,5,6	7-12

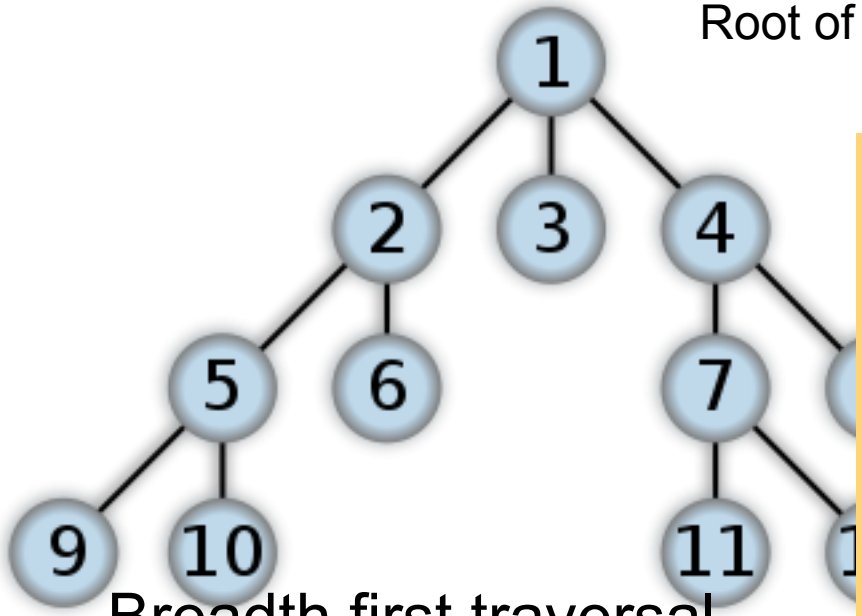
Make source node
Root of tree

Level 0; dist 0

Level 1; dist 1

Level 2; dist 2

Level 3; dist 3



Breadth first traversal

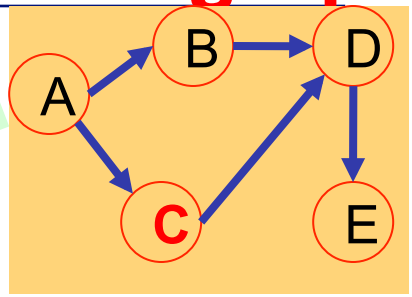
The output of BFS is the visited structure where each node of the connected component containing the source node is represented (ir a hashmap) with value of the path from source node to this node and the length of the path

BFS + counts path lengths from source Unweighted graph

```
1.  LET
2.      val q: queue = new_queue()
3.      val visited: vertexMap = create_vertexMap()
4.      (* visited maps vertex->int *)
5.      fun expand(v: vertex) =
6.       let val neighbors: vertex list = Gr
7.           val dist: int = valOf(get(visited,
8.           fun handle_edge(v': vertex) =
9.             case get(visited, v') of
10. Case 1     SOME(d') =>()/*already visited  so
11. Case 2   | NONE => ( add(visited, v', d
12.                        push(q, v') )
13.       in
14.         app handle_edge neighbors
15.       end
16.  IN
17.    S0 add(visited, v0, 0);
```

The expand function moves a fror
vertex into the completed set and
expands the frontier to include an
previously unseen neighbors of th

This is a Key slide

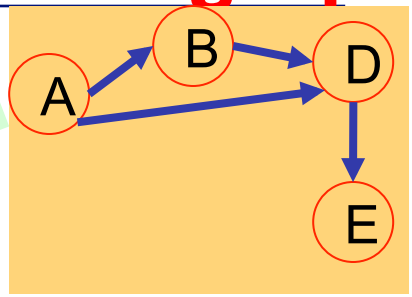


Step	Visited	q(fifo)	unvisited
0	A		BCDE
1	A,B:AB,C:AC	BC	DE
2 pop(B)	A 0 1 1 B:AB,C:AC, D:ABD 2	CD	E
2 pop(C)	A, 1 1 B:AB,C:AC, D:ABD 2	D(D deleted)	E
2 pop(D)	A, 1 1 B:AB,C:AC, D:ABD, 2 E:ABDE 3	DE	{}
2 pop(E)	----------	{}	{}

BFS + counts path lengths from source Unweighted graph

```
1.  LET
2.      val q: queue = new_queue()
3.      val visited: vertexMap = create_vertexMap()
4.      (* visited maps vertex->int *)
5.      fun expand(v: vertex) =
6.       let val neighbors: vertex list = Gr
7.           val dist: int = valOf(get(visited,
8.           fun handle_edge(v': vertex) =
9.             case get(visited, v') of
10. Case 1     SOME(d') =>()/*already visited  so
11. Case 2   | NONE => ( add(visited, v', d
12.                        push(q, v') )
13.       in
14.         app handle_edge neighbors
15.       end
16.  IN
17.    S0 add(visited, v0, 0);
```

This is a Key slide



Step	Visited	q(fifo)	unvisited
0	A		BCDE
1	A,B:AB,C:AC	BC	DE
2 pop(B)	A 0 1 1 B:AB,C:AC, D:ABD 2	CD	E
2 pop(C)	A, 1 1 B:AB,C:AC, D:ABD 2	D(D deleted)	E
2 pop(D)	A, 1 1 B:AB,C:AC, D:ABD, E:ABDE 2 3	DE	{}
2 pop(E)	----------	{}	{}

The expand function moves a fror
vertex into the completed set and
expands the frontier to include an
previously unseen neighbors of th

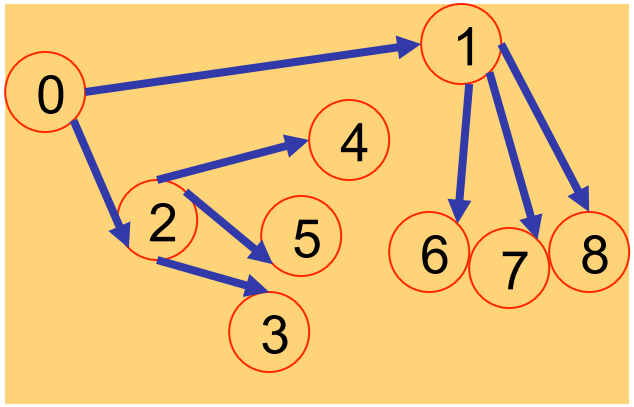
Single threaded breadth-first search

- **One common task involving a social graph is to use it to construct a tree, starting from a particular node.**
  - E.g. to construct the tree of *Frank's friends* and the *Frank's friends of friends*, etc. the simplest way to do this is to perform what is called a breadth-first search (BFS).

- **[You can read all about that here. this reference also includes a pseudo-code implementation. below follows a Java implementation. (the Node class is a simple bean. you can download Node.java here and Graph.java here.)]**

Social Graph Example (unweighted)

- **For an introduction to graph theory, start here.**
- **A common use of a graph is the "social graph" - e.g. your network of friends, as represented on a social network such as linkedin or facebook.**
  - One way to store a graph is using an adjacency list. in an adjacency list, each "node on the graph" (e.g. each person) is stored with a link to a list of the "edges emanating from that node" (e.g. their list of friends). for example :
  - Frank -> {mary, jill}
    Jill -> {frank, bob, james}
    Mary -> {william, joe, erin}
  - Or numerically:
    - 0-> {1, 2}
      2-> {3, 4, 5}
      1-> {6, 7, 8}

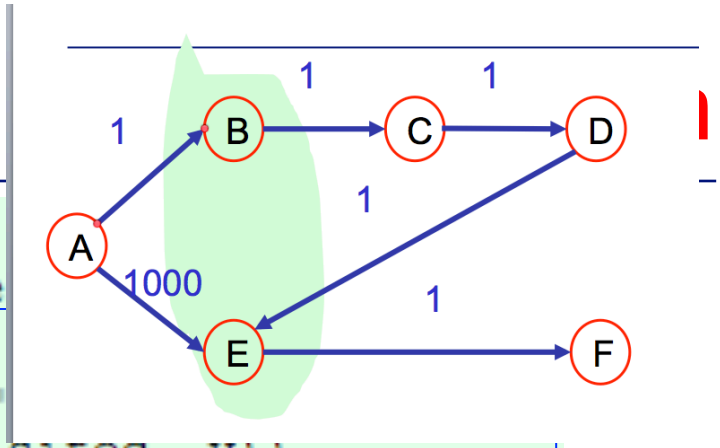


BFS with weighted edges: Versio


```
1.  let val q: queue = new_queue()
2.    val visited: vertexMap = create_vertexMap()
3.    fun expand(v: vertex) =
4.      let val neighbors: vertex list = Graph.outgoing(v)
5.          val dist: int = valOf(get(visited, v))
6.          fun handle_edge(v': vertex, weight: int) =
7.            case get(visited, v') of
8.      Case 1  SOME(d') =>
9.                if dist+weight < d'
10.               then add(visited, v', dist+weight)
11.               else ()
12.     Case 2 | NONE => ( add(visited, v', dist+weight);
13.                        push(q, v') )
14.     in
15.       app handle_edge neighbors
16.     end
17. in
18. S0 add(visited, v0, 0);
19. S1 expand(v0);
20. S2 while (not (empty_queue(q)) do expand(pop(q))
21. end
```

its adds the weight of the edge traversed. Here is a first cut at an algorithm

BFS with weighted 1



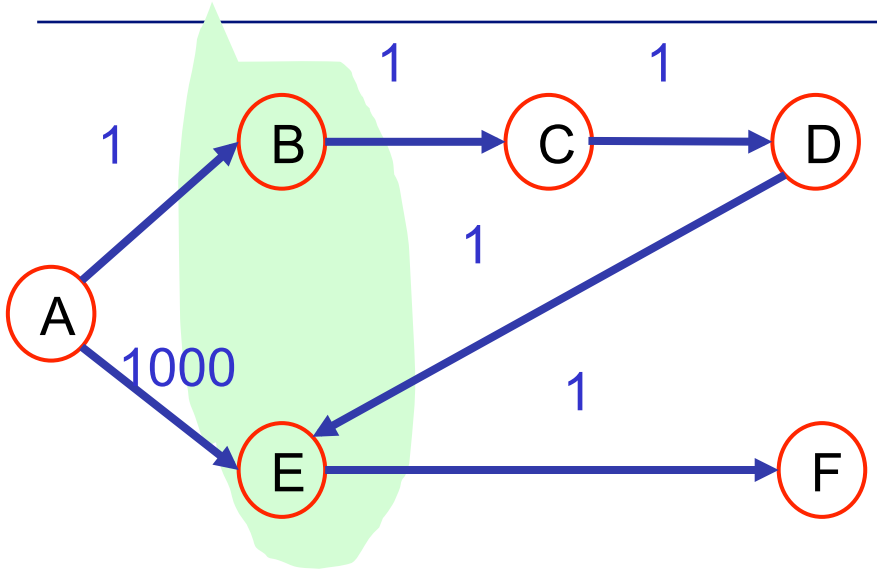

```
1.  let val q: queue = new_queue()
2.    val visited: vertexMap = create_ve
3.    fun expand(v: vertex) =
4.      let val neighbors: vertex list =
5.          val dist: int = valOf(get(visited, v))
6.          fun handle_edge(v': vertex, weight: int) =
7.            case get(visited, v') of
8.      Case 1   SOME(d') =>
9.                 if dist+weight < d'
10.                then add(visited, v', dist+weight)
11.                else ()
12.     Case 2 | NONE => ( add(visited, v', dist+weight);
13.                        push(q, v') )
14.     in
15.       app handle_edge neighbors
16.     end
17. in
18. S0 add(visited, v0, 0);
19. S1 expand(v0);
20. S2 while (not (empty_queue(q)) do expand(pop(q))
21. end
```

its adds the weight of the edge traversed. Here is a first cut at an algorithm

Length of path AE?
a: 1000
b: 4 (correct)

Length of path AF?
a: 1001
b: 5

Quiz 7.10.1

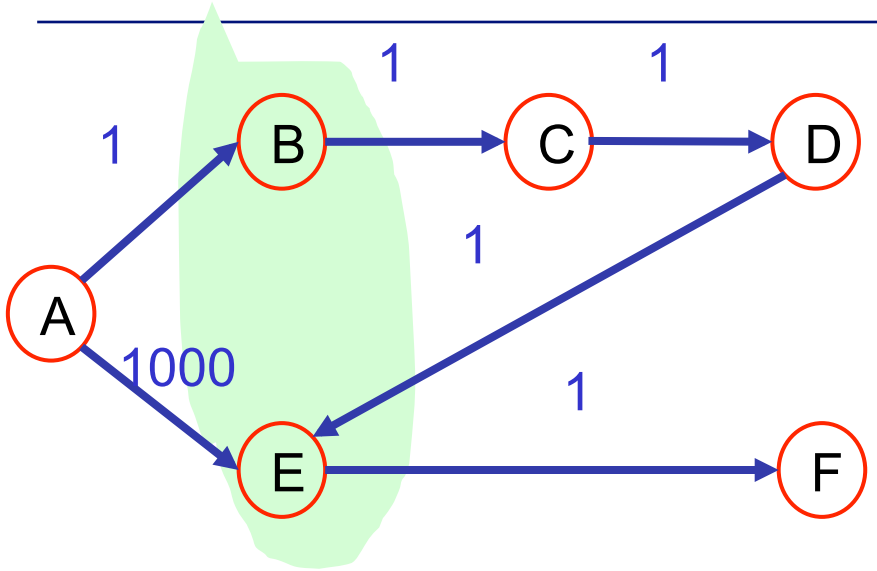


Step	Visited	FIFO_Q	unvisited
0	A	A	BCDEF
1 Expand(A)→BE	A0,B1,E5	B,E	CDF
2 Expand(B)→ C	…..	…	
	….		

Multichoice Question
Given the above graph, and using the BFS algorithm with weighted edges (Version 1) what is the path length of A to F? Feel free to use the table to generate a trace of the BFS algorithm.

A 1001
B 5
C -9999
D None of the above

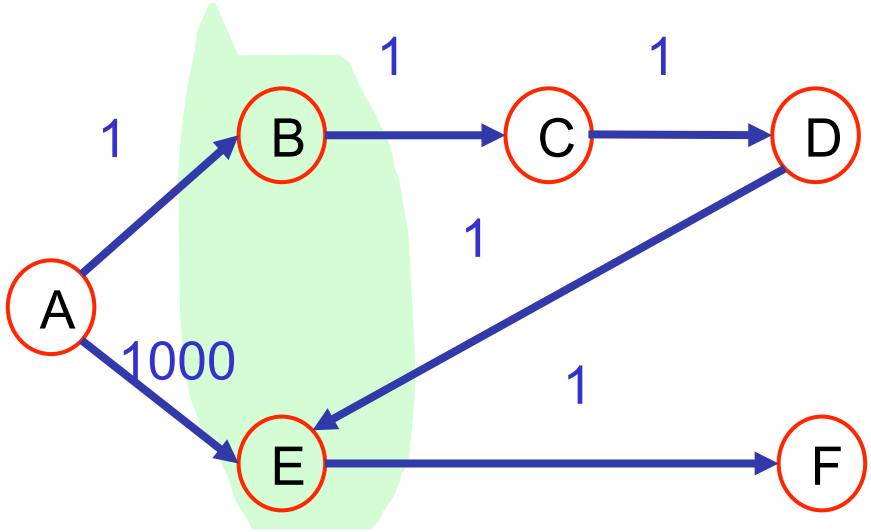
Quiz 7.10.1



Step	Visited	FIFO_Q	unvisited
0	A	A	BCDEF
1 Expand(A)→BE	A0,B1,E5	B,E	CDF
2 Expand(B)→ C	…..	…	
	….		

Multichoice Question
Given the above graph, and using the BFS algorithm with weighted edges (Version 1) what is the path length of A to F? Feel free to use the table to generate a trace of the BFS algorithm.

A 1001 (Correct Answer; NOTE a full explanation will follow in the next section)
B 5
C -9999
D None of the above



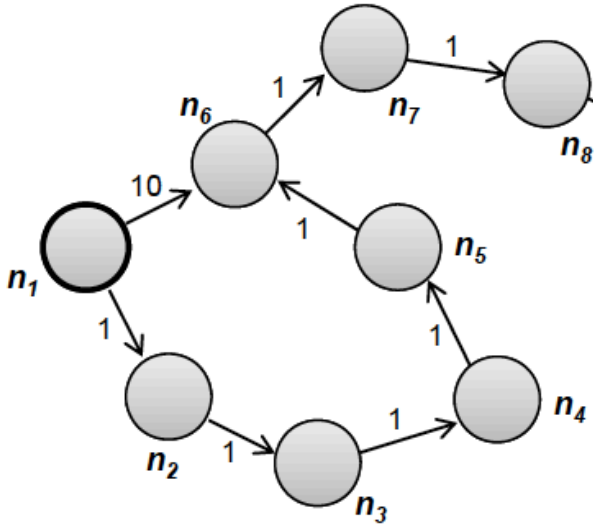
BFS does NOT work for weighted graphs

Using BFS what is the path length according to BFS? What is the shortest path?

This is a BFS, so we don't revisit node F

Step	Visited	FIFO_Q	unvisited
0	A	A	BCDEF
1 Expand(A)→BE	A0,B1,E5	B,E	CDF
2 Expand(B)→ C	A0,B1,E1000,C2	E,C	DF
3 Expand(E)→ F	A0,B1,E1000,C2, F=1001	C,F	D
4 Expand(C)→ D	A0,B1,E1000,C2, F=1001, D3	F, D	
5 Expand(F)→ {}	A0,B1,E1000,C2, F=1001, D3	D	
6 Expand(D)→ E	A0,B1,**E4**,C2, F=1001, D3	{}…replace E1000 with E4	
Finished	A0,B1,**E4**,C2, **F=1001**, D3	FIFO_Q is empty	

Quiz 7.14.1 : How many iterations of



Q: Using the parallel breadth-first search algorithm how many iterations are required to discover the shortest distances to all nodes from n1

A 7 (Correct answer)

B 8

C 4

D None of the above

Quiz 7.14.1 : How many iterations of

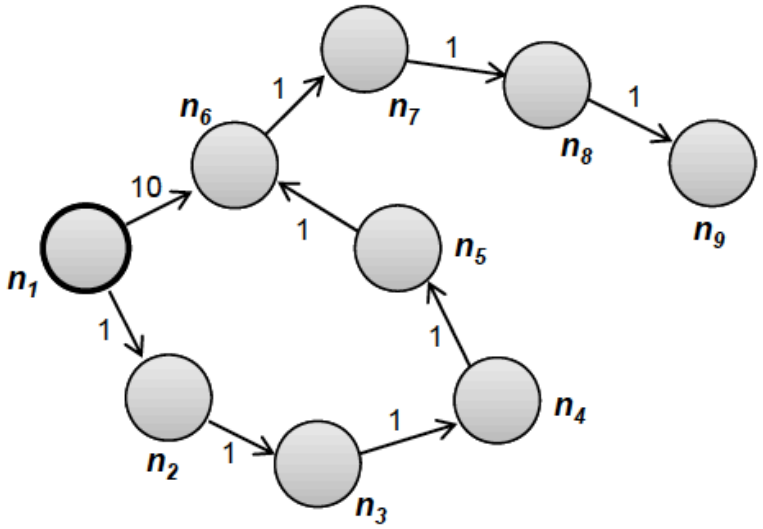


Figure 5.6: A sample graph that elicits worst-case behavior for parallel breadth-first search. Eight iterations are required to discover shortest distances to all nodes from n_1.

Q: How iterations of the parallel breadth-first shortest-path are required to discover the shortest distances to all nodes from n1

A 5

B 8 (Correct answer)

C 4

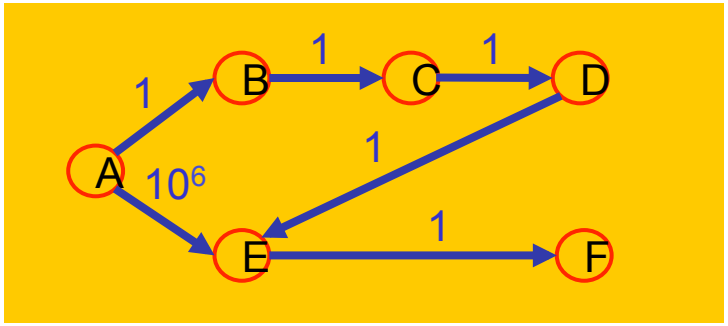
D None of the above

Eight iterations are required to discover shortest distances to all nodes from n1 .

Audience Participation

- **Who reinvented Dijkstra SSSP as you went through the lecture?**

BFS does NOT work for weighted graphs



- **2 flavors of path finding algorithms**
  - Visit all nodes
  - Stop when you have found what you need
- **BFS algorithm does not work for the second scenario**
- **To see why, consider the following graph, where the source vertex is v_0 = A.**
  - The first pass of the algorithm will add vertices B and E to the map visited, with distances 1 and 10^6 respectively.
  - Then C and F will be added to the map visited, with distances 2 and 10^6+1.
  - Then D will be added with distance 3
  - After that, the shortest distance of E will be updated to 4 given D as the frontier.
  - Now, all the node has been visited. Stop!

The distance for F is still 10^6+1, but the true value should be 5.

BFS + 2 fixes gives us Dijkstra's Shortest Path

- **1: new path an improvement over a possible previous one**
  - In the SOME case a check is needed to see whether the path just discovered to the vertex v' is an improvement on the previously discovered path (which had length d)
- **2: The queue q should not be a FIFO queue.**
  - Instead, it should be a *priority queue* where the priorities of the vertices in the queue are their distances recorded in visited.
  - That is, pop(q) should be a priority queue extract_min operation that removes the vertex with the smallest distance.
  - The priority queue must also support a new operation increase_priority(q,v) that increases the priority of an element v already in the queue q.
  - This new operation is easily implemented for heaps using the same bubbling-up algorithm used when performing heap insertions.

Dijkstra's Shortest Path

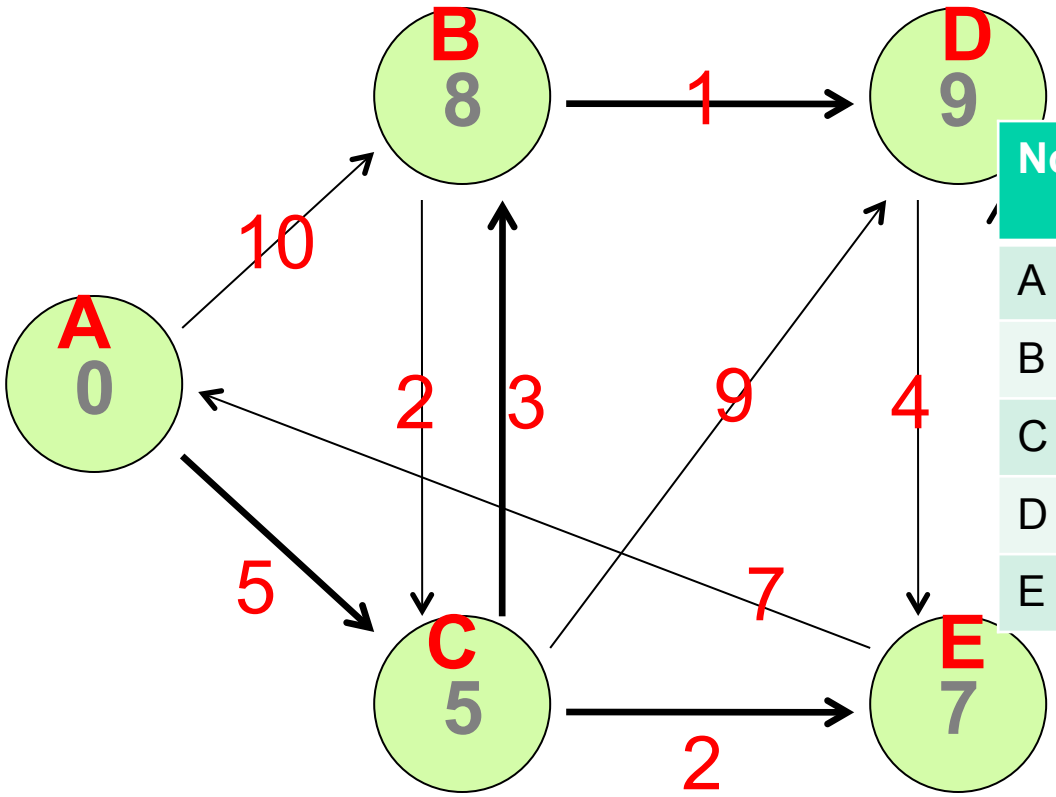
```
(* Dijkstra's Algorithm *)
let val q: queue = new_queue()
  val visited: vertexMap = create_vertexMap()
  fun expand(v: vertex) =
    let val neighbors: vertex list = Graph.outgoing(v)
        val dist: int = valOf(get(visited, v))
        fun handle_edge(v': vertex, weight: int) =
          case get(visited, v') of
            SOME(d') =>
              if dist+weight < d'
              then ( add(visited, v', dist+weight);
                     incr_priority(q, v', dist+weight) )
              else ()
          | NONE => ( add(visited, v', dist+weight);
                      push(q, v', dist+weight) )
    in
      app handle_edge neighbors
    end
in
  add(visited, v0, 0);
  expand(v0);
  while (not (empty_queue(q)) do expand(pop(q))
end
```

Case 1

Update priority

Case 2

Dijkstra's Algorithm Example



NodeID	Distance from A	Shortest Path
A	0	{}
B	8	AB
C	5	AC,
D	9	ACD
E	7	ACE

	E7		
4 pop(B)	A0,C5, E7,B8	D9	{}
5 pop(D)	A0,C5, E7,B8, D9	{}	{}

End up with a data structure: Hashtable with an entry for each node and the corresponding distance from source node to each node (and path if required).

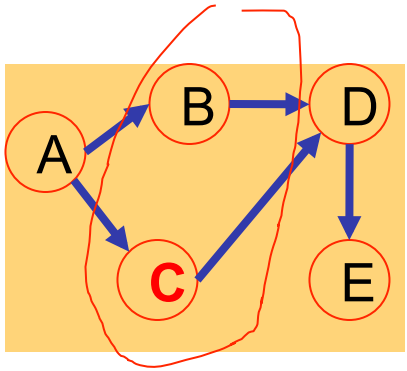
Sundry Items on SSSP: no negative weights

- **Short answer is that Dijkstra is a greedy algorithm, meaning we assume that we have made the best choice possible in every step of the algorithm.**
- **With negative edges, this is no longer possible and thus a greedy algorithm will not suffice.**

- **Unlike Dijkstra's algorithm, the Bellman–Ford algorithm can be used on graphs with negative edge weights, as long as the graph contains no negative cycle reachable from the source vertex*s*.**
  - The presence of such cycles means there is no shortest path, since the total weight becomes lower each time the cycle is traversed. It is possible to adapt Dijkstra's algorithm to handle negative weight edges by combining it with the Bellman-Ford algorithm (to remove negative edges and detect negative cycles), such an algorithm is called Johnson's algorithm.

Dijkstra's algorithm vs dynamic programming

- **Dijkstra's algorithm is a greedy algorithm.**
- **Because of this, there is no overlapping sub-problems and therefore no need for memoization of already calculated values.**
- **Dynamic Programming is a general approach to solving problems, much like "divide-and-conquer" is a general method, except that unlike divide-and-conquer, the subproblems will typically overlap.**
  - http://www.cs.cmu.edu/~avrim/451f12/lectures/lect1002.pdf

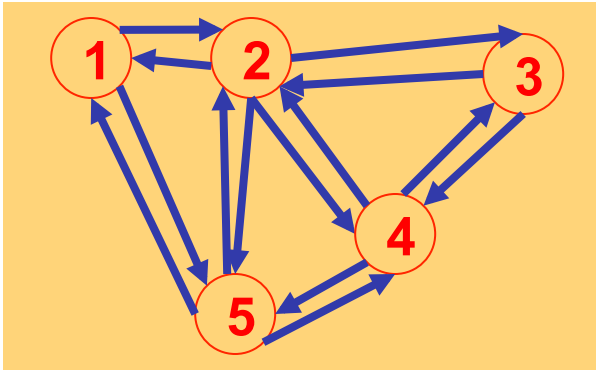

A
B
C
D
E

- **Distributed Dijkstras**

Example Problem: Input adjacency Lists + inits

- **Suppose we start with the following input graph, in which we've stated that node #1 is the source (starting point) for the search**
  - marked this one special node with distance 0 and frontier queue.
  - All other nodes are in the unvisited state U

Key	Value			
1	2,5	0	Q	
2	1,3,4,5	Integer.MAX_VALUE	U	
3	2,4	Integer.MAX_VALUE	U	
4	2,3,5	Integer.MAX_VALUE	U	
5	1,2,4	Integer.MAX_VALUE	U	

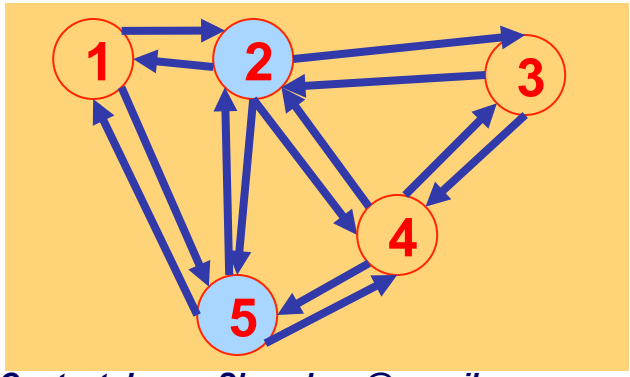


Node STATES
- Visited
- Queue
- Unvisited

SSSP First mapper Mapper

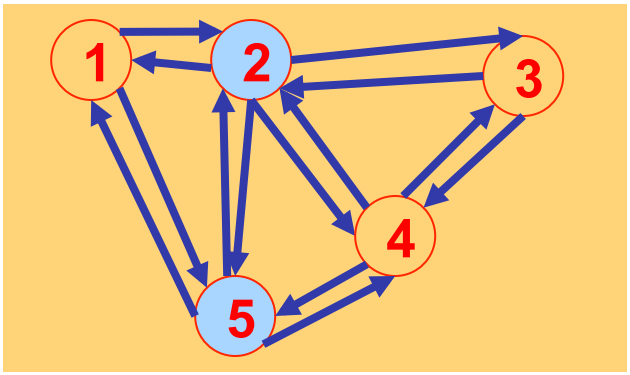
- **Mappers are responsible for expanding the frontier nodes.**
  - Similar in spirit to expand(), which adds to the frontier a newly found vertex at a distance one greater than that of its neighbor already in the frontier.
  - For each frontier node, expand(); the mappers emit a new frontier node, which is with distance = distance + 1.
  - They also then emit each old frontier node with a new status; updated to visited. (once a node has been expanded, we're done with it.)
  - Mappers also emit all "unvisited nodes, with no change.
  - ***Note*** *that when the mappers expand the frontier nodes and create a new node for each edge, they do not know what to write for the edges of this new node - so they leave it blank.*

1	2,5	0	V
2	NULL	1	Q
5	NULL	1	Q
2	1,3,4,5	Integer.MAX_VALUE	U
3	2,4	Integer.MAX_VALUE	U
4	2,3,5	Integer.MAX_VALUE	U
5	1,2,4	Integer.MAX_VALUE	U



Expand the current frontier using Mapper

Key	Value
1	2,5|0|Q|
2	1,3,4,5|Integer.MAX_VALUE|U|
3	2,4|Integer.MAX_VALUE|U|
4	2,3,5|Integer.MAX_VALUE|U|
5	1,2,4|Integer.MAX_VALUE|U|



Map

Frontier = {1}

Key	Value
1	2,5|0|V|
2	NULL|1|Q|
5	NULL|1|Q|
2	1,3,4,5|Integer.MAX_VALUE|U|
3	2,4|Integer.MAX_VALUE|U|
4	2,3,5|Integer.MAX_VALUE|U|
5	1,2,4|Integer.MAX_VALUE|U|

SSSP Reducer (merge candidate paths)

If node 2 is in visited State ignore all records in the stream and just emit the visited record for that node

- **The SSSP reducers receive all data for a given key**
  - in this case it means that they receive the data for all "copies" of each node. for example, the reducer that receives the data for key = 2 gets the following list of values

```
2    NULL  | 1                          | Q |
2    1,3,4,5 | Integer.MAX_VALUE | U |
```

- **The reducers job is to take all this data and construct/join a new node using**
  - the non-null list of edges
  - the minimum distance
  - the status

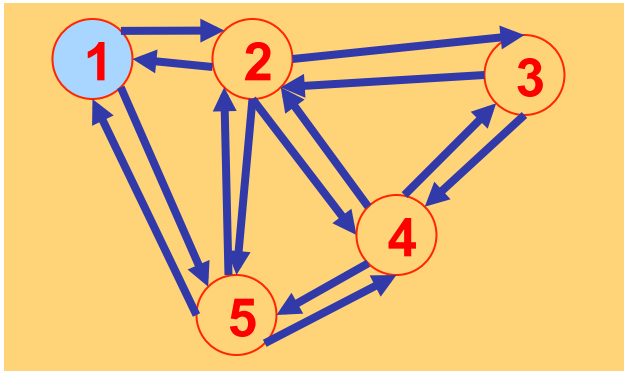
```
2    1,3,4,5 | 1 | Q |
```

SSSP MapReduce Iterations: 0th 1st & 2nd

Input Graph

Key	Value
1	2,5|0|Q|
2	1,3,4,5|Integer.MAX_VALUE|U|
3	2,4|Integer.MAX_VALUE|U|
4	2,3,5|Integer.MAX_VALUE|U|
5	1,2,4|Integer.MAX_VALUE|U|

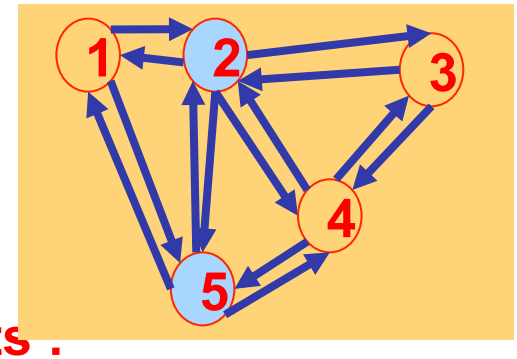
Init



Using this logic the output from our first iteration will be

1	2,5,|0|V
2	1,3,4,5,|1|Q
3	2,4,|Integer.MAX_VALUE|U
4	2,3,5,|Integer.MAX_VALUE|U
5	1,2,4,|1|Q

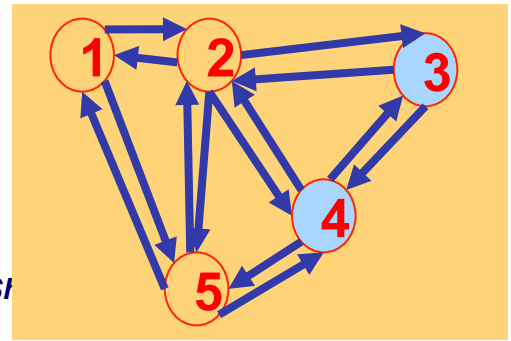
After first



the second iteration uses this as the input and outputs :

1	2,5,|0|V
2	1,3,4,5,|1|V
3	2,4,|2|Q
4	2,3,5,|2|Q
5	1,2,4,|1|V

After Second



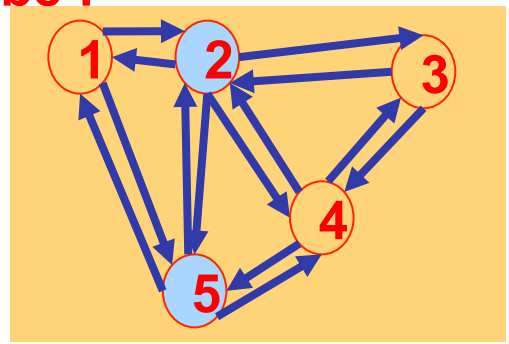
and the third iteration outputs:

SSSP MapReduce Iterations: 1st, 2nd & 3rd

Using this logic the output from our first iteration will be :

1 2,5,|0|V
2 1,3,4,5,|1|Q
3 2,4,|Integer.MAX_VALUE|U
4 2,3,5,|Integer.MAX_VALUE|U
5 1,2,4,|1|Q

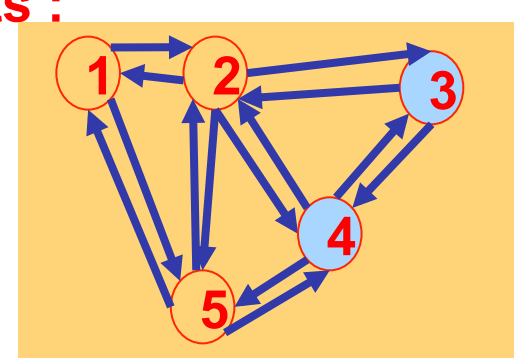
After first



the second iteration uses this as the input and outputs :

1 2,5,|0|V
2 1,3,4,5,|1|V
3 2,4,|2|Q
4 2,3,5,|2|Q
5 1,2,4,|1|V

After Second



and the third iteration outputs:

1 2,5, | 0 | V
2 1,3,4,5 | 1 | V
3 2,4 | 2 | V
4 2,3,5 | 2 | V
5 1,2,4 | 1 | V

After third…. A fourth perhaps?

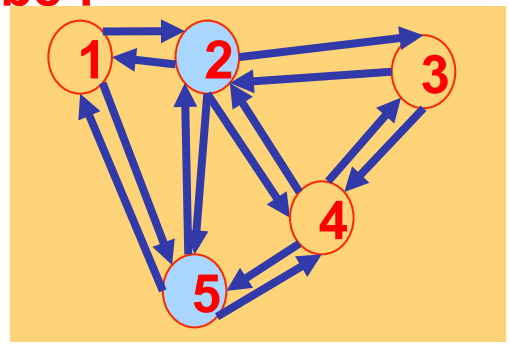
SSSP Termination: no nodes in Q State

- **Does the algorithm ever terminate?**
  - Eventually, all nodes will be discovered, all edges will be considered (in a connected graph)
  - Subsequent iterations will continue to print out the same output.
- **We are done when there are no output nodes that are frontier (i.e., in Q state).**

- **Note - if NOT all nodes in your input are actually connected to your source, you may have final output nodes that are still UNVISITED.**

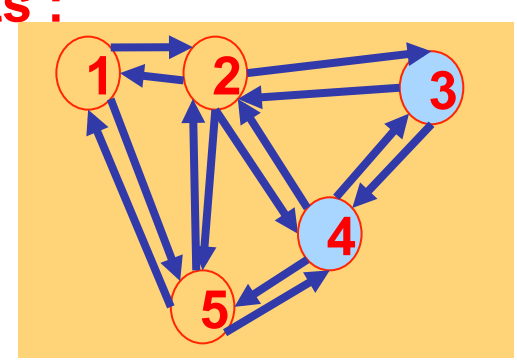
SSSP MapReduce Iterations: 1st, 2nd & 3rd

Using this logic the output from our first iteration will be :



1 2,5,|0|V
2 1,3,4,5,|1|Q
3 2,4,|Integer.MAX_VALUE|U
4 2,3,5,|Integer.MAX_VALUE|U
5 1,2,4,|1|Q

the second iteration uses this as the input and outputs :



1 2,5,|0|V
2 1,3,4,5,|1|V
3 2,4,|2|Q
4 2,3,5,|2|Q
5 1,2,4,|1|V

and the third iteration outputs:

1 2,5, | 0 | V
2 1,3,4,5 | 1 | V
3 2,4 | 2 | V
4 2,3,5 | 2 | V
5 1,2,4 | 1 | V

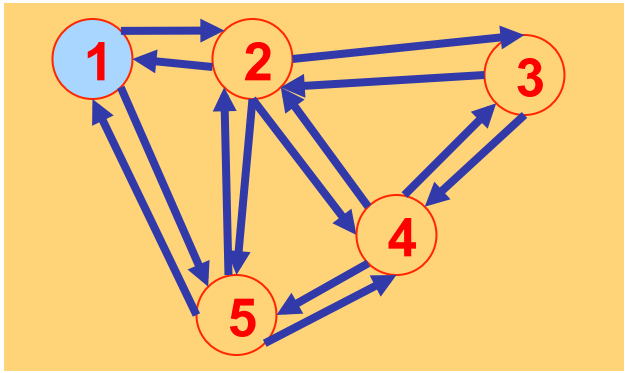
No nodes in Q-State so terminate

SSSP MapReduce Iterations: 0th 1st & 2nd

Input Graph

Key	Value
1	2,5|0|Q|
2	1,3,4,5|Integer.MAX_VALUE|U|
3	2,4|Integer.MAX_VALUE|U|
4	2,3,5|Integer.MAX_VALUE|U|
5	1,2,4|Integer.MAX_VALUE|U|

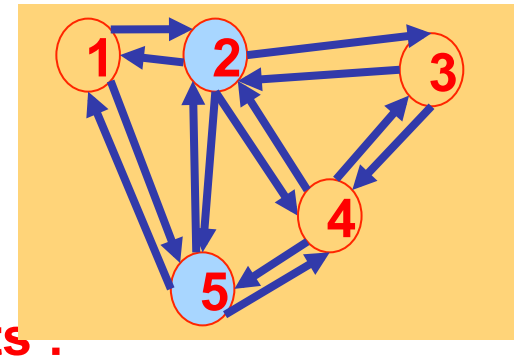
Init



Using this logic the output from our first iteration will be

1	2,5,|0|V
2	1,3,4,5,|1|Q
3	2,4,|Integer.MAX_VALUE|U
4	2,3,5,|Integer.MAX_VALUE|U
5	1,2,4,|1|Q

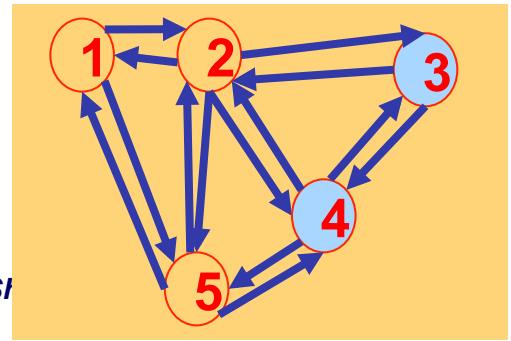
After first



the second iteration uses this as the input and outputs :

1	2,5,|0|V
2	1,3,4,5,|1|V
3	2,4,|2|Q
4	2,3,5,|2|Q
5	1,2,4,|1|V

After Second



and the third iteration outputs:

Implementation MR Job 1 of 2: init

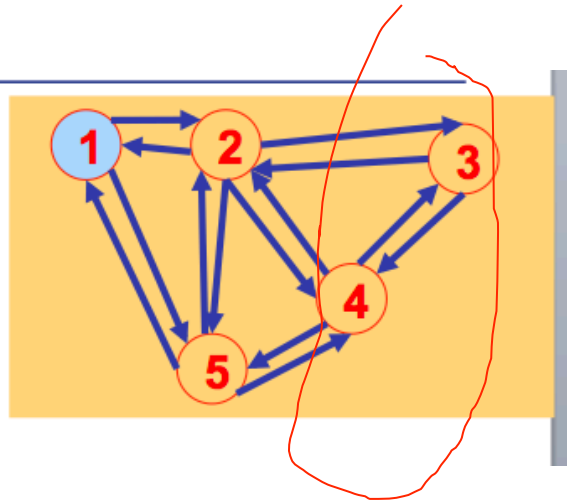
Job 1

- **Given an graph (adjacency list)**
- **Broadcast the source node**
- **Mapper (append status field to each node)**
  - If source node append Q status
  - Else append Q status U status
- **Reducer**
  - Blank

Input Graph

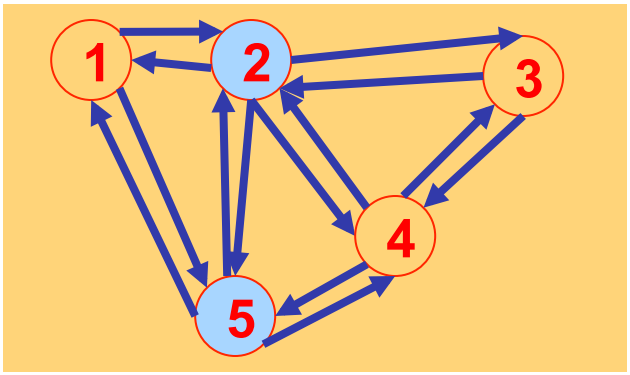
Key	Value
1	2,5|0|Q|
2	1,3,4,5|Integer.MAX_VALUE|U|
3	2,4|Integer.MAX_VALUE|U|
4	2,3,5|Integer.MAX_VALUE|U|
5	1,2,4|Integer.MAX_VALUE|U|

Init



Job 2 Mapper: Expand the current frontier

Key	Value
1	2,5|0|Q|
2	1,3,4,5|Integer.MAX_VALUE|U|
3	2,4|Integer.MAX_VALUE|U|
4	2,3,5|Integer.MAX_VALUE|U|
5	1,2,4|Integer.MAX_VALUE|U|



Map Expand the current frontier
And mark Q node as V

Frontier = {1}

Key	Value
1	2,5|0|V|
2	NULL|1|Q|
5	NULL|1|Q|
2	1,3,4,5|Integer.MAX_VALUE|U|
3	2,4|Integer.MAX_VALUE|U|
4	2,3,5|Integer.MAX_VALUE|U|
5	1,2,4|Integer.MAX_VALUE|U|

Null neighbors

Implementation MR Job 2 of 2: iteration

Job 2

- **Given an graph (adjacency list) with states**
- **Mapper (Process Q nodes by expanding)**
  - If node N in Q status then
    - put neighbors in Q mode and update path length by incrementing path length of N
    - Update status of node N to V
  - Else pass node N thru
- **Reducer**
  - ??????

Job2 SSSP Reducer (merge candidate paths)

- **If node N is in visited State**
  - ignore all records in the stream and just emit the visited record for that node
- **elseIf node in U state just emit that record**
- **Elseif Node in Q state (merge records by taking min distance)**
  - The SSSP reducers receive all data for a given key
    - in this case it means that they receive the data for all "copies" of each node. for example, the reducer that receives the data for key = 2 gets the following list of values

```
2      NULL  | 1                              | Q |
2      1,3,4,5 | Integer.MAX_VALUE | U |
```

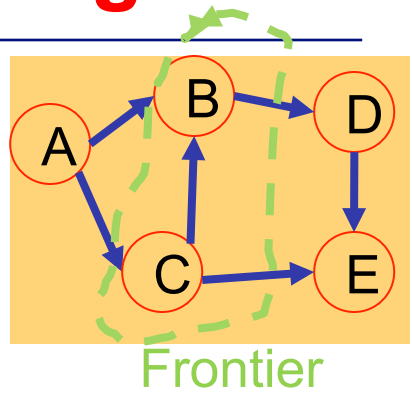
  - The reducers job is to take all this data and construct/join a new node using
    - the non-null list of edges (neighbors)
    - the minimum distance
    - the status

```
2      1,3,4,5 | 1 | Q |
```

BFS unweighted: From Intuition to Algorithm

Graph →Adjacency Lists

Key	Value
A	B,C |Frontier
B	D
C	B,E
D	E
E	



Init

Put Source node in Frontier

Repeat

- **Map**
  - A map task process all **frontier** nc
    - Key: node *n*
    - Value: *points-to* (list of nodes reachable from *n*)
    - ∀*p* ∈ *points-to*: emit path to node p (*p*,*np*); expand the frontier with *p*
    - *Emit n as Visited*
  - All other nodes are just passed thru
- **Reduce**
  - The reduce task gathers possible paths to a given node *p* and select one;
  - and joins it to the frontier nodes;
  - and puts the visited nodes in the visited state

until no nodes in frontier queue

Each node has record in the stream

Key	Value=Nbors|path|status
A	B,C |A |Visited
B	D | AB |Visited
C	B,E | AC |Visited
D	E |ABD |Visited
E	| ABDE |Visited

SSSP in MapReduce for unweighted graphs

- **Init Graph MR Job 1**
- **While nodes in Q status**
  - Perform MR Job 2 (expand each frontier node)

Job2 SSSP Reducer (merge candidate paths)

- **If node N is in visited State**
  - ignore all records in the stream and just emit the visited record for that node
- **elseIf node in U state just emit that record**
- **Elseif Node in Q state**
  - The SSSP reducers
    - in this case it
      of each node
      for key = 2 ge

      2 NU
      2 1,

  - The reducers job is to take all this data and construct/join a new node using
    - the non-null list of edges (neighbors)
    - the minimum distance
    - the status

      2 1,3,4,5 | 1 | Q |

The order inversion pattern exploits the sorting phase of MapReduce to push data needed for calculations to the reducer ahead of the data that will be manipulated

Should we use the order inversion pattern here?

Job2 SSSP Reducer (merge candidate paths)

- **If node N is in visited State**
  - ignore all records in the stream and just emit the visited record for that node
- **elself node in U state just emit that record**
- **Elseif Node in Q state**
  - The SSSP reducers receive all data for a given key

The order inversion pattern exploits the sorting phase of MapReduce to push data needed for calculations to the reducer ahead of the data that will be manipulated

Should we use the order inversion pattern here?

NOT necessary as keys/nodes with multiple records are either:
in V mode (and then ignore all other records for N
in Q mode (join records by taking min distance)
But using the order inversion pattern can simplify and speed up Reducer

Job2 SSSP Reducer (merge candidate paths)

- **If node N is in visited State**
  - ignore all records in the stream and just emit the visited record for that node
- **elself node in U state just emit that record**
- **Elseif Node in Q state**
  - The SSSP reducers receive all data for a given key

The order inversion pattern exploits the sorting phase of MapReduce to push data needed for calculations to the reducer ahead of the data that will be manipulated

Use a combiner?

Should we use the order inversion pattern here?

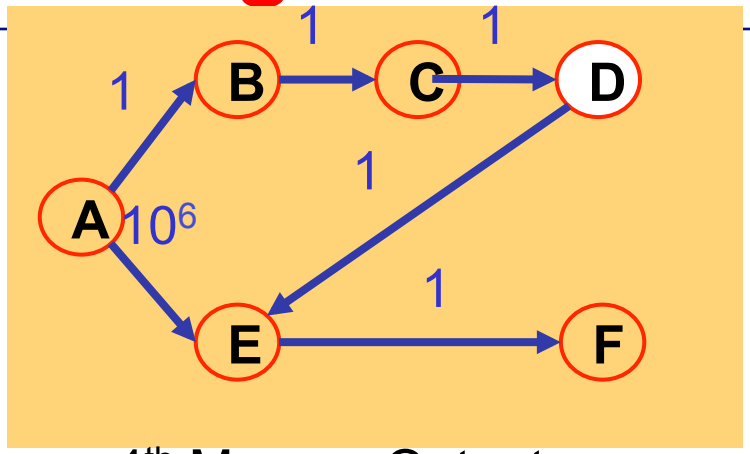
NOT necessary as keys/nodes with multiple records are either:
in V mode (and then ignore all other records for N
in Q mode (join records by taking min distance)
But using the order inversion pattern can simplify and speed up Reducer

Weighted SSSP: E is back

3rd Reducer Output

SSSP

Key	Value	Status
A	0	V
B	1	V
C	2	V
D	3	Q
E	10^6	V
F	10^6+1	V



4th Mapper Output

Key	Value
E	4

4th Reducer Output

SSSP

Key	Value	Status
A	0	V
B	1	V
C	2	V
D	3	V
E	4	Q
F	10^6+1	V

If distance to a visited node is shorter then reset status to frontier Q

Frontiers

D

E is updated with a shortest path so we need to add this to the frontier and recompute all shortest paths from paths emanating from node E

Frontiers

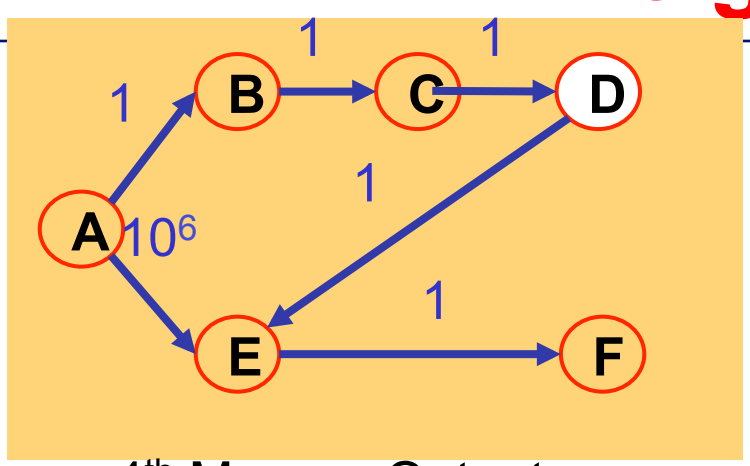
E

Weighted SSSP

3rd Reducer Output

SSSP

Key	Value	Status
A	0	V
B	1	V
C	2	V
D	3	Q
E	10^6	V
F	10^6+1	V



4th Mapper Output

4th Reducer Output

SSSP

Key	Value	Status
A	0	V
B	1	V
C	2	V
D	3	V
E	4	Q
F	10^6+1	V

Modify MapReduce Iteration Job for unweighted graphs as follows to get the weighted version

If distance to a visited node is shorter then reset status to frontier Q

Frontiers

D

E is updated with a shortest path so we need to add this to the frontier and recompute all shortest paths from paths emanating from node E

Frontiers

E

- **End of slides**