CAS CS 565, Data Mining

Course logistics

- Course webpage:
  - www.cs.bu.edu/~evimaria/teaching.html
- Schedule: Mon – Wed, 4-5:30
- Instructor: Evimaria Terzi, evimaria@cs.bu.edu
- Office hours: Mon 2:30-4pm, Tues 10:30am-12 (or by appointment)
- Mailing list : cascs565a1-l@bu.edu

Topics to be covered (tentative)

- Introduction to data mining and prototype problems
- Frequent pattern mining
  - Frequent itemsets and association rules
- Clustering
- Dimensionality reduction
- Classification
- Link analysis ranking
- Recommendation systems
- Time-series data
- Privacy-preserving data mining

Syllabus

Sept 2	**Introduction to data mining**
Sept 9	Basic algorithms and prototype problems
Sept 14, 16	Frequent itemsets and association rules
Sept 21, 23, 28, 30	Clustering algorithms
Oct 5, 7	Dimensionality reduction
Oct 12	Holiday
Oct 14	Midterm exam
Oct 19, 21, 26, 28	Classification
Nov 2, 4, 9, 11	Link-analysis ranking
Nov 16, 18, 23	Recommendation systems
Dec 1, 3	Time series analysis
Dec 8, 10	Privacy-preserving data mining
Week starting Dec 14	Final exam; exact date to be determined

Course workload

- Three programming assignments (30%)
- Three problem sets (20%)
- Midterm exam (20%)
- Final exam (30%)
- Late assignment policy: 10% per day up to three days; credit will be not given after that
- Incompletes will not be given

Textbooks

- D. Hand, H. Mannila and P. Smyth: Principles of Data Mining. MIT Press, 2001
- Jiawer Han and Micheline Kamber: Data Mining: Concepts and Techiques. Second Edition. Morgan Kaufmann Publishers, March 2006
- Toby Segaran: Programming Collective Intelligence: Building Smart Web 2.0 Applications. O'Reilly
- Research papers (pointers will be provided)

Prerequisites

- Basic algorithms: sorting, set manipulation, hashing
- Analysis of algorithms: O-notation and its variants, perhaps some recursion equations, NP-hardness
- Programming: some programming language, ability to do small experiments reasonably quickly
- Probability: concepts of probability and conditional probability, expectations, binomial and other simple distributions
- Some linear algebra: e.g., eigenvector and eigenvalue computations

Above all

- The goal of the course is to learn and enjoy
- The basic principle is to ask questions when you don’t understand
- Say when things are unclear; not everything can be clear from the beginning
- Participate in the class as much as possible

Introduction to data mining

- Why do we need data analysis?
- What is data mining?
- Examples where data mining has been useful
- Data mining and other areas of computer science and statistics
- Some (basic) data-mining tasks

Why do we need data analysis

- Really really lots of raw data data!!
  - Moore's law: more efficient processors, larger memories
  - Communications have improved too
  - Measurement technologies have improved dramatically
  - It possible to store and collect lots of raw data
  - The data-analysis methods are lagging behind
- Need to analyze the raw data to extract knowledge

The data is also very complex

- Multiple types of data: tables, time series, images, graphs, etc
- Spatial and temporal aspects
- Large number of different variables
- Lots of observations → large datasets

Example: transaction data

- Billions of real-life customers: e.g., walmart, safeway customers, etc

- Billions of online customers: e.g., amazon, expedia, etc.

Example: document data

- Web as a document repository: 50 billion of web pages
- Wikipedia: 4 million articles (and counting)
- Online collections of scientific articles

Example: network data

- Web: 50 billion pages linked via hyperlinks
- Facebook: 200 million users
- MySpace: 300 million users
- Instant messenger: ~1billion users
- Blogs: 250 million blogs worldwide, presidential candidates run blogs

Example: genomic sequences

- http://www.1000genomes.org/page.php
- Full sequence of 1000 individuals
- 310^9 nucleotides per person → 310^12 nucleotides
- Lots more data in fact: medical history of the persons, gene expression data

Example: environmental data

- Climate data (just an example)

http://www.ncdc.gov/oa/climate/ghcn-monthly/index.php

- “a database of temperature, precipitation and pressure records managed by the National Climatic Data Center, Arizona State University and the Carbon Dioxide Information Analysis Center”

- “6000 temperature stations, 7500 precipitation stations, 2000 pressure stations”

We have large datasets…so what?

- Goal: obtain useful knowledge from large masses of data
- "Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data analyst"
- Tell me something interesting about the data; describe the data
- Exploratory analysis on large datasets

What can data-mining methods do?

- Extract frequent patterns
  - There are lots of documents that contain the phrases "association rules", "data mining" and "efficient algorithm"
- Extract association rules
  - 80% of the walmart customers that buy beer and sausage also buy mustard
- Extract rules
  - If occupation=PhD student then income < 20K

What can data-mining methods do?

- Rank web-query results
  - What are the most relevant web-pages to the query: “Student housing BU”?
- Find good recommendations for users
  - Recommend amazon customers new books
  - Recommend facebook users new friends/groups
- Find groups of entities that are similar (clustering)
  - Find groups of facebook users that have similar friends/interests
  - Find groups amazon users that buy similar products
  - Find groups of walmart customers that buy similar products

Goal of this course

- Describe some problems that can be solved using data-mining methods
- Discuss the intuition behind data-mining methods that solve these problems
- Illustrate the theoretical underpinnings of these methods
- Show how these methods can be useful in practice

Data mining and related areas

- How does data mining relate to machine learning?
- How does data mining relate to statistics?
- Other related areas?

Data mining vs machine learning

- Machine learning methods are used for data mining
  - Classification, clustering
- Amount of data makes the difference
  - Data mining deals with much larger datasets and scalability becomes an issue
- Data mining has more modest goals
  - Automating tedious discovery tasks, not aiming at human performance in real discovery
  - Helping users, not replacing them

Data mining vs. statistics

- “tell me something interesting about this data” – what else is this than statistics?
  - The goal is similar
  - Different types of methods
  - In data mining one investigates lot of possible hypotheses
  - Data mining is more exploratory data analysis
  - In data mining there are much larger datasets→ algorithmics/scalability is an issue

Data mining and databases

- Ordinary database usage: deductive
- Knowledge discovery: inductive
  - Inductive reasoning is exploratory
- New requirements for database management systems
- Novel data structures, algorithms and architectures are needed

Data mining and algorithms

- Lots of nice connections
- A wealth of interesting research questions
- We will focus on some of these questions later in the course

Some simple data-analysis tasks

- Given a stream or set of numbers (identifiers, etc)
- How many numbers are there?
- How many distinct numbers are there?
- What are the most frequent numbers?
- How many numbers appear at least K times?
- How many numbers appear only once?
- etc

Finding the majority element

- A neat problem
- A stream of identifiers; one of them occurs more than 50% of the time
- How can you find it using no more than a few memory locations?
- Suggestions?

Finding the majority element (solution)

- A = first item you see; count = 1
- **for** each subsequent item B

```
    if (A==B) count = count + 1
    else {
            count = count - 1
             if (count == 0) {A=B; count = 1}
           }
 endfor
```

- Why does this work correctly?

Finding the majority element (solution and correctness proof)

- A = first item you see; count = 1
- **for** each subsequent item B

```
    if (A==B) count = count + 1
    else {
            count = count - 1
            if (count == 0)
                {A=B; count = 1}
           }
 endfor
```

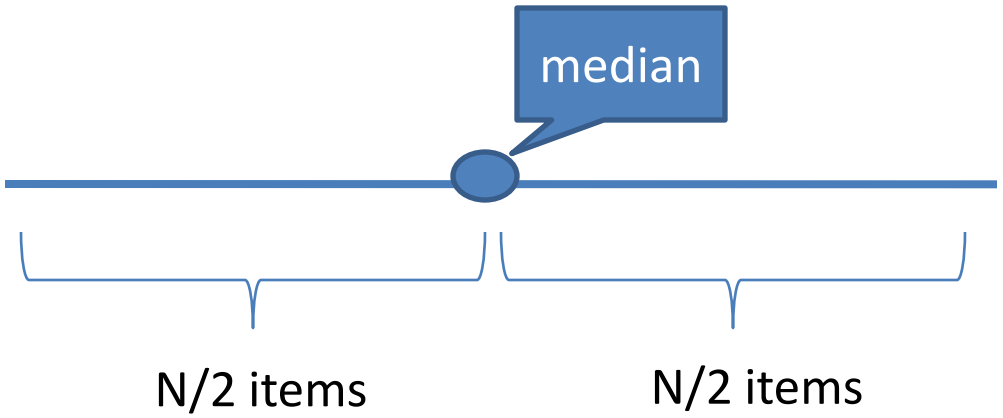
- **Basic observation:** Whenever we discard element **u** we also discard a unique element **v** different from **u**

Finding a number in the top half

- Given a set of N numbers (N is very large)
- Find a number x such that x is *likely* to be larger than the median of the numbers
- Simple solution
  - Sort the numbers and store them in sorted array A
  - Any value larger than A[N/2] is a solution
- Other solutions?

Finding a number in the top half ***efficiently***

- A solution that uses small number of operations
  - Randomly sample K numbers from the file
  - Output their maximum



- Failure probability (1/2)^K

Sampling a sequence of items

- **Problem:** Given a sequence of items P of size N form a ***random sample*** S of P that has size n ($n<N$) → sampling without replacement

- What does random sample mean?
  - Every element in P appears in S with probability n/N
  - Equivalent as if you generate a **random permutation** of the N elements and take the **first** n elements of the permutation

Sampling algorithm v.0.

- R = {} // empty set
- **for** i=1 **to** n
  rnd = Random([1…N])
  while (rnd in R)
  rnd = Random([1…N])
  endwhile
  R = R U {rnd}
  S[i] = P[rnd]

  endfor
  return S

- Running time?

- The algorithm assumes that S and its size are known in advance!

Sampling algorithm v.1.

- **Step 1:** Create a random permutation π of the elements in P
- **Step 2:** Return the first n elements of the permutation, $S[i] = \pi[i]$, for $(1 \leq i \leq n)$

Can you do **Step 1** in linear time?

You can do **Step 2** in linear time☺

Creating a random permutation in linear time

- **for** i=1…N **do**

  j = Random([1…i-1])

  swap P[i] with P[j]

  endfor
- Is this really a random permutation? (see CLR for the proof)
- It runs in linear time

Sampling algorithm v.1.

- **Step 1:** Create a ***random permutation*** π of the elements in P
- **Step 2:** Return the first n elements of the permutation, $S[i] = \pi[i]$, for $(1 \leq i \leq n)$

- The algorithm works in ***linear time O(N)***
- The algorithm assumes that P ***is known in advance***
- The algorithm makes ***2 passes*** over the data

Sampling algorithm v.2.

- **for** i = 1 to n
  S[i] = P[i]
  endfor

- t = n+1

- **while** P has more elements
  rnd = Random([1…t])
  if (rnd <= n)
  {S[rnd] = P[t]}
  t = t + 1
  endwhile

Correctness proof

- At iteration **t+1** a **new** item is included in the sample with probability **n/(t+1)**
- At iteration **(t+1)** an **old** item is kept in the sample with probability **n/(t+1)**
  - **Inductive argument:** at iteration **t** the old item was in the sample with probability **n/t**
  - **Pr(old item in sample at t+1) = Pr(old item was in sample at t) x (Pr(rnd >n) + Pr(rnd<=n) x Pr(old item was not chosen for eviction))**

= n/t((t+1-n)/(t+1) + n/(t+1)x(1-1/n))

= n/(t+1)

Sampling algorithm v.2.

- **for** i = 1 to n
  S[i] = P[i]
  endfor

- t = n+1

- **while** P has more elements {
  rnd = Random([1…t])
  if (rnd <= n)
  {S[rnd] = P[t]}
  t = t + 1
  endwhile

Advantages

- Linear time

- **Single pass** over the data

- **Any time;** the length of the sequence need not be known in advance