Data Mining: Models and Algorithms

A SOFT INTRODUCTION

By

Andrea Clementi

Slides inspired by the Book: **Algorithms for Massive Data**

by Jure Leskovec, Anand Rajaraman, Jeff Ullman available at http://www.mmds.org

Data Mining

- **Strategic Goal**

Find

  - the best Hardware
  - the best **Models**,
  - the best **Algorithms**
  - the best Software

to solve problems on **large** and/or **complex** Input Data Sets in Science, Networking, Healthcare, E-commerce, Goverment, etc.

Models for Large/Complex Data Sets

- **Stage (i).** Transform the real input Data Set into a formal **model/structure** I so that the original real problem efficiently reduces to a well-defined computational Task T on input I
- **Stage(ii).** Find the best algorithm to solve T_P on input I

- **Typical Scenario in Data Mining:**
  - Stage (i) is often the most challenging issue
  - Once Stage (i) is solved, Stage (ii) only requires standard methods

Real Data-Mining Problem: Phishing Emails

Algorithmic Solution:

- (i) Classify the received emails into two subsets: Phishing / Non-Phishing
- **(ii)*** Extract those phishing words (or phrases) that appear unusually often in Phishing (e.g. "Nigerian Prince", "send money")
- **(iii)**** Assign positive weights to phishing words and negative weights to the others.
- (iv) Algorithm: for each incoming email, compute the sum of its word weights. If this sum is greater than a given threshold, then set the email as Phishing, o.w. set to Non-Phishing

Challenging Key-Tasks in Data Mining

- **Step (ii):** Define a suitable Statistical Model **M** (i.e. a Probability Distribution) on the real (raw) input data (*emails*) so that the hidden information we search for emerges as a likely event in **M**. Then extract this information (e.g. the k-most frequent words in Phishing) **efficiently** and/or in a **dynamic/streaming** fashion
- **Step (iii):** assign the right weights to words so that the total sum of an incoming email is proportional to the probability it is phishing

Statistical Modeling: *Informal Definition*

- Construct an underlying probability distribution from which the visible raw data are sampled
- **Example:** the raw data set is a set D of numbers. *A* Statistical Model/Hypothesis for D is to assume that numbers are sampled according to a Gaussian Distribution over D. Then, mean and standard deviation completely characterize this Distribution and would become the Model of the Input Data

Adversarial Data Models: *Informal Definition*

- Other typical scenarios in ***Data Mining*** adopt worst-case hypothesis:
- The input Data Set, the target information should be extrated from, is managed by an adversarial source that generates the data with the goal to minimize the efficiency of the algorithmic solution or, even worse, its validity.
- **Example:** Computing (and updating) the **number of 1's** in the last window of length **N** over an infinite *Stream* of bits governed by an adversary that choose the next bit as a function of the previous algorithm's choices

Data Mining and Machine Learning

- *Machine Learning* (ML) is a possible, often powerful approach for ***Data Mining***.
- ML **approach**: use part of the input Data Set as a *Training Set*, to train ML systems such as *Bayes Nets*, *Support-Vector Machine*, *K-Means*, *Hidden Markov Models*, etc etc.
- *Main General Question*: When do *ML* approaches work well?

- Informal Answer: *ML* turns out to be a good approach in situation where it is not possible to define a *clean* objective function over the Data Set, i.e., whenever we don't know what the input data say about the Problem we are trying to solve

ML approaches: good situations

- Movie Ratings (The *Netflix Problem*): Predict the user rating of movies.
- **Hard Tasks*:** Select those features of a movie that make a user like or dislike it; Formalize them and provide an efficient checking system.
- ML approach proposes efficient algorithms that take samples of user ratings and make predictions with a **good** level of approximation.

* it is not clear which features of a movie play here a crucial role and how to formalize them.

ML approach: Bad Situations

- ML is often not competitive when the objective function is clear and well-defined over the Data Set.
- **Example:** Locate people's CV on the WEB: features of a CV page can be well defined by detecting the presence of typical words.
- Find Triangles in Social Networks: There are efficient, local algorithms for this task working on large and/or dynamic graphs.

- Another ML issue: Transparency

Statistical Limits of Data Mining

- Typical goal in Data-Mining Problems: discover unusual events hidden within massive Data Sets.
- **Warning:**

Overzealus use of Data Mining - Bonferroni's Principle

" If the Data Set (or the Data *Process*) collects a large number of items from different sources in random enough fashion, then *unusual* events may have no particular meanings: they are just *statistical artifacts* likely to happen "

Bonferroni's Principle: Informal Statement

Suppose you look for a specific Event within the input Data Set DS. Then….

You can expect Event to occur, even if S is completely random, and the number of occurrences of Event will grow as the size of grows DS.

These occurrences are "bogus": they have no cause other than that random data will always have some number of unusual features that look significant but aren't.

A fundamental theorem of Statistics, known as the *Bonferroni Correction,* gives a statistically sound way to avoid most of these bogus-positive responses to a search through the data.

Example of «*Bogus*»: *Gangs* in Social Networks

Mining Information from a Social Network

- **The Graph Model** (simplified version) **G(V,E)**: A (large) set V of *n* (*potentially-criminal) agents* that use to visit a (large) set H of *h* public locations (hotels, bar, restaurants, airports, etc) in a large town/region:
  - An edge (u,v) exists if u and v visit the same location in H in the same day over a sequence of $T \gg 0$ days
- **Model Hypothesis** (simplified version): The agent visiting process over the public locations (hotels, bars, airport, banks, etc) are sufficiently random and uniform

- **Problem (informal):** Detect possible «Gangs», namely *Cliques* in **G**

Gangs in Social Networks: Formal Setting

Question: Which maximal clique size do we expect in $G(V,E)$?

Answer: For any agent pair (u,v), we get $Pr[(u,v) \in E] \simeq T/h \triangleq p$

For a subset $S \subseteq V$ of size s, we get:

$Pr[S \text{ is a clique}] \simeq p^{\wedge}(s^2/2)$, so:

$E[\# \text{ cliques of size } s] = C(n,s) * p^{\wedge}(s^2/2) \simeq (n/e*s)^s * p^{\wedge}(s^2/2)$ (*)

Now, observing a big space-time system, e.g. $n \simeq 10^7$, $h \simeq 10^4$; $T \simeq 10^3 \rightarrow p \simeq 1/10$, From (*), we can expect a *large* numbers of cliques of size

$s \simeq 10$

So, a Gang of 10 guys that *all* have met each other at least once is not a surprising event! please don't call FBI! ☺

H = size of location set;
T = number of days
Hyp: H >> T

A *Computational Lens* for Data Mining

- The **Computer Science** approach to Data Mining is <u>Algorithmic</u>:

  Given a Data Set (or a Data Process) DS, provide an ***efficient*** *Algorithm* (& *Data Structure*) to answer some complex queries on DS

<u>**Example.**</u> Given an infinite stream DS of integers, maintain the ***average value*** of DS (seen so far) and its standard deviation (or its surprising number)

Remark. The algorithm **should not rely** on any *statistical* hypothesis I on

DS but if I holds, the computed values will be consistent to I

Computational Lens to Data Mining: Typical Tasks

1. **Computational Modeling** of (Input) Data Processes (e.g. Streaming)

2. **Summarizing** a Data Set

3. **Extracting** the most prominent Features of the Data Set and discarding the rest

Data Streams

- In many **Data-Mining** situations, we do not know the *entire* Data Set *in advance*
- **Stream Management** is important when the input rate is controlled **externally**:
  - *Search-Engines* queries
  - *Twitter* or *Facebook* status updates
  - IP addresses managed by Servers
- We can think of the Data Set as ***infinite*** and ***non-stationary*** (the *distribution* changes over time)

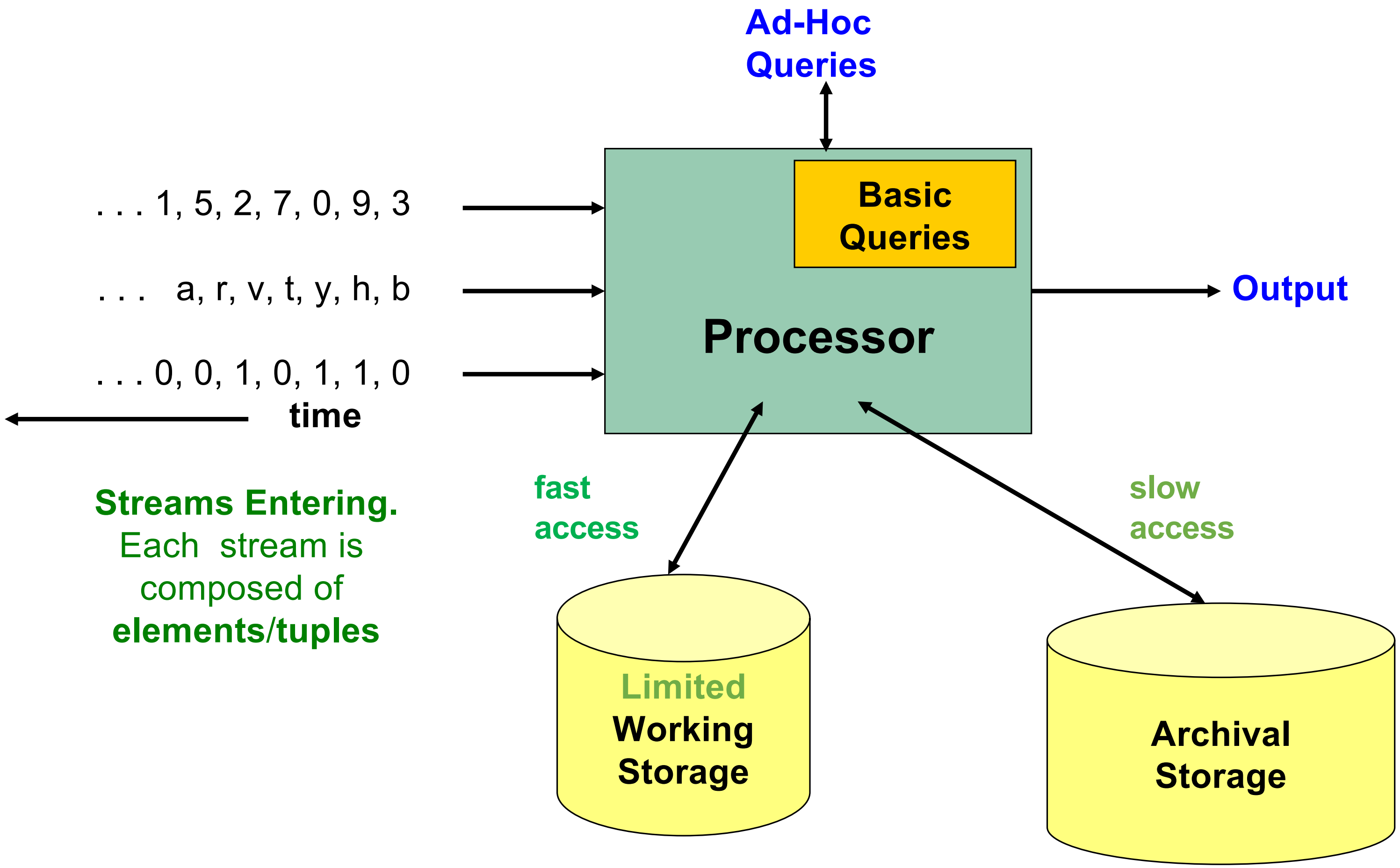
The Data-Stream Model

- Input **elements** enter at a ***rapid rate***, at one or more input ports (i.e., **Streams**)
  - **e.g. elements** of the **Streams** $\equiv$ **tuples**

- The system cannot store the entire **Stream S** accessibly.
- Only <u>short</u> *sketches* of the S can be maintained and updated

- **Q:** How do you answer <u>*critical queries*</u> about S using a *limited amount of memory*?

General Stream Processing Model



Problems on Data Streams

- Types of **queries** on a Data Stream (DS):
  - **Sampling** data from a DS: Construct a ***random*** sample
  - **Filtering** a DS: Select elements with property ***x***
  - **Counting** *distinct* elements in DS: Number of distinct elements in the last *k* elements of the DS
  - **Estimating *Moments:*** Estimate *average* and *standard deviation* of last *k* elements in DS
  - **Finding** the *k* most-*frequent* elements seen so far in DS

2) Summarization

- **Algorithmic Query**: Given a large Data Set DS, provide a short ***sketch*** H(S) that effectively *summarizes* the key features of DS

- **Optimization Challenges**:
  - **Minimize** size |H(S)| (it should be asymptotically much smaller than |DS|)
  - H(S) must *effectively* approximates the *key features* of DS
  - H(S) must allow efficient *dynamic* updating (*Dynamic Data Structure*)

2) Summarization: Popular Examples

2.a) The *Page-Rank Algorithm* for the WEB

Main Result: The entire, complex WEB structure can be *summarized* by single value (the rank) for each page x

rank(..) ≃ (Stationary) Probability Distribution of a ***Random Walk*** on the WEB graph

rank(x) ≃ It is an effective **measure** of the *importance* of page x **w.r.t. the entire** WEB

2) Summarization: Popular Examples

1.b) Data Clustering

- The input Data Set DS is modeled as a set of points in a D-*Dimensional Metric Space* (e.g. R^D) with metric (distance) $d: R^D \times R^D \rightarrow R^+$
- Summarization **Algorithm:**
  - Points that are close w.r.t. d(.,.) are assigned to the same ***Cluster***. We expect a *small* number of ***Clusters*** forming a *partition* of DS
  - Each **Cluster** is in turn summarized by a ***sketch***: e.g. its *centroid* and the *avg distance* of its points from the centroid

- **Output**: The set of **Cluster** ***sketches*** as the Summary of the entire DS

3). Feature Extraction

- **Feature Extraction** from massive Data Sets often relies on selecting the strongest statistical dependencies (correlations) among the items and using only these dependencies to represent the entire Data Set.