Introduction to Data Science

Big Data

Arijit Mondal

Dept. of Computer Science & Engineering

Indian Institute of Technology Patna

arijit@iitp.ac.in

Introduction

- Computer Science is a science of abstraction-creating the right model for a problem and devising the appropriate mechanizable techniques to solve it. — A. Aho and J. Ullman

Introduction

- Computer Science is a science of abstraction-creating the right model for a problem and devising the appropriate mechanizable techniques to solve it. — A. Aho and J. Ullman

- Problem space

Introduction

- Computer Science is a science of abstraction-creating the right model for a problem and devising the appropriate mechanizable techniques to solve it. — A. Aho and J. Ullman

- Problem space
  - Problems with polynomial time solutions — considered as simple problems, easy to describe and understand, computation requirement is relatively less
    - Examples — sorting, shortest path, etc.

Introduction

- Computer Science is a science of abstraction-creating the right model for a problem and devising the appropriate mechanizable techniques to solve it. — A. Aho and J. Ullman

- Problem space
  - Problems with polynomial time solutions — considered as simple problems, easy to describe and understand, computation requirement is relatively less
    - Examples — sorting, shortest path, etc.
  - Problems that are challenging for human beings but relatively easy for computers — can be described formally, usually takes exponential computation time
    - Examples — travelling salesman problem, chess, graph coloring, etc.

Introduction

- Computer Science is a science of abstraction-creating the right model for a problem and devising the appropriate mechanizable techniques to solve it. — A. Aho and J. Ullman

- Problem space
  - Problems with polynomial time solutions — considered as simple problems, easy to describe and understand, computation requirement is relatively less
    - Examples — sorting, shortest path, etc.
  - Problems that are challenging for human beings but relatively easy for computers — can be described formally, usually takes exponential computation time
    - Examples — travelling salesman problem, chess, graph coloring, etc.
  - Problems that are easy for human beings but relatively difficult for computers — cannot be described formally,
    - Examples — object identification, face recognition, etc.

Introduction

- Computer Science is a science of abstraction-creating the right model for a problem and devising the appropriate mechanizable techniques to solve it. — A. Aho and J. Ullman

- Problem space
  - Problems with polynomial time solutions — considered as simple problems, easy to describe and understand, computation requirement is relatively less
    - Examples — sorting, shortest path, etc.
  - Problems that are challenging for human beings but relatively easy for computers — can be described formally, usually takes exponential computation time
    - Examples — travelling salesman problem, chess, graph coloring, etc.
  - Problems that are easy for human beings but relatively difficult for computers — cannot be described formally,
    - Examples — object identification, face recognition, etc.
  - Undecidable problems — that cannot be solved on computers
    - Examples — infinite loop detection

What is big data?

- How much data is really big?

What is big data?

- How much data is really big?
- Let us look some statistics — https://www.internetlivestats.com/
  - Twitter: 600 million tweets per day
  - Facebook: 600 terabytes of incoming data per day from 1.6 billion active users
  - Google: 3.5 billion search queries each day
  - Instagram: 52 million new photos each day
  - Apple: 130 billion total app downloads
  - Netflix: 125 million hours of video streaming daily
  - Email: 205 billion messages per day

Big data

- Extract meaningful / relevant / useful information from such huge data
- Challenges are
  - Store — need large storages / databases
  - Manage — efficient access (read / write) of the databases
  - Analyze — need methodologies to identify important properties

Big data

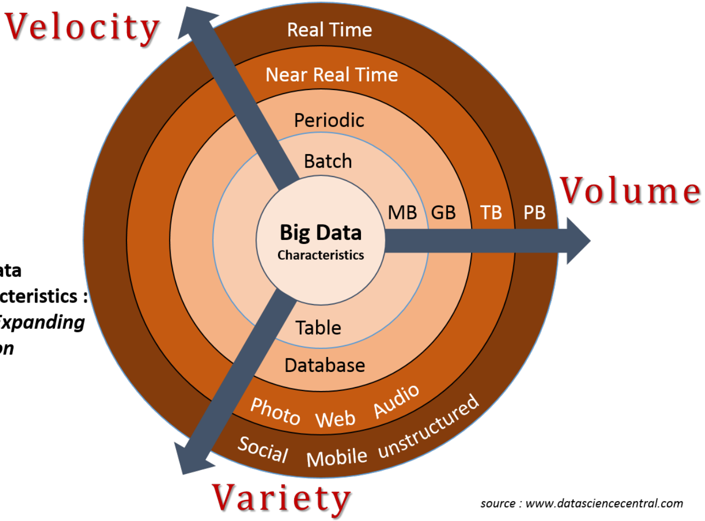
- Extract meaningful / relevant / useful information from such huge data
- Challenges are
  - Store — need large storages / databases
  - Manage — efficient access (read / write) of the databases
  - Analyze — need methodologies to identify important properties

Big data ≈ Data mining ≈ Data science ≈ Predictive analytics

Issues in analyzing the data

- Consider the task of measuring popular opinion from the posts in a social media platform
  - Unrepresentative participation
  - Spam and machine generated content
  - High redundancy
  - Susceptible to temporal bias

Three Vs for big data



Big Data Characteristics : ***Ever Expanding horizon***

source : www.datasciencecentral.com

image source: internet

Achieving solution for big data

- Algorithmics / methodologies for big data
  - Big Oh analysis
  - Hashing
  - Exploiting storage hierarchy
  - Streaming data
  - Filtering
  - Sampling, etc.
- Support from hardware
  - Parallelism
  - Cloud computing
  - MapReduce, etc.

Big Oh analysis

- Random Access Machine
  - Each simple operation takes exactly one step
  - Each memory operation takes exactly one step
- Examples
  - Nearest neighbor — $\mathcal{O}(p \cdot n)$
  - Closest pair of points — $\mathcal{O}(d \cdot n^2)$
  - Matrix multiplication — $\mathcal{O}(n^3)$
  - Adding two numbers — $\mathcal{O}(1)$
  - Binary search — $\mathcal{O}(\log n)$
  - Merge sort — $\mathcal{O}(n \log n)$

$$\boxed{\mathcal{O}(1) \ll \mathcal{O}(\log n) \ll \mathcal{O}(n) \ll \mathcal{O}(n \log n) \ll \mathcal{O}(n^2) \ll \mathcal{O}(n^3)}$$

Hashing

- This technique can reduce the computation complexity
- A hash function h takes an object x and maps to a specific integer

  $h(x)$ and $x = y \Rightarrow h(x) = h(y)$
- Turning a vector of numbers into a single representative number

  $$h(x) = \sum_{i=0}^{n-1} \alpha^{n-(i+1)} x_i \pmod{m}$$
- Examples
  - Dictionary maintenance
  - Frequency counting
  - Duplicate removal
  - Canonization
  - Cryptographic hashing

Storage hierarchy

- Big data algorithms are often storage-bound or bandwidth-bound rather compute bound
- Cost of waiting around data to arrive exceeds algorithmically manipulation time
- Need to exploit hierarchy of storage for trade-off between performance and cost
- Types of memory
  - Cache memory
  - Main memory
  - Main memory of another machine
  - Disk storage

Processing streaming data

- Data may not be stored forever
- In many cases, data can be seen only once

Processing streaming data

- Data may not be stored forever
- In many cases, data can be seen only once

- Example
  - Computation of average of streaming number is easy, store number of element and the running sum

Processing streaming data

- Data may not be stored forever
- In many cases, data can be seen only once

- Example
  - Computation of average of streaming number is easy, store number of element and the running sum
  - Computation of variance can have issues with $\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$

Processing streaming data

- Data may not be stored forever
- In many cases, data can be seen only once

- Example
  - Computation of average of streaming number is easy, store number of element and the running sum
  - Computation of variance can have issues with $\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$
  - The above issue can be resolved if we use $\sigma^2 = \left(\frac{1}{n}\sum_{i=1}^{n}(x_i)^2\right) - (\bar{x})^2$

Filtering and Sampling

- Filtering is the process of selecting relevant subset of the data based on specific task
- Removal of data is not due to error but distracting the main goal
  - Suppose, a language model needs to developed for Uttar Pradesh
  - Tweets posted in Tamil, Spanish, etc. are not meaningful
- Sampling is the process of selecting appropriate size subset in an arbitrary manner
  - Right sizing training data
  - Data partitioning
  - Exploratory data analysis and visualization

Sampling strategies

- Sampling can introduces bias in the data
  - Temporal bias
  - Lexicographic bias
  - Numerical bias, etc.

Sampling strategies

- Sampling can introduces bias in the data
  - Temporal bias
  - Lexicographic bias
  - Numerical bias, etc.
- Uniform sampling strategy can help in many cases
  - Desired number of records can be selected
  - Quick and reproducible
  - One can construct multiple disjoint samples

Sampling strategies

- Sampling can introduces bias in the data
  - Temporal bias
  - Lexicographic bias
  - Numerical bias, etc.
- Uniform sampling strategy can help in many cases
  - Desired number of records can be selected
  - Quick and reproducible
  - One can construct multiple disjoint samples
- Random sampling approach
  - Items are selected with some probability distributions
  - Generally non-reproducible
  - Multiple random samples may not be disjoint

Parallelism

- Needs support from both software and hardware
- A task can be splitted and subtasks are allotted to different computing units
  - Parallel processing
  - Distributed processing

Parallelism

- Needs support from both software and hardware
- A task can be splitted and subtasks are allotted to different computing units
  - Parallel processing
  - Distributed processing
- To understand complexity of the problem, consider a social gathering of following scenarios

Parallelism

- Needs support from both software and hardware
- A task can be splitted and subtasks are allotted to different computing units
  - Parallel processing
  - Distributed processing
- To understand complexity of the problem, consider a social gathering of following scenarios
  - 1 person: A date!

Parallelism

- Needs support from both software and hardware
- A task can be splitted and subtasks are allotted to different computing units
  - Parallel processing
  - Distributed processing
- To understand complexity of the problem, consider a social gathering of following scenarios
  - 1 person: A date!
  - > 2 persons: Dinner among friends

Parallelism

- Needs support from both software and hardware
- A task can be splitted and subtasks are allotted to different computing units
  - Parallel processing
  - Distributed processing
- To understand complexity of the problem, consider a social gathering of following scenarios
  - 1 person: A date!
  - > 2 persons: Dinner among friends
  - > 10 persons: Group meeting

Parallelism

- Needs support from both software and hardware
- A task can be splitted and subtasks are allotted to different computing units
  - Parallel processing
  - Distributed processing
- To understand complexity of the problem, consider a social gathering of following scenarios
  - 1 person: A date!
  - > 2 persons: Dinner among friends
  - > 10 persons: Group meeting
  - > 100 persons: Wedding dinner

Parallelism

- Needs support from both software and hardware
- A task can be splitted and subtasks are allotted to different computing units
  - Parallel processing
  - Distributed processing
- To understand complexity of the problem, consider a social gathering of following scenarios
  - 1 person: A date!
  - > 2 persons: Dinner among friends
  - > 10 persons: Group meeting
  - > 100 persons: Wedding dinner
  - > 1000 persons: Community festival

Parallelism

- Needs support from both software and hardware
- A task can be splitted and subtasks are allotted to different computing units
  - Parallel processing
  - Distributed processing
- To understand complexity of the problem, consider a social gathering of following scenarios
  - 1 person: A date!
  - > 2 persons: Dinner among friends
  - > 10 persons: Group meeting
  - > 100 persons: Wedding dinner
  - > 1000 persons: Community festival
  - > 10000 persons: Political rally

Parallelism

- Needs support from both software and hardware
- A task can be splitted and subtasks are allotted to different computing units
  - Parallel processing
  - Distributed processing
- To understand complexity of the problem, consider a social gathering of following scenarios
  - 1 person: A date!
  - > 2 persons: Dinner among friends
  - > 10 persons: Group meeting
  - > 100 persons: Wedding dinner
  - > 1000 persons: Community festival
  - > 10000 persons: Political rally
- Challenges of parallelization and distributed computing
  - Coordination
  - Communication
  - Fault tolerance
- Cloud computing services may be explored

Grid search

- Easiest way to exploit parallelism is to have independent execution on independent data
- Each independent run can a different setting of hyper-parameters
- Many hyper-parameters need to be selected
  - Learning rate
  - Epoch
  - k in clustering
  - Batch size
  - Depth of network
  - Weight decay, etc.

Typical big data problem

- A large scale data-science task will involve the following primarily
  - Iterate over large number of items
  - Extract something of interest from each item
  - Aggregate intermediate results
  - Produce final output

MapReduce

- MapReduce paradigm for distributed computing has spread widely through open-source implementations like Hadoop and Spark
  - Simple parallel programming model
  - Straight forward scaling to hundreds/thousands of machines
  - Fault tolerant through redundancy

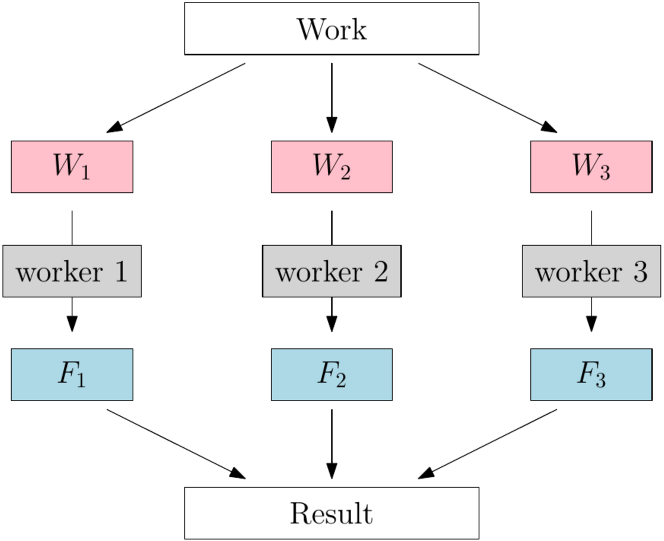


image source: The Data Science Design Manual, S.Skiena

Challenges in parallelization

- How do we assign tasks to machines?
- What if we have more tasks than machines?
- What if machines need to share partial results?
- How do we aggregate partial results?
- How do we know all the machines have finished tasks?
- What if machines fail?

MapReduce

- Scale out: large shared-memory is a big concern
- Move processing to the data: clusters have limited bandwidth
- Process data sequentially, avoid random access: seeks are expensive, disk throughput is reasonable
- Seamless scalability

- Hadoop has two primary subsystems
  - Hadoop / MapReduce — distributed big data processing infrastructure (abstract / paradigm, fault-tolerant, schedule, execution)
  - HDFS (Hadoop Distributed File System) — fault-tolerant, high-bandwidth, high availability distributed storage

MapReduce (contd.)

- Programmer needs to specify only two functions — Map and Reduce
  - Map — reads in a file and produces key-value pairs
  - Reduce — aggregates and processes the key-value pairs having the same key
- MapReduce runtime system
  - Processor scheduling
  - Data distribution
  - Synchronization
  - Error and fault tolerance

Example of MapReduce

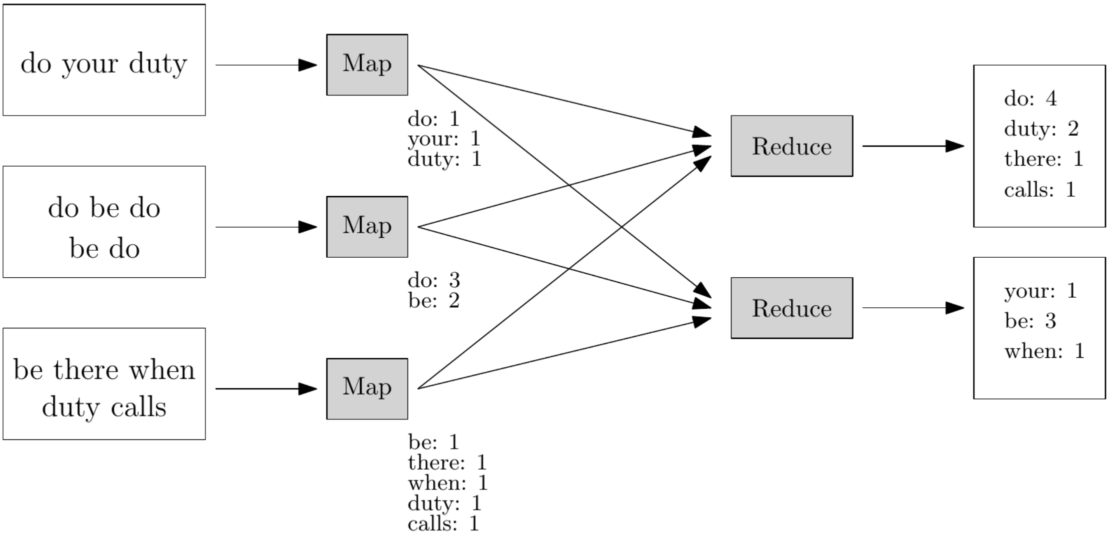


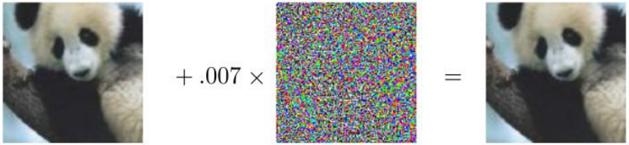
image source: The Data Science Design Manual, S.Skiena

Hadoop Distributed File System

- Store data on the local disks of nodes in the cluster, because RAM may not be sufficient to hold all the data in memory
- Disk access is typically slow, but disk throughput is reasonable, so linear scans through files are fine
- Replicate everything multiple times for reliability on commodity hardware

Trustworthy systems

- AI systems have achieved good performance level such that it can be deployed in practical field
  - Object recognition can help cars to see
  - Personalized voice assistant - alexa / siri
  - Computer can beat the best alphago player
- AI has a very big role to play in many different domains such as medical, law, etc.
- However AI systems are brittle and unfair in many cases
  - Assign small noise to traffic signal can change the interpretation
  - A small noise can make a benign tumor to be reported as malignant



Panda Noise Gibbon

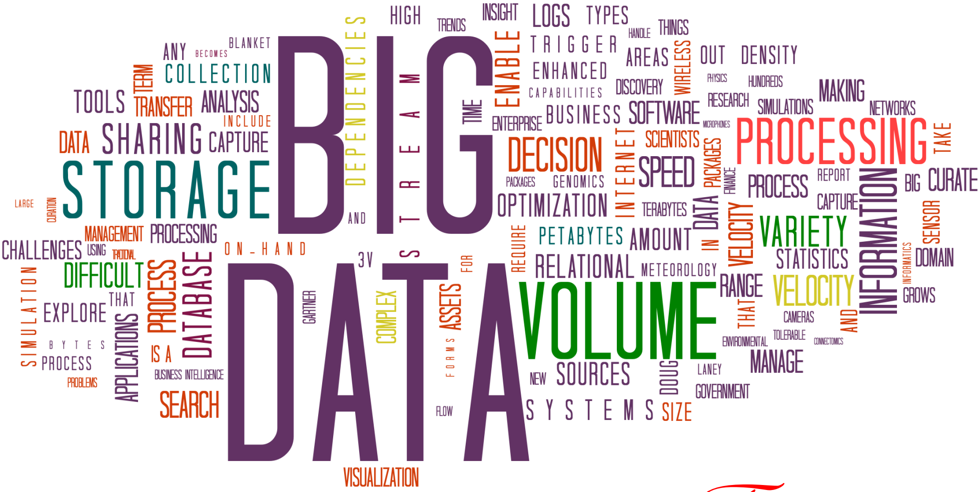
Image source: Deep Learning, Ian Goodfellow et.al.

Properties for trustworthy systems

- Accuracy — How well does the AI system do on new (unseen) data compared to data on which it was trained and tested
- Robustness — How sensitive is the system's outcome to a change in the input
- Fairness — Are the system outcome unbiased?
- Accountability — Who or what is responsible for the system's outcome
- Transparency — Is it clear to external observer how the system's output produced
- Interpretability — Can the system's output be justified with an explanation that a human can understand
- Ethical — Was the data collected in ethical manner? Will the system's outcome be used in an ethical manner?

Summary

- Big data requires knowledge from multiple domains
  - Data bases
  - Computer architecture
  - Machine learning
  - Handling of large data
- Scope of big data is huge
  - Large data is a huge resource
  - Mining or analyzing such data can provide more insights
  - Better predictive models can be build
  - Many interesting applications can be developed
  - Needs to ensure reliability, safety, ethics, etc.



Thank you!

image source: internet