Ashlyn Goh

Computational Data Science

Week 1 – Intro to Big Data

Introduction to Data Science

KDD Data Process:

- Understand goals
- Create dataset for study
- Data cleaning & preprocessing
- Data reduction & projection
- Choose Data Analytics task & algorithms
- Use algorithms to perform task
- Iterate through if necessary

SEMMA Methodology:

- **S**ample from dataset
- **E**xplore dataset (visualization)
- **M**odify data (create/transform features)
- **M**odel
- **A**ssess: compare models, test datasets, evaluate reliability/usefulness

“No Free Lunch Theorem” -> no one algorithm works best for every problem, especially relevant for supervised learning

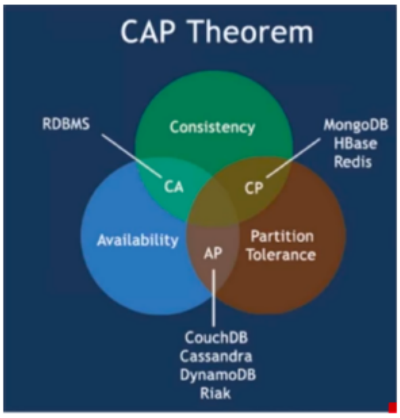
Big Data

- **Why:** Increase of data
- **What:** Too big to be processed on a single machine
    - Challenges: Data is created fast, data from different sources in different formats
- 3V’s
    - **Volume**: size of data
        - Most data are useful, so want to keep as much as possible
    - **Variety**: data comes from different sources & different formats
        - Estimated 90% of data is unstructured or semi-structured
        - Types of data:
            - Structured: e.g. SQL
            - Unstructured e.g. text, numbers, files can be all mixed -> Hadoop
            - Semi-structured: may have a certain structure but not all data has necessarily identical structure -> xml, json -> MongoDB
    - **Velocity**: speed at which it is being generated & needs to be able to be processed
    - 4^{th} V? **veracity**: trustworthiness of data
        - Various data uncertainty & unreliability
        - Imprecision of data e.g. text message can have double meaning
- CAP theorem:
  Impossible for a distributed data store to simultaneously provide more than 2 out of the following 3 guarantees:

- **Consistency**:
  Every read received the most recent write or an error
- **Availability**:
  Every request receives a non-error response (without guarantee that it's the most recent write)
- **Partition Tolerance**:
  System continues to operate despite an arbitrary # of messages being dropped (or delayed) by network between nodes (Partition = communications break)

- **Hadoop** -> **AP**. Consistency not supported because only name node has the information of where the replicas are placed

- CA: Single cluster, all nodes are always in contact. When a partition between nodes should be created the data is out of sync until partition is resolved.
- CP: Some data may not be accessible, but the rest is still consistent/accurate
- AP: System is still available under partitioning, but some of the data returned may be inaccurate. Will resync data once the partition is resolved



-
- **PACELC Theorem:**
  "if there is a **partition** (P), how does the system trade off **availability** and **consistency** (A and C); **else** (E), when the system is running normally in the absence of partitions, how does the system trade off **latency** (L) and **consistency** (C)?"
  - A high availability requirement implies that the system must replicate data. As soon as a distributed system replicates data, a tradeoff between consistency and latency arises.

Hadoop

- 2 Parts:
  - **Storage**: Hadoop Distributed File System (HDFS)
  - **Processing**: MapReduce (manipulates data where it is stored, data locality principle)
- Name node: stores meta data (where each block is stored)
- Data node: actual data blocks
- **Replication** (3) -> so if disk failure on a data node, there are backups on other data nodes
- What if disk failure on name node -> have a **standby** name node
- **MapReduce**
  - Why? – Processing documents top-bottom -> slow
  - MR -> parallel processing
  - Mappers -> give intermediate records <key, value>
  - Reducers get assigned a key -> ask for the stack of that key at each Mapper: **shuffle**
  - Reducer **alphabetically** go through their stacks: sort & process them
- MapReduce jobs submitted to **Job Tracker**
- **Job Tracker** splits work to mappers & reducers
- **Task Tracker**: runs on each data node, executes the actual mapping & reducing
  - On the same machine as data node so saves network traffic

- **Mappers** perform filtering & sorting and pass the intermediate data to reducers
- **Reducers** process this (summary operation) & write final output to hdfs

Week 2 – Features & Data Processing

Features

Properties:

- Distinctness: $= \neq$
  - E.g., Cat ≠ Dog
- Order: $< > \leq \geq$
  - E.g., A+ > B-
- Meaningful differences: $+ \ -$
  - E.g., 08 Oct 2018 is three days after 05 Oct 2018
- Meaningful ratios: $\times \ \div$
  - E.g., Tom (18 years) is two times older than John (9 years)

Type of Features:

- Nominal
  - Property: Distinctness
  - Examples: gender, eye colour, postal codes
- Ordinal
  - Properties: Distinctness and ordered
  - Examples: school level (primary/secondary), grades
- Interval
  - Properties: Distinctness, ordered and meaningful differences
  - Examples: calendar dates, temperatures (Celsius or Fahrenheit)
- Ratio
  - Properties: Distinctness, ordered and meaningful differences/ratios
  - Examples: length, time, counts

- All 4 of them can be represented by discrete or continuous values

Categorical Qualitative: Nominal & Ordinal
Numeric Quantitative: Interval & Ratio

Feature	Binary, Discrete, or Continuous?	Nominal, Ordinal, Interval, or Ratio?
Postal code	Discrete	Nominal
Gender	Binary	Nominal
Height / Weight	Continuous	Ratio
Student ID	Discrete	Nominal, Ordinal (if ID assigned by sequence)
Grading system	Binary (P/F), Discrete (A+,..,F), Continuous (Scores)	Ordinal, Ratio (Scores)
Date	Discrete (MM/YY), Continuous (time)	Interval

Data

Dataset Characteristics:

- Dimensionality (number of features)
  - Challenges of high-dimensional data, "Curse of dimensionality"
- Sparsity
  - E.g., In bag-of-words, most words will be zero (not used)
  - Advantage for computing time and space
- Resolution
  - Patterns depend on the scale
  - E.g., travel patterns on scale of hours, days, weeks

Ashlyn Goh

Possible Issues with Dataset

- Low quality dataset/features lead to poor models
  - E.g., a classifier build with poor data/features may incorrectly diagnose a patient as being sick when he/she is not
- Possible issues with dataset/features
  - Noise
  - Outliers
  - Missing values
  - Duplicate data
  - Wrong/Inconsistent data