Computer Science Department Picnic

Welcome to the 2019-2020 Academic year!

Meet your faculty, department staff, and fellow students in a social setting. Food and drink will be provided.

When: Saturday, August 31st
Time: 1pm-4pm
Where: City Park Shelter #7

FAQs

- PA0 has been posted
  - Sep 5, 5:00PM via Canvas
  - Individual submission (no team submission)
  - Grading: Interview with GTA(s)

- TP0
  - Sep 4, 5:00PM via Canvas
  - Accommodation request, honors student
  - Contact me by Sept 6, 2019
  - No laptop in the class

- Readings
  - Reading research papers
  - Keshav’s “How to read a paper”
  - “How to Read and Understand a Scientific Paper: A Step-by-Step Guide for Non-Scientists”

Topics

- Introduction to Big Data Analytics
  - Data Collection, Sampling, and Preprocessing
  - Introduction to MapReduce

This Material is Built Based on,

The most time-consuming step is the data selection and preprocessing step. This is usually around 80% of the total time needed to build an analytical model.


### Types of Analytics
- Analytics is a term that is often used interchangeably with:
  - Data science
  - Knowledge discovery
- **Descriptive analytics**
  - No target variable
  - E.g. Clustering, association rules
- **Predictive analytics**
  - A target variable is typically available
  - E.g. linear/logistic regression, decision trees, neural networks, support vector machines

### Types of Data Sources
- **Transactions**
  - Structured, low-level, detailed information
  - Customer transactions
  - Purchase, cash transfers, credit card payment
  - Stored in massive online transaction processing (OLTP) relational database
  - Can be summarized over longer time horizons (e.g. averages, relative trends, Max/Min values)

- **Unstructured data embedded in text documents**
  - Emails, web pages, claim forms
  - Requires extensive preprocessing

- **Qualitative, expert-based data**
  - Requires subject matter experts’ (SME) analysis
  - Scientific data

### Sampling
- **Taking a subset of data for analytics**
  - Generating hypothesis
  - Model selection
  - Feature selection
  - Speculative process
  - Building analytics model
- **Stratified sampling**
  - Taking samples according to predefined strata
  - E.g. Fraud detection with very skewed (99 percent non-fraud customers, 1 percent fraud customers)
  - Sample should contain the same percentage of fraud customers as in the original data

### Types of Data Elements
- **Continuous**
  - Data elements that are defined on an interval that can be limited or unlimited
  - E.g. income, sales, temperature

- **Categorical Nominal**
  - Data elements that can only take on a limited set of values with no meaningful ordering between them
  - E.g. marital status, profession, purpose of loan

- **Ordinal**
  - Data elements that can only take on a limited set of values with a meaningful ordering between them
  - E.g. credit rating, age coded as young, middle age and old

- **Binary**
  - Data elements that can only take on two values
  - E.g. smoking habits: allowed to smoke

### Missing Values
- **Missing values** can occur because of various reasons
  - The information can be non-applicable
  - The information can be undisclosed
  - The information can be unavailable
Dealing with Outliers

- **Replace (impute)**
  - Replaces the missing value with a computed/selected value
  - Imputation algorithm examples
    - Hot deck: replaces with a randomly selected similar records
    - Cold deck: selects replacement from another dataset
  - Mean substitution: replaces with the mean of that variable for all other cases
- **Delete**
  - Deletes observations with lots of missing values
  - This assumes that information is missing at random and has no meaningful interpretation and/or relationship to the target
- **Keep**
  - Missing values can be meaningful
    - e.g. a customer did not disclose the income for current condition

### Identifying Outliers using Box Plots

- A box plot represents three key quartiles of the data
  - Q1: 25% of the observations have a lower value
  - Q2: 50% of the observations have a lower value
  - Q3: 75% of the observations have a lower value
  - The minimum and maximum values are added
- **Too far away** is now quantified as more than \( 1.5 \times \text{Interquartile Range} \) \((IQR = Q_3 - Q_1)\)

### Identifying Outliers using Z-Score

- Measuring how many standard deviations an observation is away from the mean
  - \( z = \frac{x - \mu}{\sigma} \)
  - where \( \mu \) represents the average of the variable and \( \sigma \) its standard deviation
- A practical rule of thumb then defines outliers when the absolute value of the z-score \(| z |\) is bigger than 3

### Missing Values --continued

- **Replace (impute)**
  - Replaces the missing value with a computed/selected value
  - Imputation algorithm examples
  - Hot deck: replaces with a randomly selected similar records
  - Cold deck: selects replacement from another dataset
  - Mean substitution: replaces with the mean of that variable for all other cases
- **Delete**
  - Deletes observations with lots of missing values
  - This assumes that information is missing at random and has no meaningful interpretation and/or relationship to the target
- **Keep**
  - Missing values can be meaningful
    - e.g. a customer did not disclose the income for current condition

### Outliers of Dataset

- **Outliers** are extreme observations that are very dissimilar to the rest of the population
  - **Valid observation**
    - Said of observations in a valid region
  - **Invalid observation**
    - Said of observations that are extreme
  - **Multivariate outliers**
    - Observations that are outlying in multiple dimensions
      - e.g. Temperature in Fort Collins is 100 degrees but on a midnight in December

### Standardizing Data

- **Scaling variables to a similar range**
  - e.g. two variables: education and income
  - Elementary school (1), middle school (2), high school (3), college (4), graduate school (5)
  - Income: \( 0 \leq 5M \)
  - When building logistic regression models, the coefficient for education might become very small.
- **Min/Max standardization**
  - \( x_{new} = \frac{x - \min}{\max - \min} (x - \min) + \min \)
  - Where \( \min \) and \( \max \) are the newly imposed maximum and minimum (e.g. 1 and 0)
Standardizing Data. -- continued

- Z-Score based
  - Calculate the z-scores
- Decimal scaling
  - $X_{new} = \frac{X_{old}}{\sigma}$
  - Dividing by a power of 10
- Standardization is useful for regression-based approaches
- It is not needed for decision trees
What is MapReduce?

MapReduce

- MapReduce is inspired by the concepts of map and reduce in Lisp.

- “Modern” MapReduce
  - Developed within Google as a mechanism for processing large amounts of raw data.
    - Crawled documents or web request logs
    - Distributes these data across thousands of machines
    - Same computations are performed on each CPU with different dataset

Sorting 1TB of numbers with a single machine

- What will be the challenges?

Sorting 1TB of numbers with multiple machines

- What will be the requirements to perform this sorting?

MapReduce

- MapReduce provides an abstraction that allows engineers to perform simple computations while hiding the details of parallelization, data distribution, load balancing and fault tolerance

Mapper

- Mapper maps input key/value pairs to a set of intermediate key/value pairs
  - Maps are the individual tasks that transform input records into intermediate records
  - The transformed intermediate records do not need to be of the same type as the input records
  - A given input pair may map to zero or many output pairs
  - The Hadoop MapReduce framework spawns one map task for each InputSplit generated by the InputFormat for the job
Reducer

- Reducer reduces a set of intermediate values which share a key to a smaller set of values
- Reducer has 3 primary phases
  - Shuffle, sort and reduce
- Shuffle
  - Input to the reducer is the sorted output of the mappers
  - The framework fetches the relevant partition of the output of all the mappers via HTTP
- Sort
  - The framework groups input to the reducer by keys

Example 1: WordCount [1/5]

- For text files stored under `usr/joe/wordcount/input`, count the number of occurrences of each word
- How do files and directory look?

```bash
$ bin/hadoop dfs -ls /usr/joe/wordcount/input
$ bin/hadoop dfs -ls /usr/joe/wordcount/input/file01
$ bin/hadoop dfs -ls /usr/joe/wordcount/input/file02
$ bin/hadoop dfs -cat /usr/joe/wordcount/input/file01
Hello World, Bye World!
$ bin/hadoop dfs -cat /usr/joe/wordcount/input/file02
Hello Hadoop, Goodbye to hadoop.
```

Example 1: WordCount [2/5]

- Run the MapReduce application

```bash
$ bin/hadoop jar /usr/joe/wordcount.jar org.myorg.WordCount /usr/joe/wordcount/input /usr/joe/wordcount/output
$ bin/hadoop dfs -cat /usr/joe/wordcount/output/part-00000
Bye 1
Goodbye 1
Hadoop, 1
Hello 2
World, 1
hadoop, 1
to 1
```

Example 1: WordCount [3/5]

Mappers
- Read line
- Tokenize the string
- Pass the `key, value` pair to the Reducer

Reducer
- Collect `key, value` pair
- Aggregate total number of occurrences

```java
public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
}
```

Example 1: WordCount [4/5]

```java
public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
    private IntWritable result = new IntWritable();
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);
        context.write(key, result);
    }
}
```
Example 1: WordCount

```java
public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

Questions?