Scalable Data Analysis (CIS 602-02)

Databases

Dr. David Koop
Classification

[MNIST Digits, Y. LeCun]
Regression

![Graph showing a linear relationship between weight (kg) and height (cm).]
Clustering

k-Means Clusters

Iris Species

Cluster 1
Cluster 2
Cluster 3

Iris setosa
Iris versicolor
Iris virginica
Machine Learning Problems

- **Supervised**: have input data and known labels, build solution that will predict labels from new inputs
  - Classification: predict class (yes/no, apple/banana/pear)
  - Regression: predict parameters (e.g. given weight, predict height)
- **Unsupervised**: have input data but no given labels, want to cluster the data
- **Semi-supervised**: e.g. cluster handwritten digits, ask a human to label a few from each cluster (perhaps those at boundaries), continue learning
Overview: Methods of Clustering

• Hierarchical:
  - Agglomerative (bottom up):
    • Initially, each point is a cluster
    • Repeatedly combine the two “nearest” clusters into one
  - Divisive (top down):
    • Start with one cluster and recursively split it

• Point assignment:
  - Maintain a set of clusters
  - Points belong to “nearest” cluster

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters after round 1

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters after round 1

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters after round 1

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters after round 1

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters after round 1

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters after round 2

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters after round 2

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters after round 2

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters at the end

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters at the end

[Leskovec et al., Mining of Massive Datasets]
Example: Assigning Clusters

Clusters at the end

[Leskovec et al., Mining of Massive Datasets]
Example: Picking $k$

Too few; many long distances to centroid.

[Leskovec et al., Mining of Massive Datasets]
Example: Picking $k$

Just right; distances rather short.

[Leskovec et al., Mining of Massive Datasets]
Example: Picking $k$

Too many; little improvement in average distance.

[Leskovec et al., Mining of Massive Datasets]
BFR Algorithm

- BFR [Bradley-Fayyad-Reina] is a variant of k-means designed to handle very large (disk-resident) data sets
- Assumes that clusters are normally distributed around a centroid in a Euclidean space
  - Standard deviations in different dimensions may vary
    - Clusters are axis-aligned ellipses
- Efficient way to summarize clusters (want memory required O(clusters) and not O(data))

[Leskovec et al., Mining of Massive Datasets]
BFR Algorithm

- Points are read from disk one main-memory-full at a time
- Most points from previous memory loads are summarized by simple statistics
- To begin, from the initial load we select the initial k centroids by some sensible approach:
  - Take k random points
  - Take a small random sample and cluster optimally
  - Take a sample; pick a random point, and then k–1 more points, each as far from the previously selected points as possible
Three Classes of Points

- 3 sets of points which we keep track of:
  - **Discard set (DS):**
    - Points close enough to a centroid to be summarized
  - **Compression set (CS):**
    - Groups of points that are close together but not close to any existing centroid
    - These points are summarized, but not assigned to a cluster
  - **Retained set (RS):**
    - Isolated points waiting to be assigned to a compression set
BFR: “Galaxies” Picture

Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

[Leskovec et al., Mining of Massive Datasets]
A cluster. Its points are in the **DS**.

**Discard set (DS):** Close enough to a centroid to be summarized

**Compression set (CS):** Summarized, but not assigned to a cluster

**Retained set (RS):** Isolated points

[Leskovec et al., Mining of Massive Datasets]
A cluster. Its points are in the **DS**.

**Compressed sets.** Their points are in the **CS**.

**Discard set (DS):** Close enough to a centroid to be summarized

**Compression set (CS):** Summarized, but not assigned to a cluster

**Retained set (RS):** Isolated points

[Leskovec et al., Mining of Massive Datasets]
A cluster. Its points are in the **DS**.

Compressed sets. Their points are in the **CS**.

Points in the **RS**

**Discard set (DS):** Close enough to a centroid to be summarized

**Compression set (CS):** Summarized, but not assigned to a cluster

**Retained set (RS):** Isolated points

[Leskovec et al., Mining of Massive Datasets]
Assignment 2

- [http://www.cis.umassd.edu/~dkoop/cis602/assignment2.html](http://www.cis.umassd.edu/~dkoop/cis602/assignment2.html)
- Bikes and Weather
- Bike sharing data and weather for Washington DC
- Hypothesis might be that people are more likely to use the program when the weather is "nicer"
- Integrate data, visualize it, do regression
- Use pandas, matplotlib, and scikit-learn libraries
- Can generate a solution with relatively little code (~50 lines) but will require experimentation to get there
- Due Tuesday, October 20
Project Proposal

• Type of project: (a) Analyze real data, (b) Improve a technique

• Analyze real data
  - What are the datasets?
  - How much data is there (size/scope/time span)?
  - What questions would you be interested in?
  - What has already been done? Related Work
  - What techniques do you plan to use to answer these questions?

• Improve technique
  - What is the problem?
  - What ideas do you have to solve the problem?
  - What has already been done? Related Work
  - How do you plan to evaluate your improved technique? (e.g. data)
Project Proposal

• You may work on the same dataset but must have independent questions, investigations, code, and reports
• Datasets must be large enough to provide scalability challenges
• Potential datasets:
  - https://github.com/caesar0301/awesome-public-datasets
  - https://github.com/fivethirtyeight/data
  - http://datahub.io/dataset
  - http://www.37billionmilechallenge.org
  - https://data.cityofboston.gov
Introduction

Fig. 1.1 Main components of a DBMS.

A well-understood point of reference for new extensions and revolutions in database systems that may arise in the future. As a result, we focus on relational database systems throughout this paper.

At heart, a typical RDBMS has five main components, as illustrated in Figure 1.1. As an introduction to each of these components and the way they fit together, we step through the life of a query in a database system. This also serves as an overview of the remaining sections of the paper.

Consider a simple but typical database interaction at an airport, in which a gate agent clicks on a form to request the passenger list for a flight. This button click results in a single-query transaction that works roughly as follows:

1. The personal computer at the airport gate (the “client”) calls an API that in turn communicates over a network to establish a connection with the Client Communications Manager of a DBMS (top of Figure 1.1). In some cases, this connection...

Relational Database Architecture

[Hellerstein et al., Architecture of a Database System]
156 Process Models

Fig. 2.3 Process Pool: each DBMS Worker is allocated to one of a pool of OS processes as work requests arrive from the Client and the process is returned to the pool once the request is processed. If all processes are already servicing other requests, the new request must wait for a process to become available.

Process pool has all of the advantages of process per DBMS worker but, since a much smaller number of processes are required, is considerably more memory efficient. Process pool is often implemented with a dynamically resizable process pool where the pool grows potentially to some maximum number when a large number of concurrent requests arrive. When the request load is lighter, the process pool can be reduced to fewer waiting processes. As with thread per DBMS worker, the process pool model is also supported by several current generation DBMS in use today.

2.1.4 Shared Data and Process Boundaries

All models described above aim to execute concurrent client requests as independently as possible. Yet, full DBMS worker independence and isolation is not possible, since they are operating on the same shared process pool architecture.

Connections Multiplexed Over Process Pool

[Hellerstein et al., Architecture of a Database System]
Parallel DB Architecture: Shared Memory

Figure 3.1 Shared-memory architecture.

Multi-core processors support multiple processing cores on a single chip and share some infrastructure such as caches and the memory bus. This makes them quite similar to a shared-memory architecture in terms of their programming model. Today, nearly all serious database deployments involve multiple processors, with each processor having more than one CPU. DBMS architectures need to be able to fully exploit this potential parallelism. Fortunately, all three of the DBMS architectures described in Section 2 run well on modern shared-memory hardware architectures.

The process model for shared-memory machines follows quite naturally from the uniprocessor approach. In fact, most database systems evolved from their initial uniprocessor implementations to shared-memory implementations. On shared-memory machines, the OS typically supports the transparent assignment of workers (processes or

1 The dominant cost for DBMS customers is typically paying qualified people to administer high-end systems. This includes Database Administrators (DBAs) who configure and maintain the DBMS, and System Administrators who configure and maintain the hardware and operating systems.
3.2 Shared-Nothing

A shared-nothing parallel system (Figure 3.2) is made up of a cluster of independent machines that communicate over a high-speed network interconnect or, increasingly frequently, over commodity networking components. There is no way for a given system to directly access the memory or disk of another system.

Shared-nothing systems provide no hardware sharing abstractions, leaving coordination of the various machines entirely in the hands of the DBMS. The most common technique employed by DBMSs to support these clusters is to run their standard process model on each machine, or node, in the cluster. Each node is capable of accepting client SQL requests in parallel. All three models run well on these systems and support the execution of multiple, independent SQL requests in parallel. The main challenge is to modify the query execution layers to take advantage of the ability to parallelize a single query across multiple CPUs; we defer this to Section 5.

[Figure 3.2: Shared-nothing architecture.]

[Hellerstein et al., Architecture of a Database System]
Sharding

![Sharding Diagram](MongoDB)

D. Koop, CIS 602-02, Fall 2015
Parallel DB Architecture: Shared Disk

A shared-disk parallel system (Figure 3.3) is one in which all processors can access the disks with about the same performance, but are unable to access each other’s RAM. This architecture is quite common with two prominent examples being Oracle RAC and DB2 for zSeries SYSEX. Shared-disk has become more common in recent years with the increasing popularity of Storage Area Networks (SAN). A SAN allows one or more logical disks to be mounted by one or more host systems making it easy to create shared disk configurations.

One potential advantage of shared-disk over shared-nothing systems is their lower cost of administration. DBAs of shared-disk systems do not have to consider partitioning tables across machines in order to achieve parallelism. But very large databases still typically do require partitioning so, at this scale, the difference becomes less pronounced.

Another compelling feature of the shared-disk architecture is that the failure of a single DBMS processing node does not affect the other nodes’ ability to access the entire database. This is in contrast to both shared-memory systems that fail as a unit, and shared-nothing systems that lose access to at least some data upon a node failure (unless some alternative data redundancy scheme is used). However, even with these advantages, shared-disk systems are still vulnerable to some single

Fig. 3.3 Shared-disk architecture.

[Hellerstein et al., Architecture of a Database System]
Snowflake Schema

[SqIPac at English Wikipedia]
Online Transactional Processing (OLTP)

- Used in business applications
- Data entry and retrieval transactions
- e.g. retail sales
- OLTP is focused on the day-to-day operations while Online Analytical Processing (OLAP) is focused on analyzing that data for trends, etc.
Relational Databases: One size fits all?

- Lots of work goes into relational database development:
  - B-trees
  - Cost-based query optimizers
  - ACID (Atomicity, Consistency, Isolation, Durability) makes sure that transactions are processed reliably
- Vendors have stuck with this model since the 1980s
- Having different systems leads to business problems:
  - cost problem
  - compatibility problem
  - sales problem
  - marketing problem

[Stonebraker and Çetinetmel, 2005]
The End of an Architectural Era
(It's Time for a Complete Rewrite)

M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, P. Helland
Next Week

• More on databases: Google's database for AdWords
• Project proposal
• Midterm details
• Assignment 2 Due