CIS4301 Notes: Data Mining

Ryan Roden-Corrent

April 14, 2014

1 MapReduce

MapReduce Class Slides Popular in **functional** languages (Ruby, Lisp, Python).

1.1 Functional Languages

Functional languages treat functions as first class objects. Functions take other functions as arguments and return them as results. Operators do not mutate collections. For example:

```
1  // isprof is a function that returns true if the argument is a professor
2  // list is a list of people, some of which may be professors
3  count(isprof, list);
```

1.2 Common functions

map apply a function to each element to create a new collection

```
map (+3 [1,2,3,4,5] => [4,5,6,7,8]
```

reduce apply a function between each element and an accumulator to reduce a list to a single value

any return true if a function returns true for ≥ 1 value

all return true if a function returns true for all values

filter keep only values where function returns true

fold like reduce, but take an initial parameter for the accumulator

1.3 Unix Philosophy

Unix Philosophy: A tool should do one thing well. Complex processes should be achieved by chaining several commands together.

1.4 MapReduce

Proposed by Google engineers. It includes:

- MapReduce programming model
- Google file system
- Job scheduling system

1.5 MapReduce programming model

Generalize map and reduce funtions.

Listing 1: mapreduce example

```
1  map(String key, String value):
2    // key: document name
3    // value: document contents
4    for each word w in value:
5      EmitIntermediate(w, "1");
6  reduce(String key, Iterator values):
7    int result = 0;
8    for each v in values:
9      result += ParseInt(v);
10     Emit(AsString(result));
```

This code runs on a large scale system. The map job is split up between various machines. Once these results are obtained, they are passed on to another set of machines for reduction. The process of assigning map results to reduce machines is called the **shuffle** step.