Big data and its applications

Mayank Kejriwal

Just last week in *Science*…

A healthy dose of skepticism can help one spot misleading data.

PHOTO: ISTOCK.COM/WUTWHANFOTO

PODCAST

Calling Bullshit: The Art of Skepticism in a Data-Driven World

Carl T. Bergstrom and Jevin D. West
Random House, 2020. 336 pp.

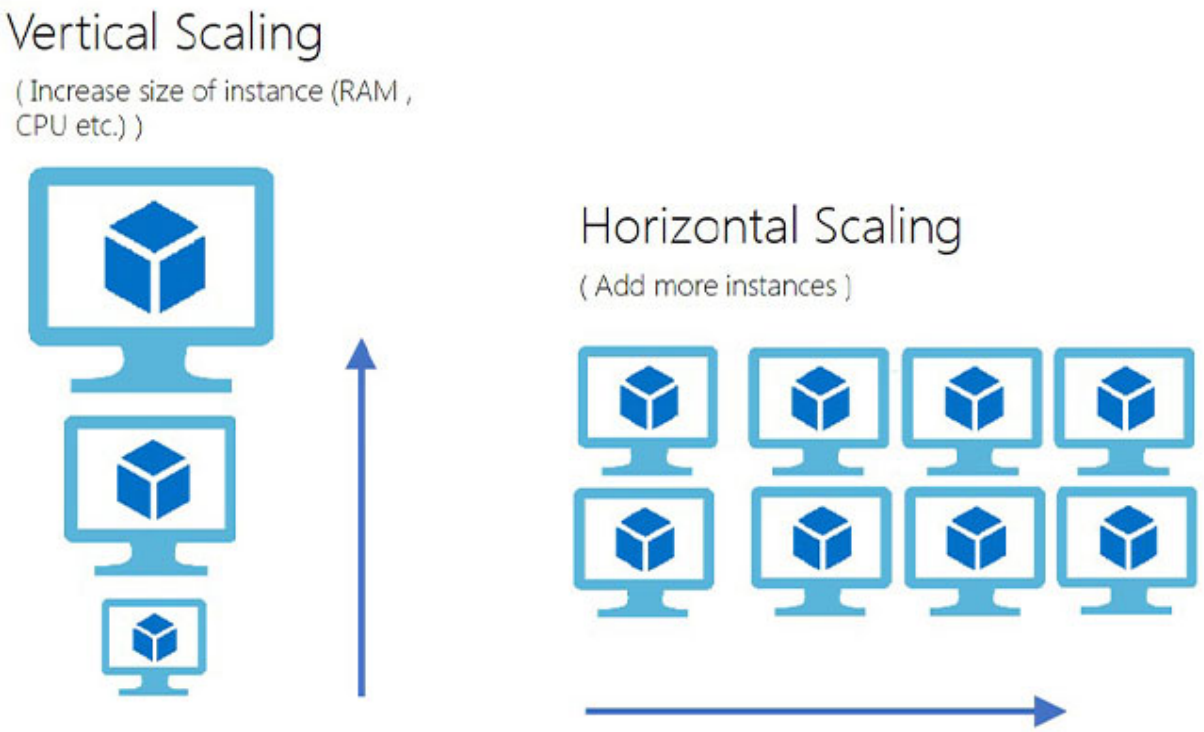
There are three kinds of lies, the oft-quoted saying goes: lies, damned lies, and statistics. In a world drowning in data, statistical methods and other tools of scientific inquiry are increasingly being used to advance erroneous claims. This week on the *Science* podcast, evolutionary biologist Carl Bergstrom explains how to identify data-driven misinformation and disinformation.

10.1126/science.abd9788

Originally 4 Vs (variety, velocity, volume and veracity)

- Today many more…
- Think of a **text** application that ‘activates’ or requires as many of these 8Vs as possible. What about non-text? Which Vs are most important for text, in your opinion?

Horizontal vs. vertical scaling



Must it be one or the other? Can you think of hybrids?

MapReduce and Hadoop

- MapReduce is a ‘framework’ for embarrassingly parallel programming
- Popularized in the article by Google researchers Dean and Ghemawat (https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf)
  - Highly recommended reading
- Implemented as Apache Hadoop, available on all cloud platforms!

Best illustrated through an example

- (But to truly understand, you must go through the first three pages of the linked paper)
- **Example problem:** Suppose we wanted to count the number of occurrences of each word in a large collection of documents

MapReduce ‘hello world’: word counting in large corpus

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

Mapreduce Case study: k-means

Very applicable to large-scale, unsupervised text analytics

Introduction to K-Means

- Those of you who have done machine learning are probably familiar with it (but doesn't hurt to visit)
- Even today, one of the few (if not the only) clustering algorithms appropriate for truly large datasets!
- Good to use it as a lens for understanding the various kinds of scaling and Big Data platforms

The k-means Clustering Algorithm
Input : Data points D, Number of clusters k
Step 1: Initialize k centroids randomly
Step 2: Associate each data point in D with the nearest centroid. This will divide the data points into k clusters.
Step 3: Recalculate the position of centroids.
Repeat steps 2 and 3 until there are no more changes in the membership of the data points
Output : Data points with cluster memberships

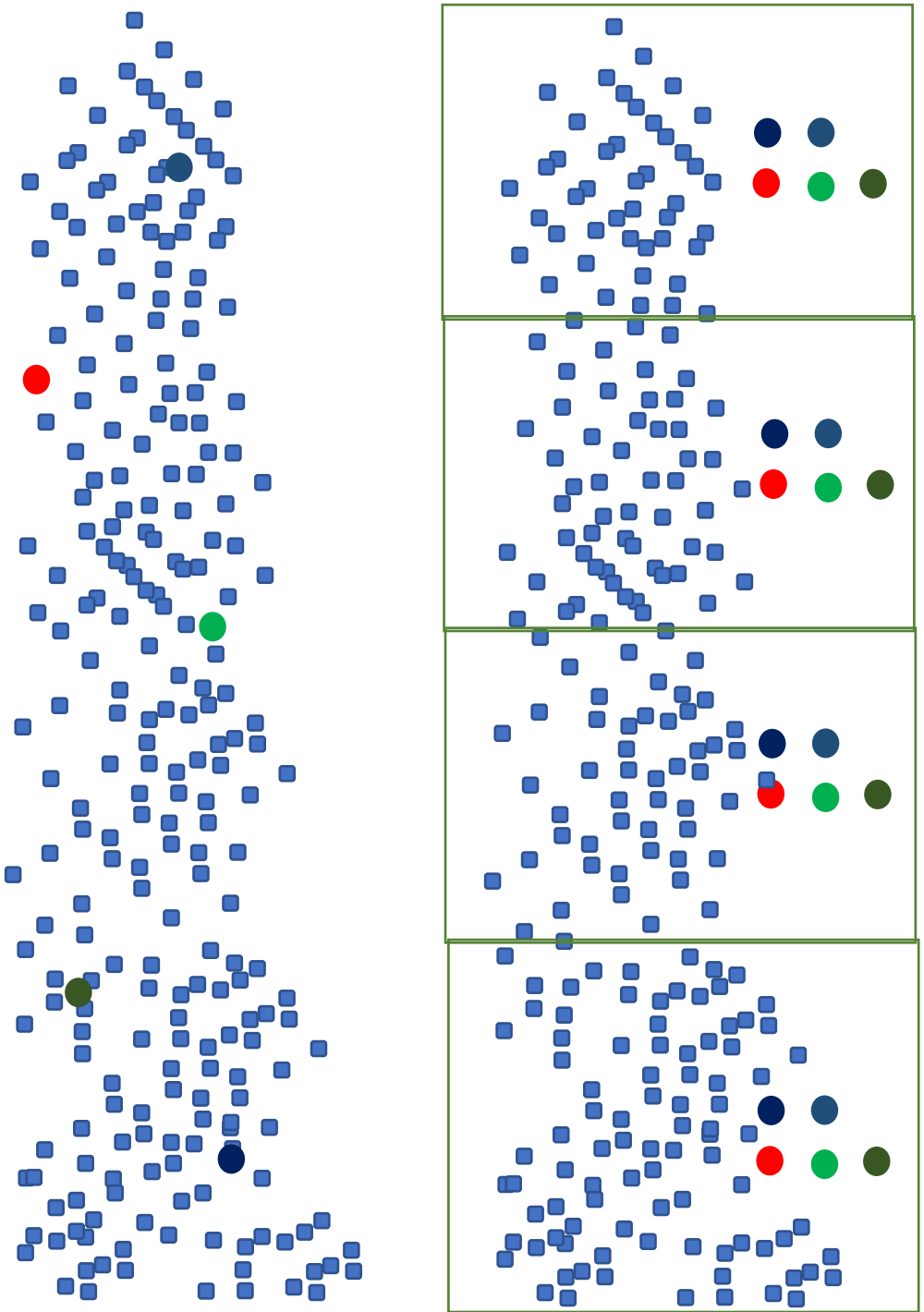
https://www.youtube.com/watch?v=5I3Ei69I40s

K-MEANS IN MAPREDUCE

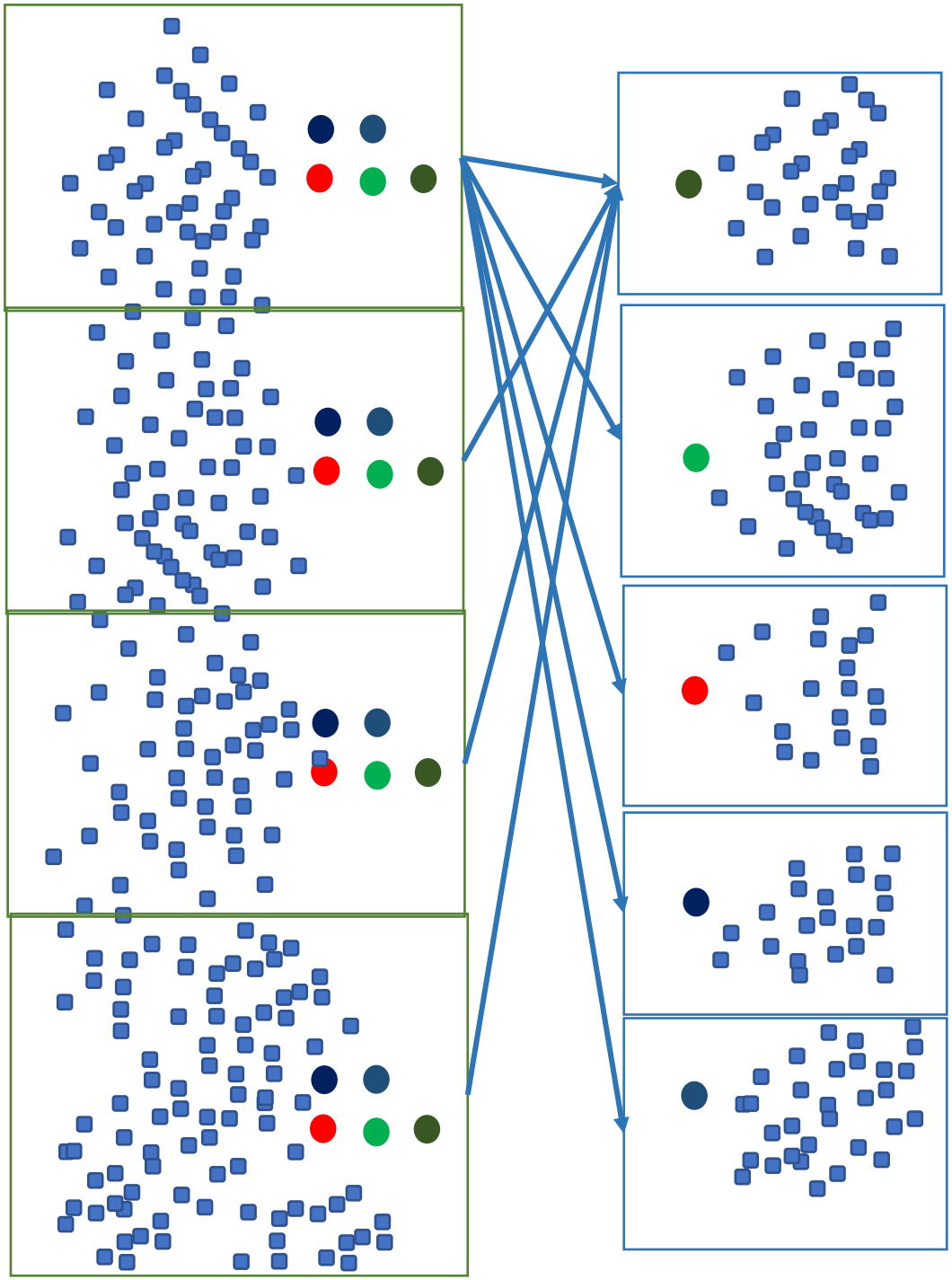
k-means::Map
Input: Data points D, number of clusters k and centroids
1: for each data point d Є D do
2: Assign d to the closest centroid
Output: centroids with associated data points

---

k-means::Reduce
Input: Centroids with associated data points
1: Compute the new centroids by calculating the average of data points in cluster
2: Write the global centroids to the disk
Output: New centroids



- Illustration (k=5)
- 4 mappers
- Each mapper receives roughly N/4 data points+all centroids



- Illustration (k=5)
- 4 mappers
- k reducers (centroid serves as key)
- Reducers re-compute centroids and write to file
- Requires multiple iterations of mapreduce to converge!
- I haven't shown all arrows in figure, but potentially all mappers send data to all reducers (can you think of an exception?)

What can go wrong?

Think about specific examples….