Big Data Part Three: Examples of Big Data Programming

Chris Jermaine

Rice University

First Big Data Activity

- We'll actually use a Spark cluster
  - Very easy to set up a cluster using Amazon's EC2 service
  - Other vendors (Microsoft, Google, etc.) have similar services
- Activity is running word count on Spark/EC2
  - Word count is everyone's favorite first-big-data-computation!
- See `cmj4.web.rice.edu/DSDay2/GettingStarted.html`

Next, a Simple kNN Classifier on Spark

- Say we have a very large database of text documents (aka a "corpus")
- Each doc is labeled (say, "spam" or "not spam")
- Want to learn how to label documents...

The Classic Workflow

- First, build a dictionary for the corpus
- We talked about this before...
- A dictionary of size d is a map from each word in the corpus to an integer from 0 to $d-1$
- Then, process each doc to obtain a "bag of words"
    - Start with an array/vector x of length d, initialized to all zeros
    - Then, for each word in the doc:
    - (1) Look it up in the dictionary to get its corresponding int i
    - (2) Increment $x[i]$

Example

start dictionary with zero, so numbers can be 0..9

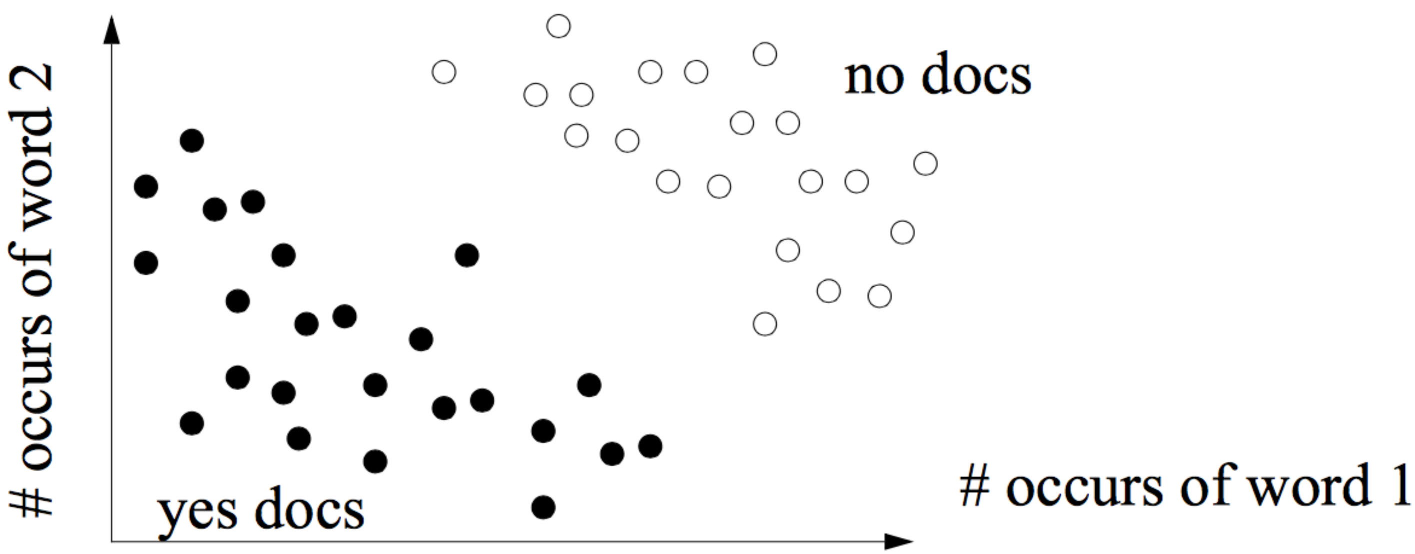
- Doc is "This was followed by radiotherapy."
  - Dictionary is: $\{(patient, 1), (status, 2), (followed, 3), (radiotherapy, 4), (negative, 5), (was, 6), (this, 7), (treated, 8), (by, 9)\ (with, 10)\}$
  - $x = \langle 0, 0, 1, 1, 0, 1, 1, 0, 1, 0 \rangle$
- Now want to figure out how to classify (label) text documents...
  - For example: "+1: this patient had breast cancer"
  - "-1: this patient didn't have breast cancer"
- How?

Using Existing Labeled Data

- Assume we have a set of labeled data
  - For example, check to see if the patient was billed for BC in next 6 months
  - This gives a set of $(x, \boldsymbol{label})$ pairs
- Feed these as training data into your classifier-of-choice
- Then, when have a new record to classify
  - Convert it into a bag-of-words
  - And feed it into the classifier for labeling

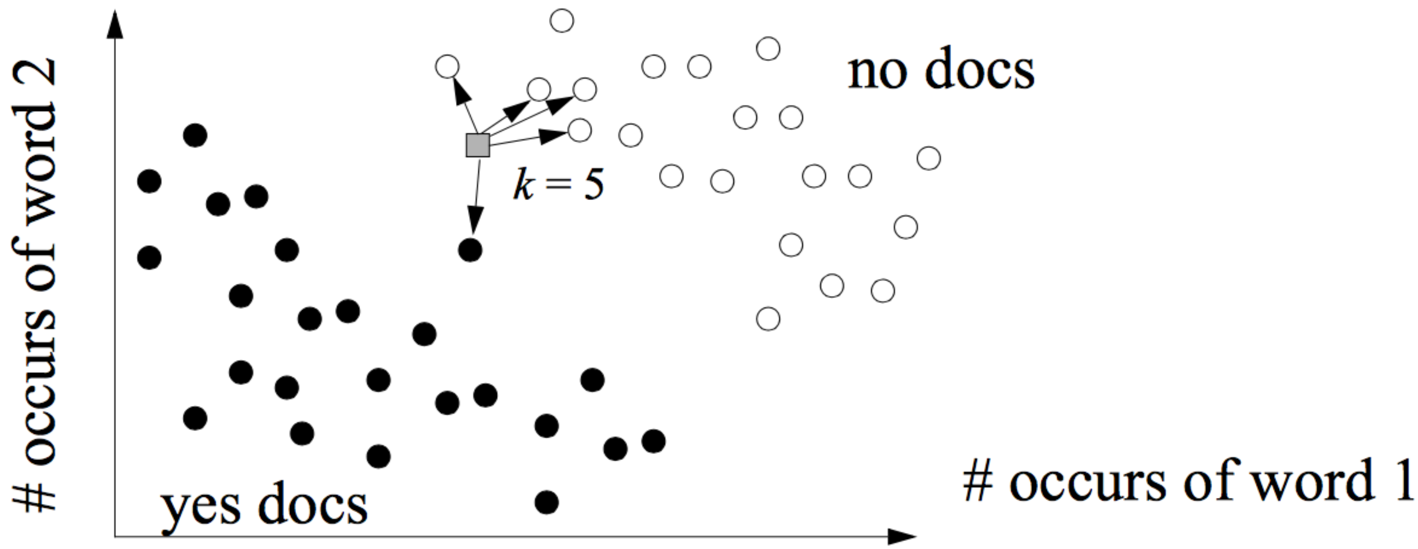
A Common Classifier is kNN

- This is the first one that you'll be implementing on Spark
- Idea: place docs in multi-dim space



A Common Classifier is kNN

- To classify a new doc...
  - You place it in the space, and find its k nearest neighbors (hence "kNN")
  - And you give it the most common label of its k nearest neighbors



How to Define Distance?

- Most common is Euclidean distance
  - Note: small values imply closeness

$$ED(x_1, x_2) = \sum_{i=1}^{d} (x_1[i] - x_2[i])^2$$

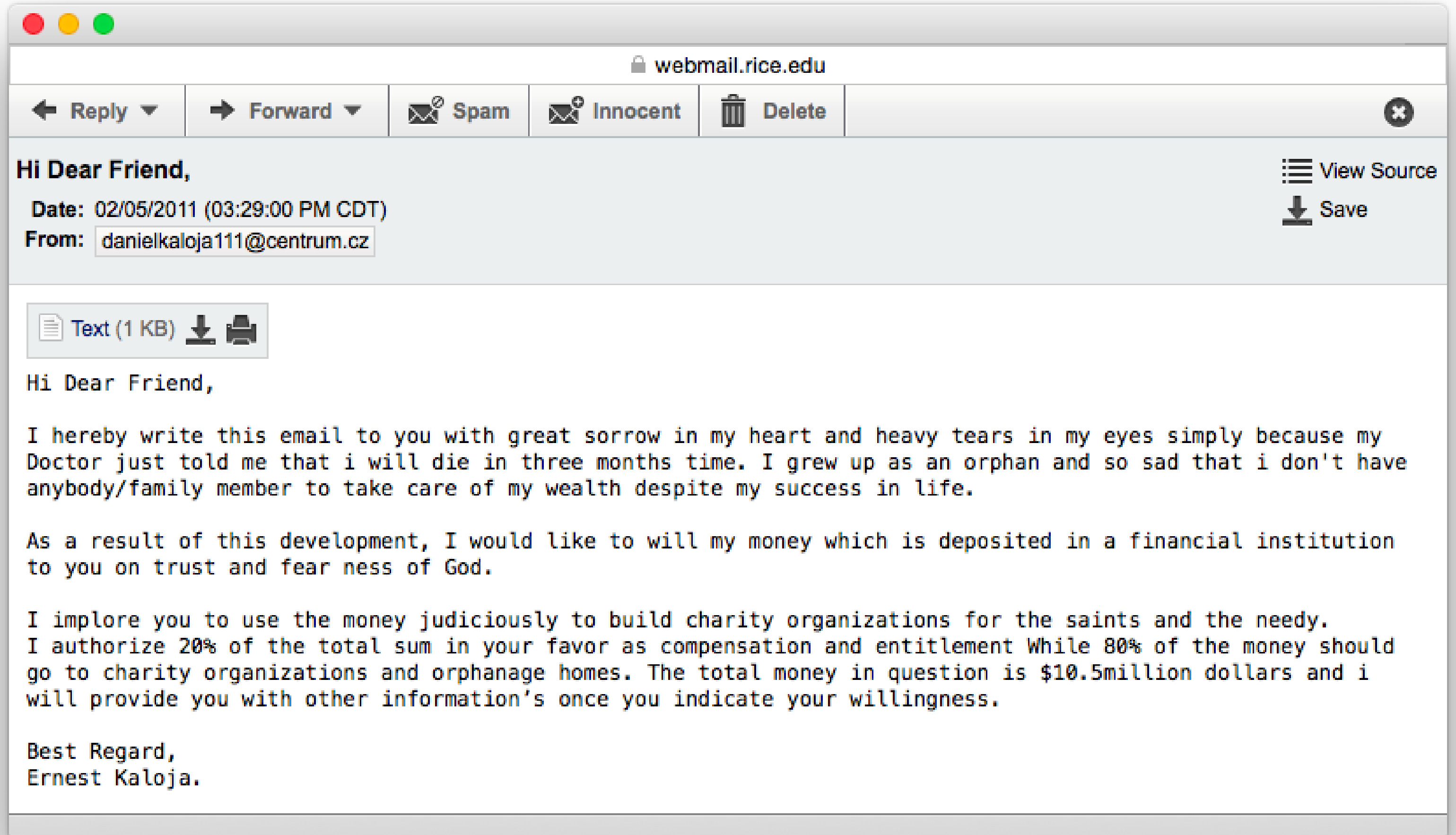
- We will use cosine similarity (note: big values mean "close"):
  - Rewards similar concentration in a few dims
  - Just a dot product

$$Cos(x_1, x_2) = \sum_{i=1}^{d} x_1[i] \times x_2[i]$$

Activity: KNN over bag-of-words vectors

- Basic idea: pre-process a text data set into bag-of-words vectors
- Then, given a query vector, find closest using cosine similarity
  - Activity at `cmj4.web.rice.edu/DSDay2/kNNBasic.html`

Beyond Term Counts



Hi Dear Friend,

Date: 02/05/2011 (03:29:00 PM CDT)
From: danielkaloja111@centrum.cz

```
Hi Dear Friend,

I hereby write this email to you with great sorrow in my heart and heavy tears in my eyes simply because my
Doctor just told me that i will die in three months time. I grew up as an orphan and so sad that i don't have
anybody/family member to take care of my wealth despite my success in life.

As a result of this development, I would like to will my money which is deposited in a financial institution
to you on trust and fear ness of God.

I implore you to use the money judiciously to build charity organizations for the saints and the needy.
I authorize 20% of the total sum in your favor as compensation and entitlement While 80% of the money should
go to charity organizations and orphanage homes. The total money in question is $10.5million dollars and i
will provide you with other information’s once you indicate your willingness.

Best Regard,
Ernest Kaloja.
```

TF-IDF and Inverse Document Frequency

Term Frequency

- Words like "tears", "sorrow" probably rare in my email
- So we want to weight them more heavily
- "Inverse Document Frequency"
  - Defined as:

$$IDF = \log \frac{\text{num of docs}}{\text{num of docs having the word}}$$

Value rarer words

Term Frequency

- Another issue: length of doc (probably) should not matter
- If it does, just have it be another feature
- "Term Frequency" normalizes by doc length
  - Defined as:

$$TF = \frac{\text{num occurs of word in doc}}{\text{num words in doc}}$$

- TF-IDF defined as $TF \times IDF$
  - Most common in this sort of application

Normalize word freq by words in each doc

N-Grams

- Words in this doc might not be suspicious
- Might be how they are put together
  - "great sorrow"
  - "heavy tears"
  - "financial institution"
  - "fear ness"
- Idea: also include all 2-grams, 3-grams, 4-grams, etc. as features

Other Standard Techniques

- Remove stop words since they convey little meaning
  - the, which, at, on...
- Perform stemming of words
  - running, run, runner, runs are all reduced to run
- Use only the top k most frequent words
  - Removes obscure terms, mis-spellings, etc.

Activity: kNN Using TF-IDF

- Same as last time, but normalize and multiply by IDF vector
  - See `cmj4.web.rice.edu/DSDay2/kNNTFIDF.html`

Linear Regression

- kNN often works well, but it's expensive
- Alternative: LR
  - Expensive to build model
  - But very inexpensive to apply

Typically for classification you use Logistic Regression (not LR)

How LR Works

- In LR, compute a vector of weights r
  - Big positive value in dim i means positive feature i implies "yes"
  - Big negative value in dim i means positive feature i implies "no"

- Known as the vector of "regression coefficients"
  - Then to classify TF-IDF vector x, compute:

$$\sum_{i=i}^{d} r[i] \times x[i]$$

  - This is just the dot product of r and w

- If greater than 0, say yes, otherwise no

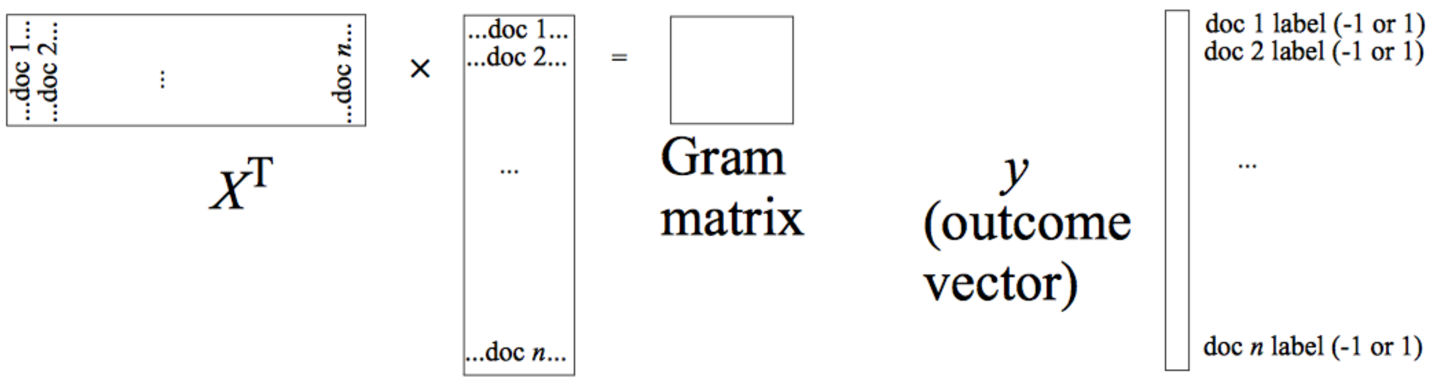
- Drawback compared to kNN: doesn't trivially extend to multiple classes

How to Compute a LR Model

- Let X be the data matrix, y the outcome vector
  - ▷ Each row in X is a doc, column TF-IDF value for a word
  - ▷ Each entry in y is $+1$ for "yes", -1 for "no"
- The Gram matrix is $X^T X$... we did this yesterday! (but no trimming this time)
- r is computed as (Gram matrix)$^{-1} X^T y$

Computing LR

- r is computed as $(\text{Gram matrix})^{-1} X^T y$



How to Compute on Spark?

- Computing the Gram matrix
  - We know matrix multiply fastest, but not feasible here
  - Why? X is distributed in an RDD
  - So use `map ()` that computes outer product
  - And then use `aggregate ()` to sum them, collect locally, invert locally
  - `aggregate ()` just like `aggregateByKey ()`, but no grouping by key

Large # of docs across multiple machines;
compute in a distributed fashion

Simple sum can use reduce instead of more complicated aggregate

How to Compute on Spark? (cont)

- Then, use a map()
  - To multiply (Gram mat)$^{-1}$ by every data record
  - Gets (Gram mat)$^{-1}\boldsymbol{X}^T$ stored as one vec per doc
- Finally, multiply every result rec by corresponding outcome/class label
  - And sum using aggregate()
  - Gets us (Gram mat)$^{-1}\boldsymbol{X}^T\boldsymbol{y}$
- That's it! Well, almost...

Gram Matrix Too Expensive

- Have to sum 20,000 different 20,000 by 20,000 matrices
- Not gonna happen fast enough
- So map each 20,000-dimensional vector down to 1,000 dims
  - ▷ Now we sum 20,000 different 1,000 by 1,000 matrices
  - ▷ Much more reasonable!

Dimensionality Reduction

- How to map?
- Could use something like PCA
  - This is classic
  - But too expensive
- We just use a random mapping...
  - In practice, almost as good as PCA
  - Unless you map to very low dimensions

Dim Reduction via Random Projection

- How does it work?
  - ▷ Fill a matrix M of 1,000 columns and 20,000 rows
  - ▷ With samples from a Normal$(0, 1)$ distribution

- Then before we compute $(\text{Gram mat})^{-1} X^T y$
  - ▷ Just multiply each data vector with M
  - ▷ 20,000-dimensional problem becomes 1,000 dimensional!

Last Activity

- LR over TF-IDF bag-of-words vectors
- Tries to classify religion vs. not
- Check out
  - `cmj4.web.rice.edu/DSDay2/LinReg.html`

Questions?