Machine Learning at Scale
Linear Regression at Scale
Synonym detection
GMM

James G. Shanahan [2]

Assistants: Liang Dai[1, 3]

[1]NativeX, [2]iSchool UC Berkeley, CA, [3]UC Santa Cruz

EMAIL: James_DOT_Shanahan_AT_gmail_DOT_com

Live Session #6

Feb 16, 2016

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW3, HW4, HW5, HW5
  - AWS: (Not necessary for week 6)
  - Synonym detection in Map-Reduce: Case studies + Metrics
  - Async lecture recap plus Q&A [Jimi]
  - MLE, MAP, Bayesian optimal hypothesis/classifier
  - MLE for Gaussian Mixture Models
  - EM for Gaussian Mixture Models
  - EM Algorithm for a mixture of Bernoulli distributions
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

HW 5 Extension

- Homework HW5
- extension to Friday at 8AM for HW5!

Time to complete your Survey

- https://www.surveymonkey.com/r/JXT56QT

Spring 2016 Mid Course Evaluation - DATA SCI W261.4 Shanahan

MID SEMESTER REVIEW FORM

COURSE NO: DATA SCI W261.4

INSTRUCTOR: JAMES SHANAHAN

TERM: SPRING 2016

Please consider all relevant aspects of the course, such as cases, lectures, class discussion, teaching style, and readings, and answer the following questions:

1. What do you find most valuable about this course?

2. With a realistic view as to what can be changed at this point in the semester, what one aspect should the instructor change in this course? How?

Peer Grading

- **HW 3 and 4 (100%)**
  - Solution Notebooks will be sent out on Friday
- **HW5 will be peer graded as will HW6**
  - [redacted]

- **Homework HW 6 Folder Questions + Data**
  - HW is a group oriented homework
  - https://www.dropbox.com/s/ctehxulla3o2sim/HW6-Questions.txt?dl=0

Important Links for Week 6

- **Live session Slides**

- **Instructions for Peer grading of homework HW**
  - https://www.dropbox.com/s/97m31frthj4ac28/HOMEWORK%20GRADING%20INSTRUCTIONS%20for%20MIDS%20MLS?dl=0
- **Homework HW Folder Questions + Data**
  - HW is a group oriented homework
  - https://www.dropbox.com/s/ctehxulla3o2sim/HW6-Questions.txt?dl=0
- **Team assignments**
  - https://docs.google.com/spreadsheets/d/1ncFQl5Tovn-16slD8mYjP_nzMTPSfiGeLLzW8v_sMjg/edit?usp=sharing
- **Please submit your homeworks (one per team) going forward via this form (and not thru the ISVC):**
  - https://docs.google.com/forms/d/1ZOr9RnIe_A06AcZDB6K1mJN4vrLeSmS2PD6Xm3eOiis/viewform?usp=send_form

- **MRJob on the cloud (see Week 4 live session)**

MrJob

Configuration ¶

mrjob has an overflowing cornucopia of configuration options. You'll want to specify some on the command line, some in a config file.

You can put a config file at `/etc/mrjob.conf`, `~/.mrjob.conf`, or `./mrjob.conf` for mrjob to find it without passing it via `--conf-path`.

Config files are interpreted as YAML if you have the `yaml` module installed. Otherwise, they are interpreted as JSON.

See *Config file format and location* for in-depth information. Here is an example file:

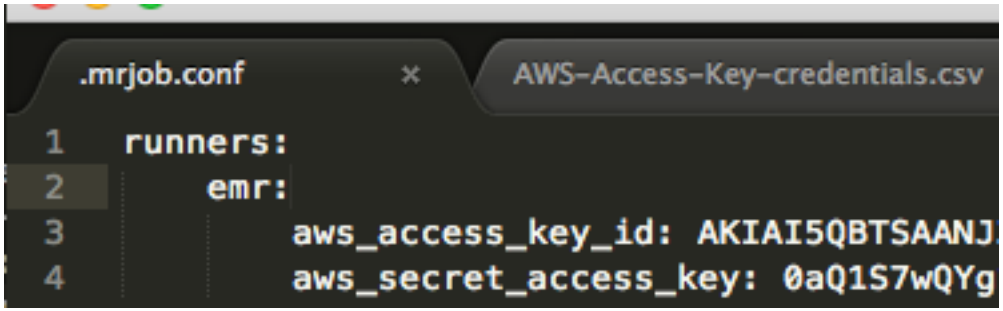
```
runners:
  emr:
    aws-region: us-west-1
    python_archives:
      - a_library_I_use_on_emr.tar.gz
  inline:
    base_tmp_dir: $HOME/.tmp
```

Let's first take a look at mrjob.conf

We can use a pretty basic configuration for all of this, but without multiple instances, we would have to wait a long time for things to finish.

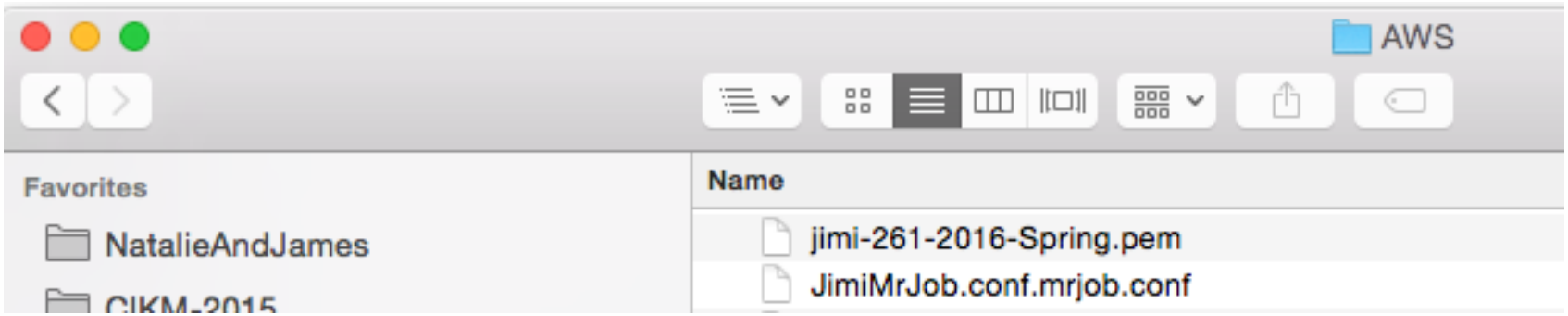
cat ~/.mrjob.conf

```
runners:
  emr:
    aws_access_key_id: XXXX
    aws_secret_access_key: XXXX
    num_ec2_core_instances: 4
    ec2_core_instance_type: m1.medium
    ec2_master_instance_type: m1.medium
    strict_protocols: true
  hadoop:
    strict_protocols: true
  inline:
    strict_protocols: true
  local:
    strict_protocols: true
```



.mrjob.conf

- ..



Dropbox Slides/AWS

Run the code in command line in AWS

- Check you configration file path
- Create .mrjob.conf
- Put your access key info in configuration file

Run an MRJob on AWS EMR from your local computer
STEP 1: set up access credentials

```
from mrjob import conf
conf.find_mrjob_conf()
```

```
'/Users/jshanahan/.mrjob.conf'
```

##Create or replace .mrjob.conf file

```
runners:
    emr:
        aws_access_key_id: your_access_key_id
        aws_secret_access_key: your_secret_access_key
```

1 master node +1/2 workers (cheap nodes)
5-10 minutes to provision the machines
2-3 minutes to run the job

Run the code in command line in AWS

```
!python WordCount.py WordCount.txt -r emr
```

```
using configs in /Users/jshanahan/.mrjob.conf
using existing scratch bucket mrjob-26635e5bb8bb690f
using s3://mrjob-26635e5bb8bb690f/tmp/ as our scratch dir on S3
creating tmp directory /var/folders/j4/95k348x940xcz40fkdmgy_n40000gn/T/WordCount.jshanahan.20160203.221800.233479
writing master bootstrap script to /var/folders/j4/95k348x940xcz40fkdmgy_n40000gn/T/WordCount.jshanahan.20160203.2218
00.233479/b.py

PLEASE NOTE: Starting in mrjob v0.5.0, protocols will be strict by default. It's recommended you run your job with --
strict-protocols or set up mrjob.conf as described at https://pythonhosted.org/mrjob/whats-new.html#ready-for-strict-
protocols

Copying non-input files into s3://mrjob-26635e5bb8bb690f/tmp/WordCount.jshanahan.20160203.221800.233479/files/
Waiting 5.0s for S3 eventual consistency
Creating Elastic MapReduce job flow
Job flow created with ID: j-1NFNBUU1OYOMT
Created new job flow j-1NFNBUU1OYOMT
```

Run the code in command line in A

- Check you configration file path
- Create .mrjob.conf
- Put your access key info in configuration file

```
from mrjob import conf
conf.find_mrjob_conf()
```

```
'/Users/jshanahan/.mrjob.conf'
```

##Create or replace .mrjob.conf file

```
runners:
    emr:
        aws_access_key_id: your_access_key_id
        aws_secret_access_key: your_secret_access_
```

Run the code in command line in A

```
!python WordCount.py WordCount.txt -r emr
```

```
using configs in /Users/jshanahan/.mrjob.conf
using existing scratch bucket mrjob-26635e5bb8bb69
using s3://mrjob-26635e5bb8bb690f/tmp/ as our scra
creating tmp directory /var/folders/j4/95k348x940x
writing master bootstrap script to /var/folders/j4
00.233479/b.py

PLEASE NOTE: Starting in mrjob v0.5.0, protocols wi
strict-protocols or set up mrjob.conf as described
protocols

Copying non-input files into s3://mrjob-26635e5bb8
Waiting 5.0s for S3 eventual consistency
Creating Elastic MapReduce job flow
Job flow created with ID: j-1NFNBUU1OYOMT
Created new job flow j-1NFNBUU1OYOMT
```

Run the code in command line in AWS

```
!python WordCount.py WordCount.txt -r emr
```

```
using configs in /Users/jshanahan/.mrjob.conf
using existing scratch bucket mrjob-26635e5bb8bb690f
using s3://mrjob-26635e5bb8bb690f/tmp/ as our scratch dir on S3
creating tmp directory /var/folders/j4/95k348x940xcz40fkdmgy_n40000gn/T/WordCount.jshanahan.20160203.221800.233479
writing master bootstrap script to /var/folders/j4/95k348x940xcz40fkdmgy_n40000gn/T/WordCount.jshanahan.20160203.221800.233479/b.py

PLEASE NOTE: Starting in mrjob v0.5.0, protocols will be strict by default. It's recommended you run your job with --strict-protocols or set up mrjob.conf as described at https://pythonhosted.org/mrjob/whats-new.html#ready-for-strict-protocols

Copying non-input files into s3://mrjob-26635e5bb8bb690f/tmp/WordCount.jshanahan.20160203.221800.233479/files/
Waiting 5.0s for S3 eventual consistency
Creating Elastic MapReduce job flow
Job flow created with ID: j-1NFNBUU1OYOMT
Created new job flow j-1NFNBUU1OYOMT
Job launched 30.8s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 62.2s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 93.1s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 124.6s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 155.4s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 186.4s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 217.3s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 248.6s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 279.5s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 310.4s ago, status RUNNING: Running step (WordCount.jshanahan.20160203.221800.233479: Step 1 of 1)
Job launched 341.2s ago, status RUNNING: Running step (WordCount.jshanahan.20160203.221800.233479: Step 1 of 1)
Job launched 372.7s ago, status RUNNING: Running step (WordCount.jshanahan.20160203.221800.233479: Step 1 of 1)
Job launched 403.5s ago, status RUNNING: Running step (WordCount.jshanahan.20160203.221800.233479: Step 1 of 1)
Job completed.
Running time was 113.0s (not counting time spent waiting for the EC2 instances)
ec2_key_pair_file not specified, going to S3
Fetching counters from S3...
Waiting 5.0s for S3 eventual consistency
Counters from step 1:
  File Input Format Counters :
    Bytes Read: 227
  File Output Format Counters :
    Bytes Written: 82
  FileSystemCounters:
    FILE_BYTES_READ: 116
    FILE_BYTES_WRITTEN: 134262
    HDFS_BYTES_READ: 588
    S3_BYTES_READ: 227
    S3_BYTES_WRITTEN: 82
  Job Counters :
    Launched map tasks: 4
    Launched reduce tasks: 1
    Rack-local map tasks: 4
    SLOTS_MILLIS_MAPS: 76840
    SLOTS_MILLIS_REDUCES: 38567
    Total time spent by all maps waiting after reserving slots (ms): 0
    Total time spent by all reduces waiting after reserving slots (ms): 0
  Map-Reduce Framework:
    CPU time spent (ms): 4790
    Combine input records: 0
    Combine output records: 0
    Map input bytes: 89
    Map input records: 3
    Map output bytes: 178
    Map output materialized bytes: 187
    Map output records: 18
    Physical memory (bytes) snapshot: 858865664
    Reduce input groups: 9
    Reduce input records: 18
    Reduce output records: 9
    Reduce shuffle bytes: 187
    SPLIT_RAW_BYTES: 588
    Spilled Records: 36
    Total committed heap usage (bytes): 606822400
    Virtual memory (bytes) snapshot: 3211935744
Streaming final output from s3://mrjob-26635e5bb8bb690f/tmp/WordCount.jshanahan.20160203.221800.233479/output/
"bar"   1
"data"  2
"foo"   4
"is"    1
"jimi"  5
"labs"  1
"mining"        1
"quux"  2
"science"       1
removing tmp directory /var/folders/j4/95k348x940xcz40fkdmgy_n40000gn/T/WordCount.jshanahan.20160203.221800.233479
Removing all files in s3://mrjob-26635e5bb8bb690f/tmp/WordCount.jshanahan.20160203.221800.233479/
Removing all files in s3://mrjob-26635e5bb8bb690f/tmp/logs/j-1NFNBUU1OYOMT/
Terminating job flow: j-1NFNBUU1OYOMT
```

Run the code in command line in AWS

```
!python WordCount.py WordCount.txt -r emr
```

```
using configs in /Users/jshanahan/.mrjob.conf
using existing scratch bucket mrjob-26635e5bb8bb690f
using s3://mrjob-26635e5bb8bb690f/tmp/ as our scratch dir on S3
creating tmp directory /var/folders/j4/95k348x940xcz40fkdmgy_n40000gn/T/WordCount.jshanahan.20160203.221800.233479
writing master bootstrap script to /var/folders/j4/95k348x940xcz40fkdmgy_n40000gn/T/WordCount.jshanahan.20160203.2218
00.233479/b.py

PLEASE NOTE: Starting in mrjob v0.5.0, protocols will be strict by default. It's recommended you run your job with --
strict-protocols or set up mrjob.conf as described at https://pythonhosted.org/mrjob/whats-new.html#ready-for-strict-
protocols

Copying non-input files into s3://mrjob-26635e5bb8bb690f/tmp/WordCount.jshanahan.20160203.221800.233479/files/
Waiting 5.0s for S3 eventual consistency
Creating Elastic MapReduce job flow
Job flow created with ID: j-1NFNBUU1OYOMT
Created new job flow j-1NFNBUU1OYOMT
Job launched 30.8s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 62.2s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 93.1s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 124.6s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 155.4s ago, status STARTING: Provisioning Amazon EC2 capacity
Job launched 186.4s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 217.3s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 248.6s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 279.5s ago, status BOOTSTRAPPING: Running bootstrap actions
Job launched 310.4s ago, status RUNNING: Running step (WordCount.jshanahan.20160203.221800.233479: Step 1 of 1)
Job launched 341.2s ago, status RUNNING: Running step (WordCount.jshanahan.20160203.221800.233479: Step 1 of 1)
Job launched 372.7s ago, status RUNNING: Running step (WordCount.jshanahan.20160203.221800.233479: Step 1 of 1)
Job launched 403.5s ago, status RUNNING: Running step (WordCount.jshanahan.20160203.221800.233479: Step 1 of 1)
Job completed.
Running time was 113.0s (not counting time spent waiting for the EC2 instances)
ec2_key_pair_file not specified, going to S3
Fetching counters from S3...
Waiting 5.0s for S3 eventual consistency
Counters from step 1:
  File Input Format Counters :
    Bytes Read: 227
  File Output Format Counters :
    Bytes Written: 82
  FileSystemCounters:
    FILE_BYTES_READ: 116
    FILE_BYTES_WRITTEN: 134262
    HDFS_BYTES_READ: 588
    S3_BYTES_READ: 227
    S3_BYTES_WRITTEN: 82
  Job Counters :
    Launched map tasks: 4
    Launched reduce tasks: 1
    Rack-local map tasks: 4
    SLOTS_MILLIS_MAPS: 76840
    SLOTS_MILLIS_REDUCES: 38567
    Total time spent by all maps waiting after reserving slots (ms): 0
    Total time spent by all reduces waiting after reserving slots (ms): 0
  Map-Reduce Framework:
    CPU time spent (ms): 4790
    Combine input records: 0
    Combine output records: 0
    Map input bytes: 89
    Map input records: 3
    Map output bytes: 178
    Map output materialized bytes: 187
    Map output records: 18
    Physical memory (bytes) snapshot: 858865664
    Reduce input groups: 9
    Reduce input records: 18
    Reduce output records: 9
    Reduce shuffle bytes: 187
    SPLIT_RAW_BYTES: 588
    Spilled Records: 36
    Total committed heap usage (bytes): 606822400
    Virtual memory (bytes) snapshot: 3211935744
Streaming final output from s3://mrjob-26635e5bb8bb690f/tmp/WordCount.jshanahan.20160203.221800.233479/output/
"bar"   1
"data"  2
"foo"   4
"is"    1
"jimi"  5
"labs"  1
"mining"        1
"quux"  2
"science"       1
removing tmp directory /var/folders/j4/95k348x940xcz40fkdmgy_n40000gn/T/WordCount.jshanahan.20160203.221800.233479
Removing all files in s3://mrjob-26635e5bb8bb690f/tmp/WordCount.jshanahan.20160203.221800.233479/
Removing all files in s3://mrjob-26635e5bb8bb690f/tmp/logs/j-1NFNBUU1OYOMT/
Terminating job flow: j-1NFNBUU1OYOMT
```

Number of maptasks

- **Set according to the total size of the input and the block size, i.e. the number of the splits. even though you set the number of map task, that is just a hint.**
  - The number of reduce task can be user defined, and if it is not defined explicitly, the default reduce number is 1.
  - The number of map tasks is dependent on the data volume, block size and split size.
  - For example: If you have block size 128 MB and your file size is 1 GB then there will be 8 number of map tasks. You can control it by using split size.
- **The right level of parallelism for maps seems to be around 10-100 maps/node, although we have taken it up to 300 or so for very cpu-light map tasks.**
- **Task setup takes awhile, so it is best if the maps take at least a minute to execute.**

https://wiki.apache.org/hadoop/HowManyMapsAndReduces

- Actually controlling the number of maps is subtle. The mapred.map.tasks parameter is just a hint to the InputFormat for the number of maps. The default InputFormat behavior is to split the total number of bytes into the right number of fragments.
- However, in the default case the DFS block size of the input files is treated as an upper bound for input splits.
- A lower bound on the split size can be set via mapred.min.split.size.
- Thus, if you expect 10TB of input data and have 128MB DFS blocks, you'll end up with 82k maps, unless your mapred.map.tasks is even larger.
- Ultimately the InputFormat determines the number of maps.

AWS CLI installation

Step 1:

- **sudo pip install awscli**


```
  312202 googlebooks-eng-all-5gram-20090715-93-filtered.txt
  312245 googlebooks-eng-all-5gram-20090715-94-filtered.txt
  312630 googlebooks-eng-all-5gram-20090715-95-filtered.txt
  312592 googlebooks-eng-all-5gram-20090715-96-filtered.txt
  311714 googlebooks-eng-all-5gram-20090715-97-filtered.txt
  312118 googlebooks-eng-all-5gram-20090715-98-filtered.txt
  312354 googlebooks-eng-all-5gram-20090715-99-filtered.txt
 58682266 total
JAMES-SHANAHANs-Desktop-Pro-2:filtered-5Grams jshanahan$ pwd
/Users/jshanahan/Dropbox/Lectures-UC-Berkeley-ML-Class-2015/Slides/Homework-with-solutions/hw5/HW5-Questions/filtered-5Grams
JAMES-SHANAHANs-Desktop-Pro-2:filtered-5Grams jshanahan$
```

```
AWS Secret Access Key [None]: 0aQ1S7wQYg1D/GbnvoeMRM4X61lgpFwkTGhPhYUb
Default region name [None]:
Default output format [None]:
JAMES-SHANAHANs-Desktop-Pro-2:filtered-5Grams jshanahan$
JAMES-SHANAHANs-Desktop-Pro-2:filtered-5Grams jshanahan$
JAMES-SHANAHANs-Desktop-Pro-2:filtered-5Grams jshanahan$ aws s3 sync s3://filtered-5grams .
download: s3://filtered-5grams/googlebooks-eng-all-5gram-20090715-1-filtered.txt to ./googlebooks-eng-all-5gram-20090715-1-filtered.txt
download: s3://filtered-5grams/googlebooks-eng-all-5gram-20090715-101-filtered.txt to ./googlebooks-eng-all-5gram-20090715-101-filtered.txt
download: s3://filtered-5grams/googlebooks-eng-all-5gram-20090715-102-filtered.txt to ./googlebooks-eng-all-5gram-20090715-102-filtered.txt
download: s3://filtered-5grams/googlebooks-eng-all-5gram-20090715-10-filtered.txt to ./googlebooks-eng-all-5gram-20090715-10-filtered.txt
download: s3://filtered-5grams/googlebooks-eng-all-5gram-20090715-100-filtered.txt to ./googlebooks-eng-all-5gram-20090715-100-filtered.txt
download: s3://filtered-5grams/googlebooks-eng-all-5gram-20090715-0-filtered.txt to ./googlebooks-eng-all-5gram-20090715-0-filtered.txt
download: s3://filtered-5grams/googlebooks-eng-all-5gram-20090715-103-filtered.txt to ./googlebooks-eng-all-5gram-20090715-103-filtered.txt
download: s3://filtered-5grams/googlebooks-eng-all-5gram-20090715-104-filtered.txt to ./googlebooks-eng-all-5gram-20090715-104-filtered.txt
download: s3://filtered-5grams/googlebooks-eng-all-5gram-20090715-105-filtered.txt to ./googlebooks-eng-all-5gram-20090715-105-filtered.txt
```

Step 2:

Python -r emr See Live Lecture 4 and related slides

Quick Configuration

For general use, the `aws configure` command is the fastest way to setup your AWS CLI installation.

```
$ aws configure
AWS Access Key ID [None]: AKI
AWS Secret Access Key [None]:
Default region name [None]: us-west-2
Default output format [None]: json
```

The AWS CLI will prompt you for four pieces of information. AWS Access Key ID and AWS Secret Access Key are your account credentials. If you don't have keys, see the Getting Set Up section earlier in this guide.

Default region is the name of the region you want to make calls against by default. This is usually the region closest to you, but it can be any region.

DropBox: 5NGram dataset

- There are over 58 million rows over 191 PART files (2Gig of data):

```
 wc -l *
......
  312118 googlebooks-eng-all-5gram-20090715-98-filtered.txt
  312354 googlebooks-eng-all-5gram-20090715-99-filtered.txt
 58,682,266 total


JAMES-SHANAHANs-Desktop-Pro-2:filtered-5Grams
jshanahan$ ls -l |wc -l
     191
```

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW3, HW4, HW5, HW5
  - AWS: (Not necessary for week 6)
  - Synonym detection in Map-Reduce: Case studies + Metrics
  - Async lecture recap plus Q&A [Jimi]
  - MLE, MAP, Bayesian optimal classifier
  - MLE for Gaussian Mixture Models
  - EM for Gaussian Mixture Models
  - EM Algorithm for a mixture of Bernoulli distributions
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

Patrick's Question

Just when I thought I'm done with the Jaccard implementation, I then realized I got something misunderstood. So I am to confirm something basic here.

Let's use a **simpler** version of the instruction:
"Build stripes for the most frequent 4 words using cooccurrence information based on the words ranked from 3-4 as a basis/vocabulary

Here is the list of the words sorted by frequency:
a, b, c, d, e, f

So my most frequent 4 words will be: **a, b, c, d**
And my vocab will be: **c, d**

And here are my two 5-grams (assume count = 1)
a b c c f
e f d a c

Using this toy example, my stripes (**before** binarized) will be:
a: {c:3, d:1}
b: {c:2}
c: {d:1}
d: {c:1}

Is that correct?

"Build stripes for the most frequent 4 words using cooccurrence information based on the words ranked from 3-4 as a basis/vocabulary. Here is the list of the words sorted by frequency:
a, b, c, d, e, f

So my most frequent 4 words will be**:** **a, b, c, d**
And my vocab will be**:** **c, d**

And here are my two 5-grams (assume count = 1)
a b c c f
e f d a c

Using this toy example, my stripes (before binarized) will be:

(A)
a: {c:3, d:1}
b: {c:2}
c: {d:1}
d: {c:1}

(B)
c: {d:1}
d: {c:1}

(C)
c: {d:1}

(D)
c: {a:1,b:1,d:1}
d:{a:1,c:1}

Question 1: Limit our stripe representation to {c, d} onl
YES {c, d} or NO {a, b,c,d}
{c, d}
The, for, Hadoop, spark : (limit the vocab to hadoop, spark)
10^9 dcouments: docFreq (the) = 10^9; hadoop: rare 10^3

Question 2: do we build stripes for a, b, c and d? Or just c and d?

"Build stripes for the most frequent 4 words using cooccurrence information based on the words ranked from 3-4 as a basis/vocabulary. Here is the list of the words sorted by frequency:
a, b, c, d, e, f

So my most frequent 4 words will be**:** **a, b, c, d**
And my vocab will be**:** **c, d**

And here are my two 5-grams (assume count = 1)
a b c c f
e f d a c

Using this toy example, my stripes (before binarized) will be:

(A)
a: {c:3, d:1}
b: {c:2}
c: {d:1}
d: {c:1}

(B)
c: {d:1}
d: {c:1}

(C)
c: {d:1}

(D)
c: {a:1,b:1,d:1}
d:{a:1,c:1}

Question 1: Limit our stripe representation to {c, d} onl
YES {c, d} or NO {a, b,c,d}
YES is the correct answer.
The, for, Hadoop, spark : (limit the vocab to hadoop, spark)
10^9 dcouments: docFreq (the) = 10^9; hadoop: rare
Posting list for "the" is going to be 10^9 in length; #pairs 10^{9+9};
Posting list for "hadoop" is going to be 10^3 in length; #pairs1

Question 2: do we build stripes for a, b, c and d?
Instructions YES;
Utility might be very limited in practice, so in practice NO
In HW 5.4: I will accept both YES and NO

Computing Pairwise Document Similarity in Large Collections:

A MapReduce Perspective

https://terpconnect.umd.edu/~oard/pdf/acl08elsayed2.pdf

Tamer Elsayed, Jimmy Lin, and Douglas W. Oard

Adapted from a talk at
iSchool, Cloud Computing Class Talk, Oct 6th 2008

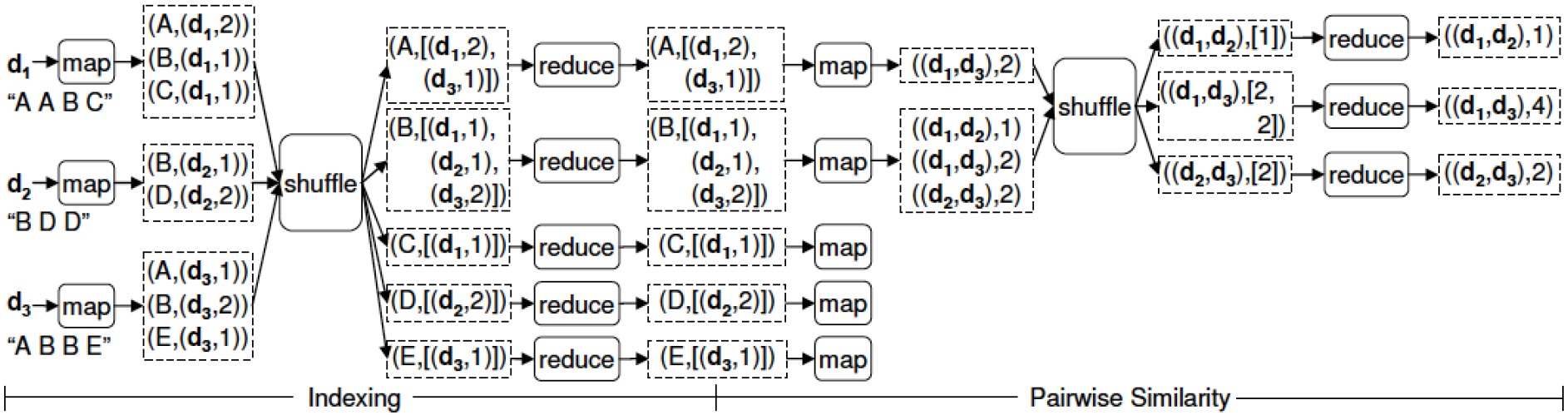


Figure 2: Computing pairwise similarity of a toy collection of 3 documents. A simple term weighting scheme ($w_{t,d} = tf_{t,d}$) is chosen for illustration.

First Dataset: Acquaint Corpus

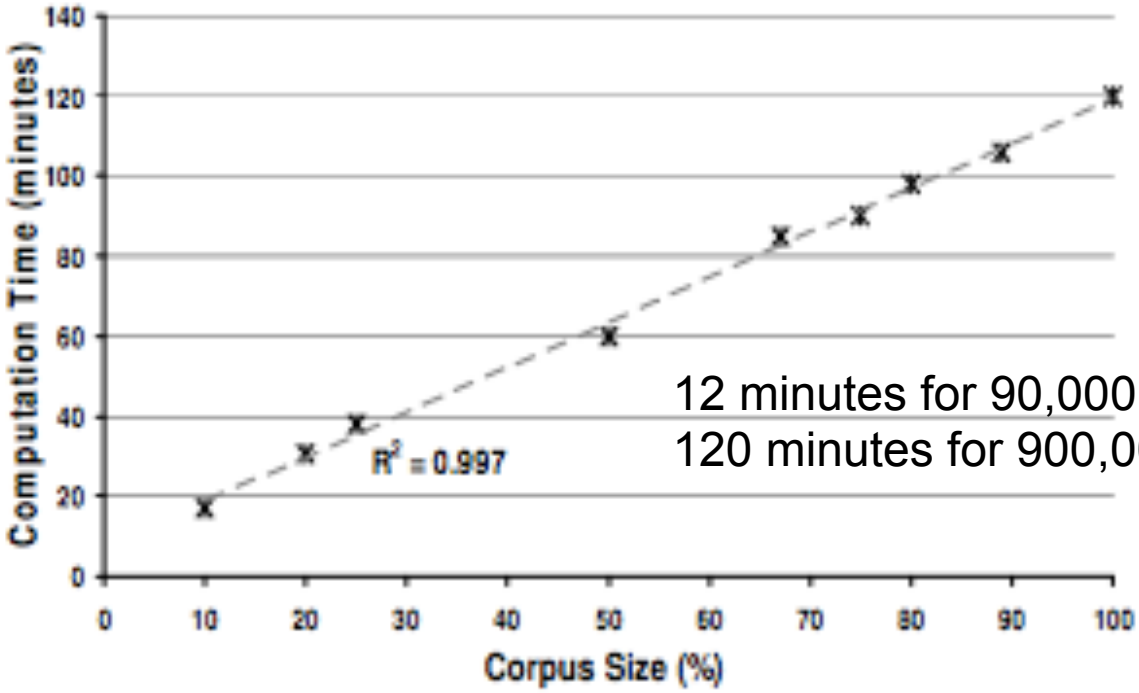
- **900,000 documents**
- **2.5 gigabytes of data**

- **Preprocessing**
  - Performed stopword removal (using Lucene's stopword list)
  - Stemming was performed: walking, walked, walker → walk
  - Resulting in total vocabulary size of 909,326

2 hours for 900,000 documents
Pair wise document similarity phase

$10^6 (10^6 - 1) = 10^{12}$ Trillion similarity ops
20 node cluster

- ..



120 minutes

12 minutes for 90,000 document
120 minutes for 900,000 documents

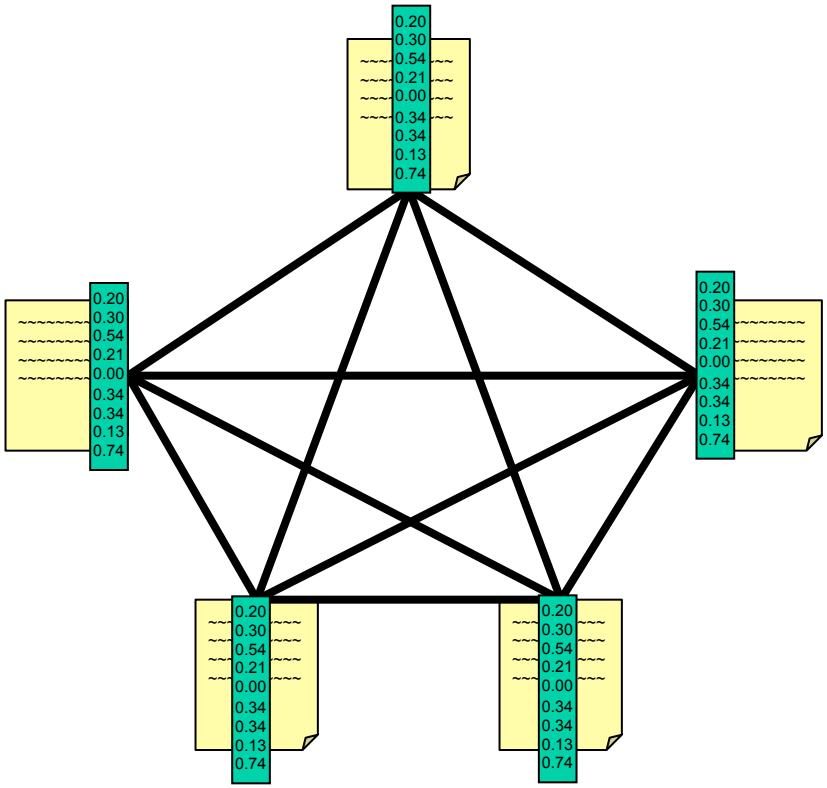
Figure 3: Running time of pairwise similarity comparisons, for subsets of AQUAINT-2.

10,000 stripe documents pairwise similarity should only a couple of minutes (with 1,000 terms in the vocabulary Rank 9,000-10,000) takes 5 minutes on 4 node cluster: 1 master 3 slaves)? Constructing the inverted index takes about 30 minutes.

Overview

- Abstract Problem
- Trivial Solution
- MapReduce Solution
- Efficiency Tricks

Abstract Problem: pairwise similarity of objects

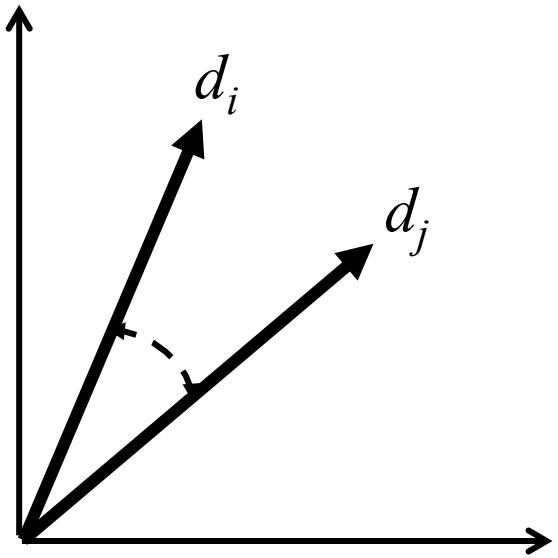


- Applications:
  - Clustering
  - Coreference resolution
  - "more-like-that" queries

Similarity of Documents

$$sim(d_i, d_j) = \sum_{t \in V} w_{t,d_i} w_{t,d_j}$$

- **Simple inner product**
- **Cosine similarity**
- **Term weights**
  - Standard problem in IR
  - tf-idf, BM25, etc.



Trivial Solution

$$sim(d_i, d_j) = \sum_{t \in V} w_{t,d_i} w_{t,d_j}$$

- **load each vector o(*N*) times**
- **load each term o(df_t^2) times**

Goal

scalable and efficient solution for large collections

Better Solution

Each term contributes only if appears in $d_i \cap d_j$

$$sim(d_i, d_j) = \sum_{t \in d_i \cap d_j} w_{t,d_i} w_{t,d_j}$$

$$sim(d_i, d_j) = \sum_{t \in d_i \cap d_j} term\_contrib(t, d_i, d_j)$$

- **Load weights for each term once**
- **Each term contributes o*(df_t^2)* partial scores**
- **Allows efficiency tricks**

Decomposition → MapReduce

Each term contributes only if appears in $d_i \cap d_j$

$$sim(d_i, d_j) = \sum_{t \in d_i \cap d_j} w_{t,d_i} w_{t,d_j}$$

reduce

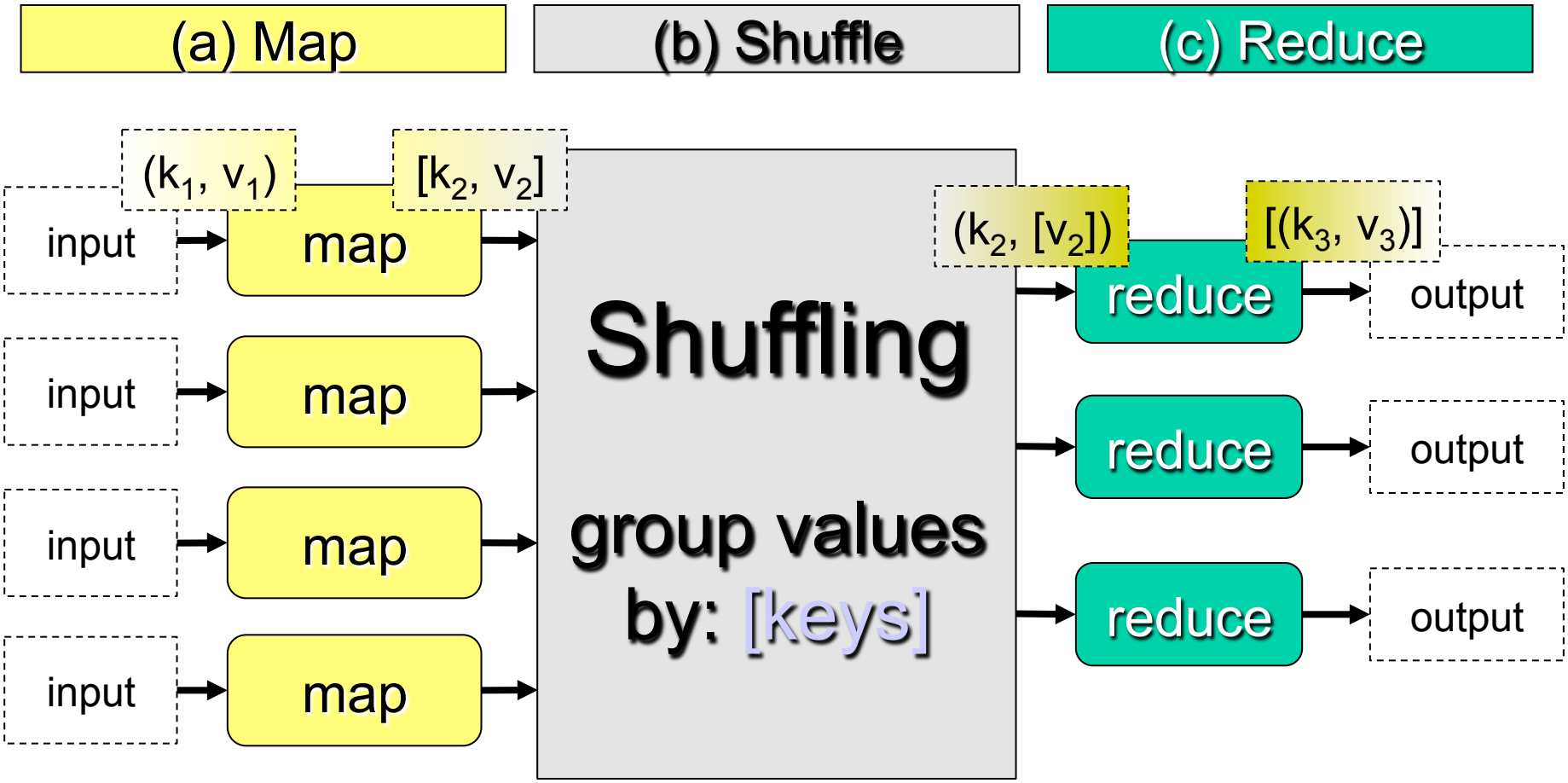
$$sim(d_i, d_j) = \sum_{t \in d_i \cap d_j} term\_contrib(t, d_i, d_j)$$

index

map

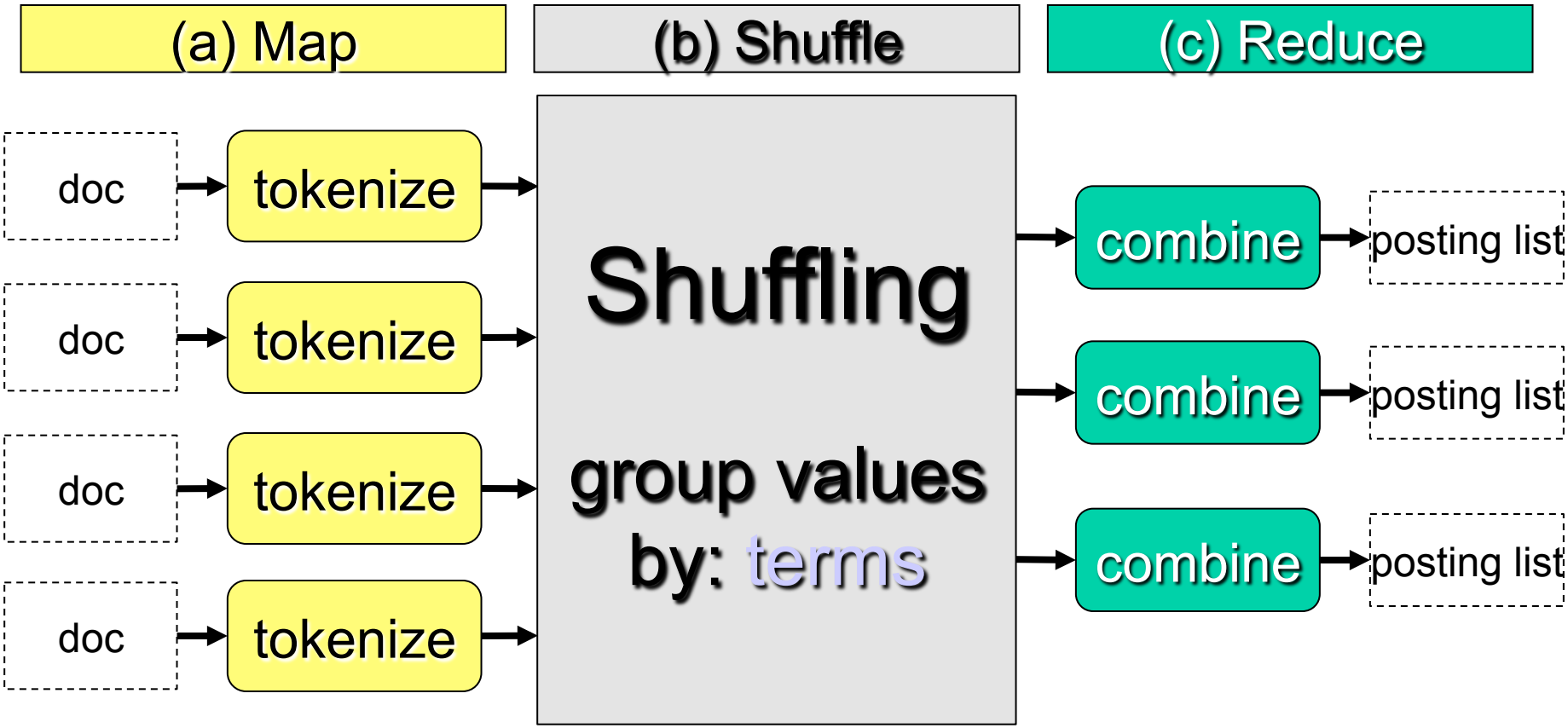
- **Load weights for each term once**
- **Each term contributes o(df_t^2) partial scores**

MapReduce Framework



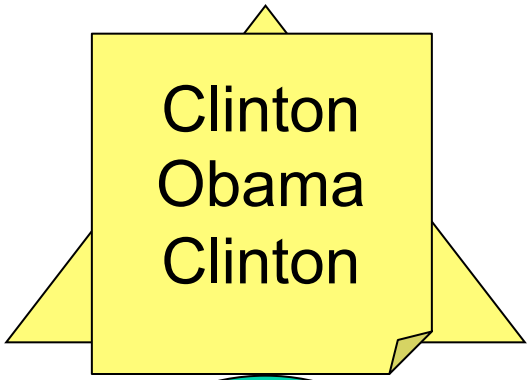
handles low-level details **transparently**

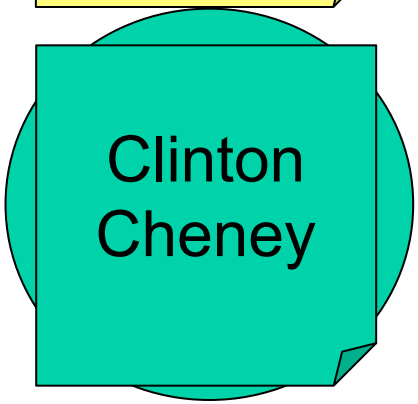
Standard inverted index construction

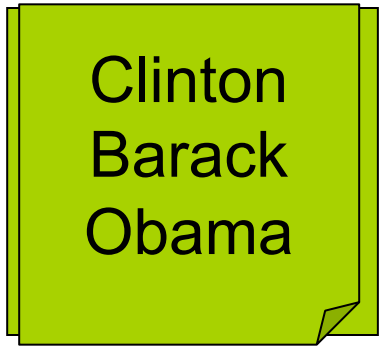


$$sim(d_i, d_j) = \sum_{t \in d_i \cap d_j} term\_contrib(t, d_i, d_j)$$

Indexing (3-doc toy collection)





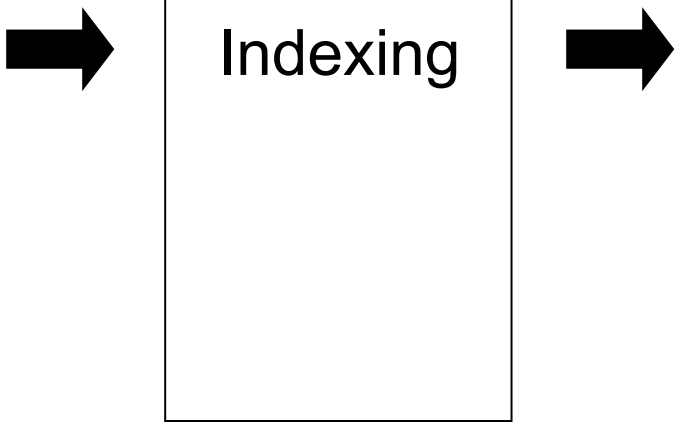


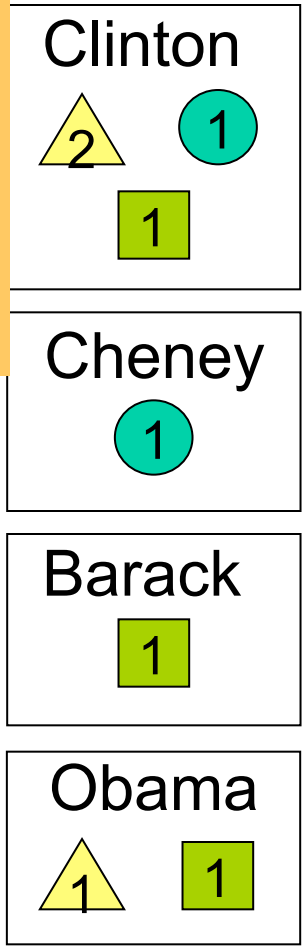
Inverted Index

Clinton: {d1:2, d2:1, d3:1} → $3_{choose}2$=3 pairs
Cheney:{d2:1}
Obama {d1:1, d3:1} 1 pair
Barack {d3:1}

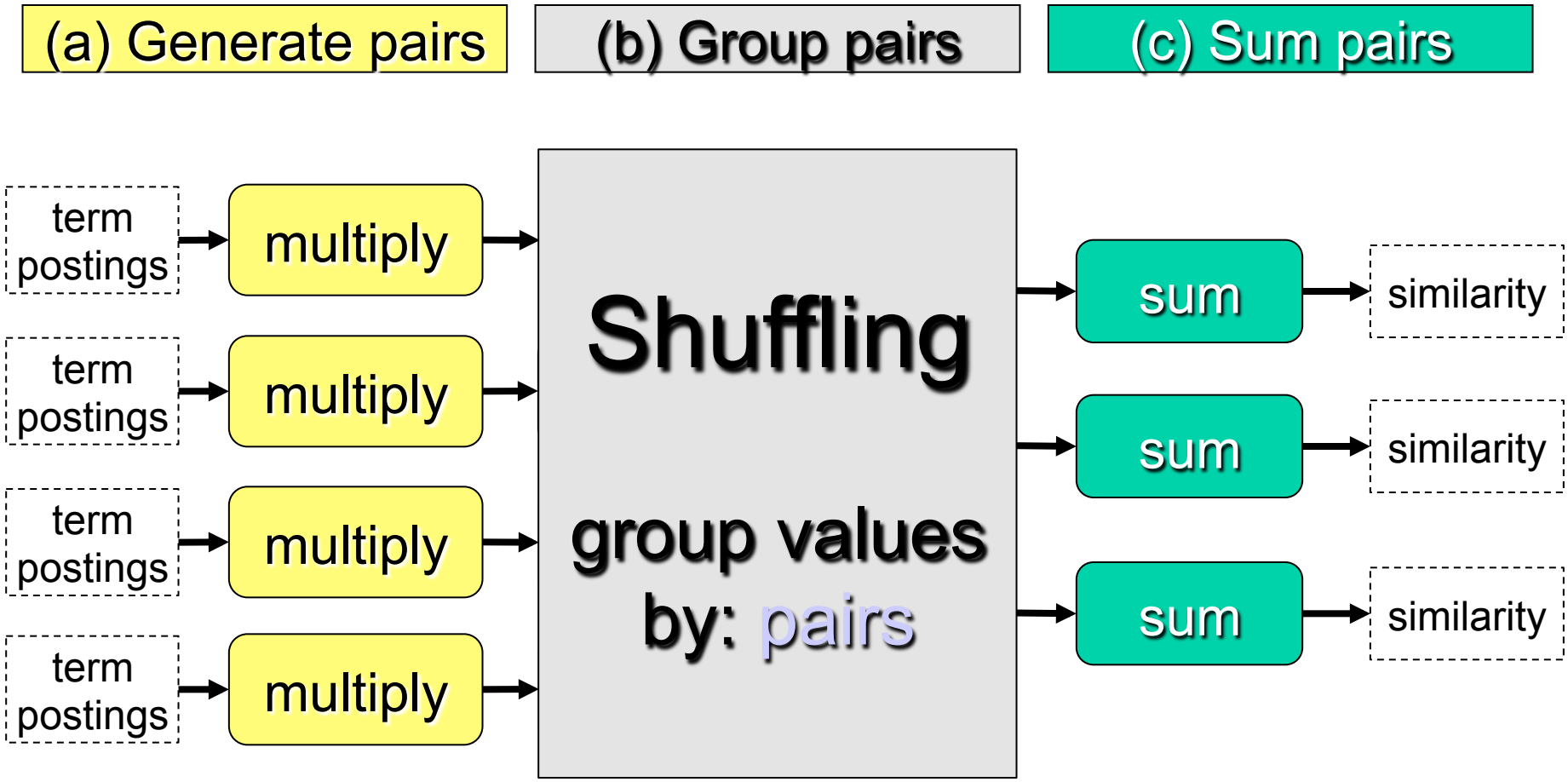
Pairwise similarity between documents

Similarity (d1, d3) =

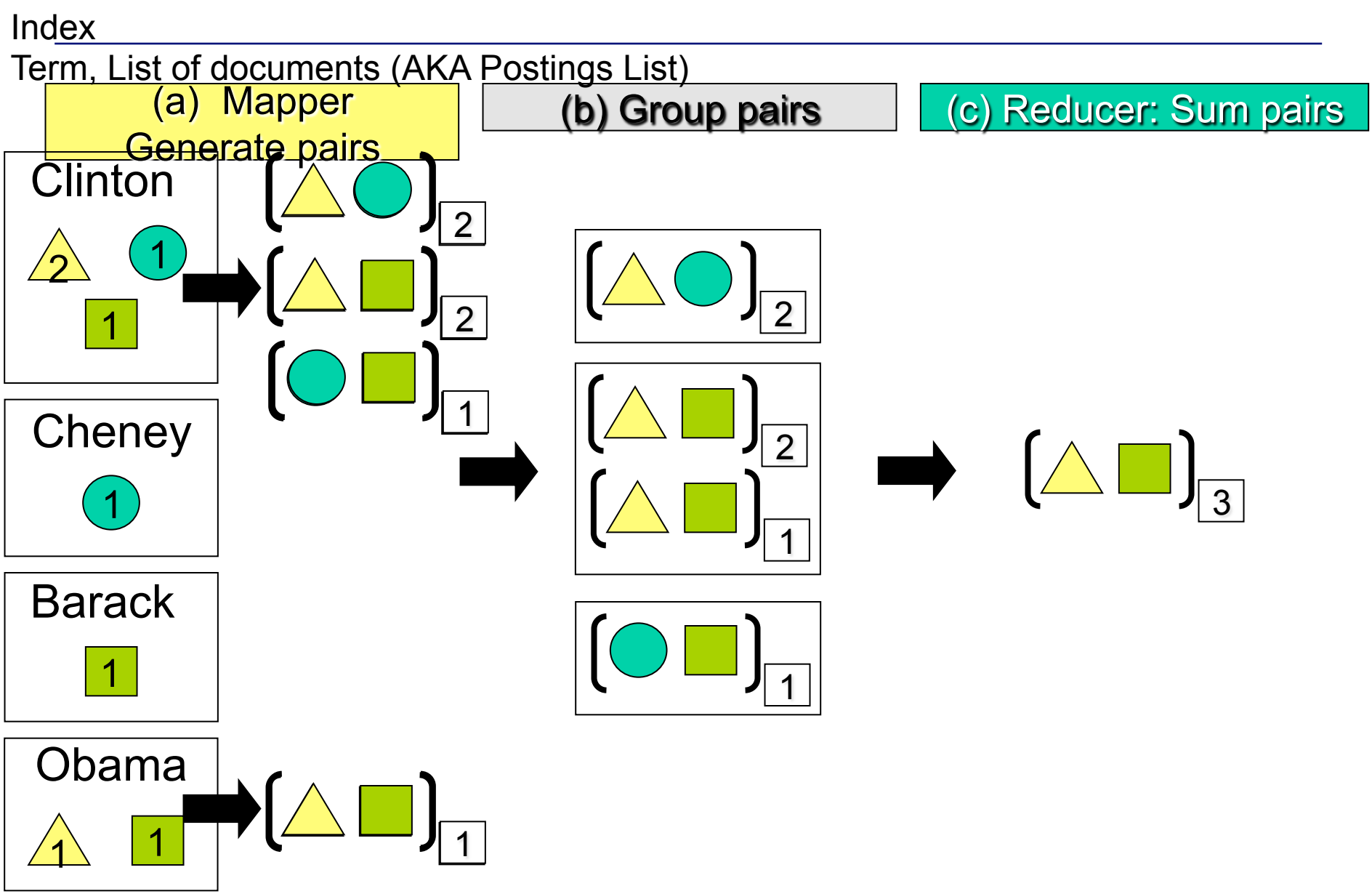




Pairwise Similarity (abstract)



Pairwise Similarity via inverted index

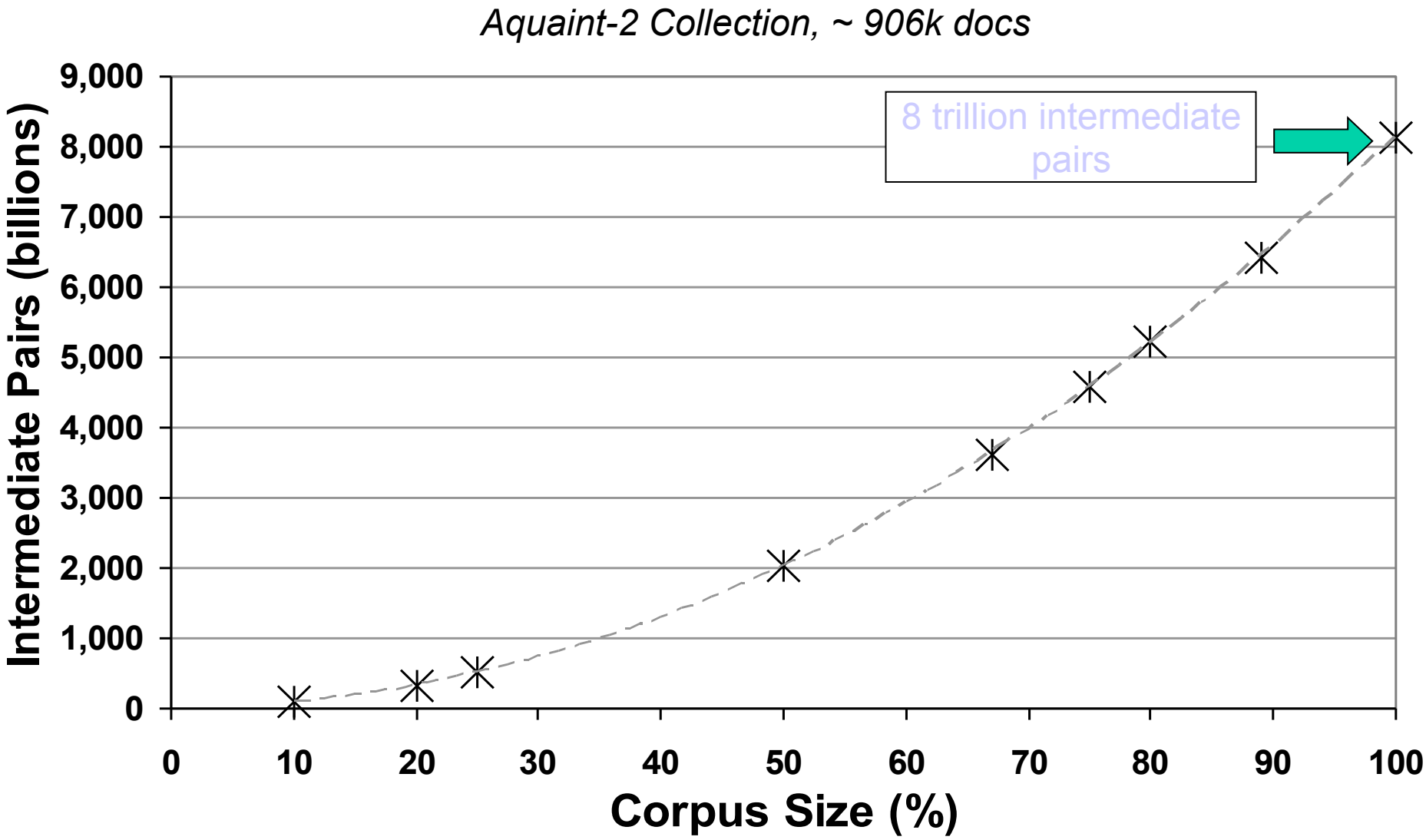


Experimental Setup

Elsayed, Lin, and Oard, ACL 2008

- **0.16.0**
  - Open source MapReduce implementation
- **Cluster of 19 machines**
  - Each w/ two processors (single core)
- **Aquaint-2 collection**
  - 906K documents
  - Medline4
- **Okapi BM25**
- **Subsets of collection**

Efficiency (disk space)



Hadoop, 19 PCs, each: 2 single-core processors, 4GB memory, 100GB disk

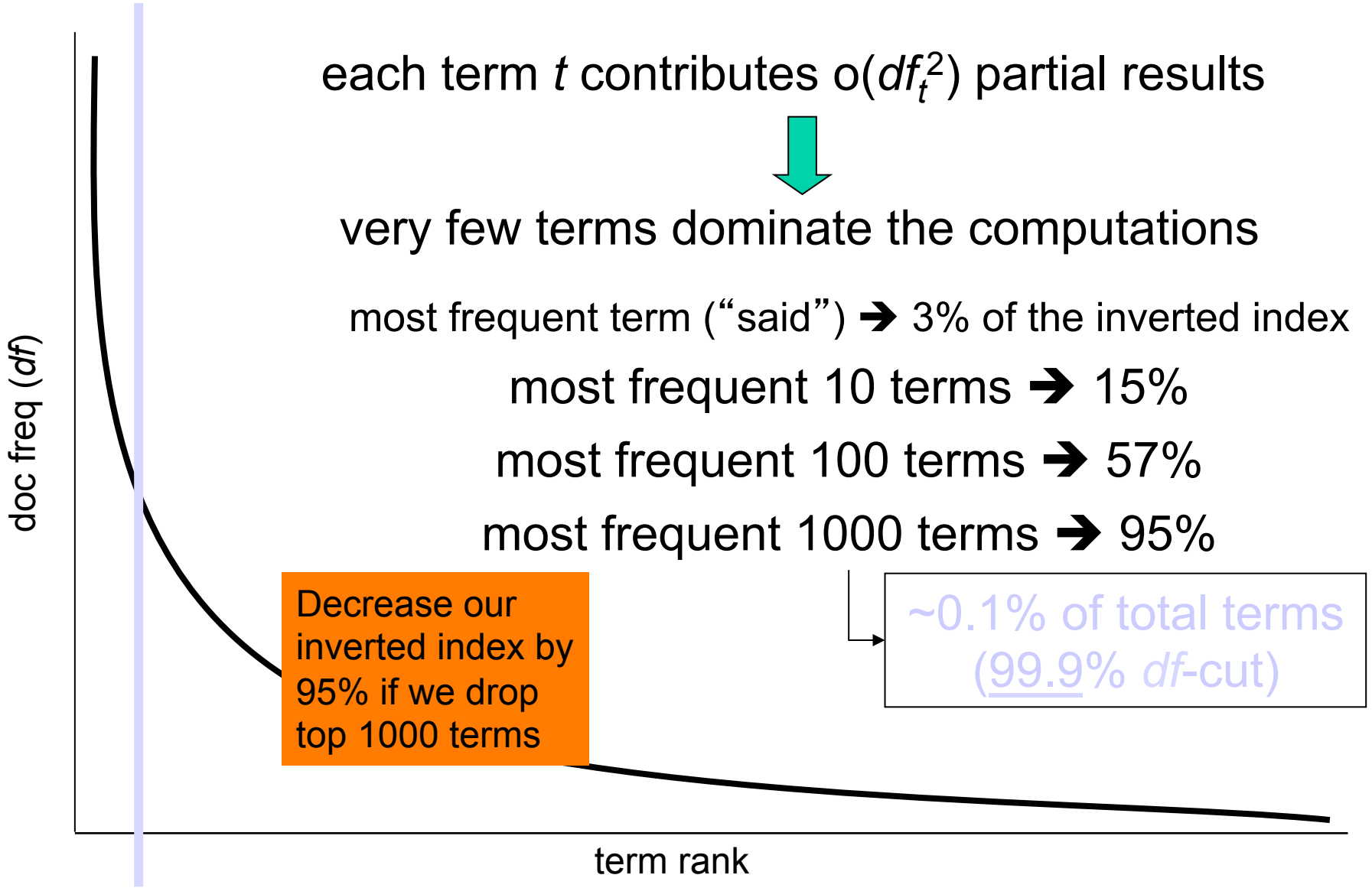
Remove common terms

- **Vocabulary size is 909,326 (after stopword removal and lower casing)**
- **1% of vocabulary accounts for 9,093 terms**
- **0.1% of vocabulary accounts for 1,000 t**

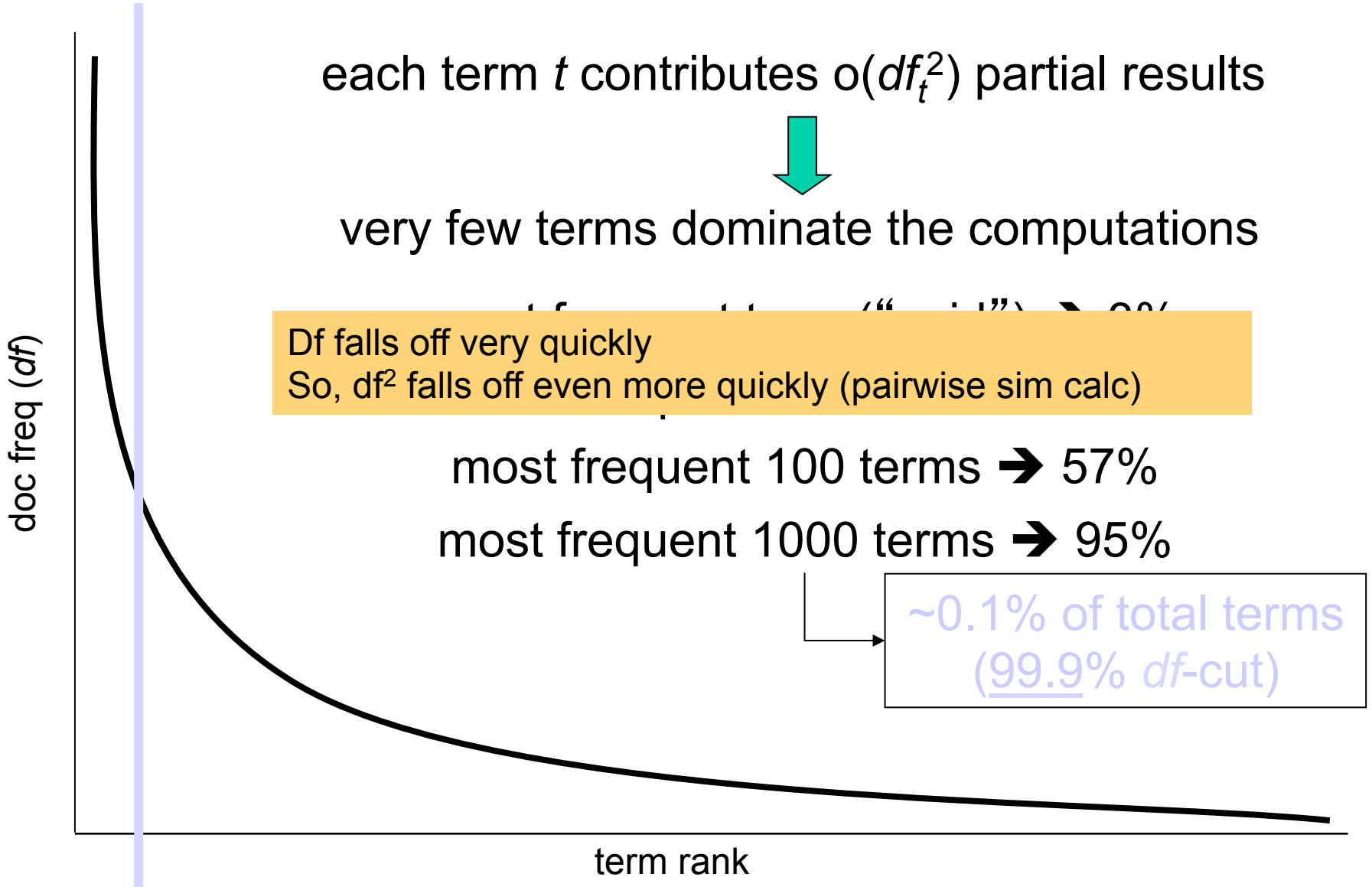
~0.1% of total terms (99.9% *df*-cut)

- **use** df**-cut to remove common terms.**
  - After stopword removal (using Lucene's stopword list), implemented a df-cut, where a fraction of the terms with the highest document frequencies is eliminated. This has the effect of removing non-discriminative terms.
  - we adopt a 99% cut, which means that the most frequent1% of terms were discarded (9,093 terms out of a total vocabulary size of 909,326).

Terms: Zipfian Distribution

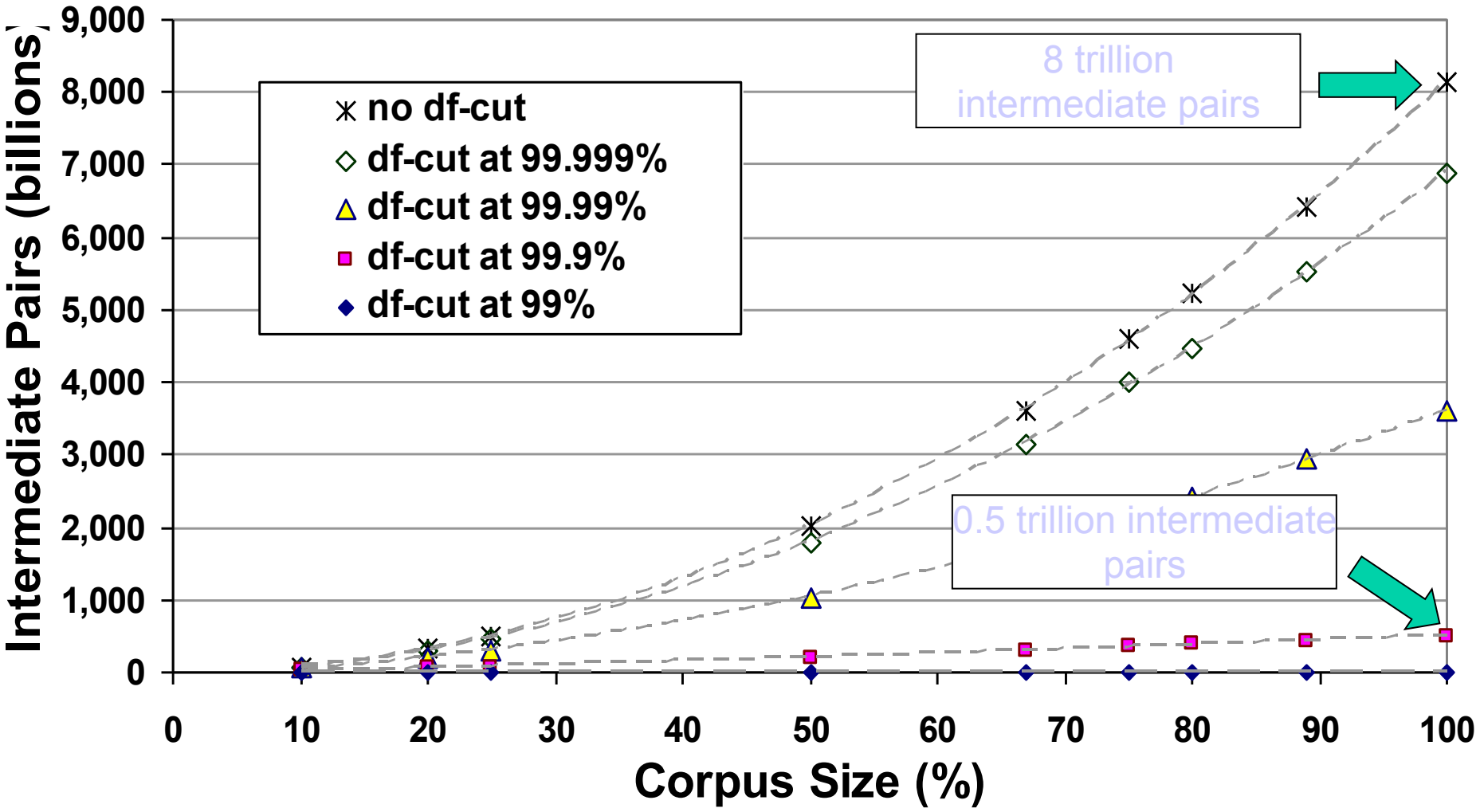


Terms: Zipfian Distribution



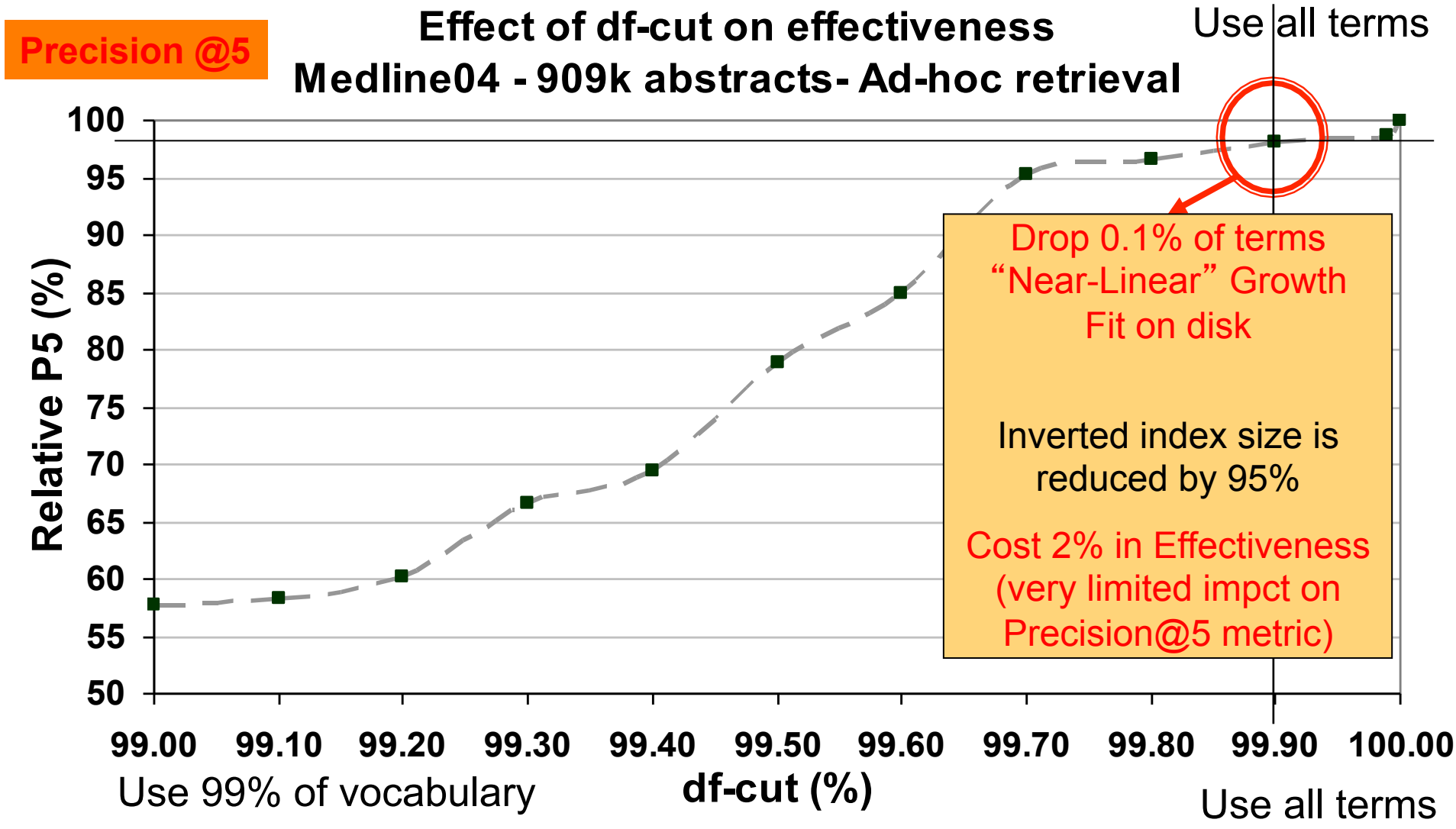
Efficiency (disk space)

Aquaint-2 Collection, ~ 906k doc



Hadoop, 19 PCs, each w/: 2 single-core processors, 4GB memory, 100GB disk

Relative Effectiveness on the Medline Ad-hoc IR task



Hadoop, 19 PCs, each w/: 2 single-core processors, 4GB memory, 100GB disk

Implementation Issues

- **BM25s Similarity Model**

$$\frac{tf_1}{tf_1 + 0.5 + 1.5\frac{dl_1}{avg\ dl}} * \frac{tf_1}{tf_1 + 0.5 + 1.5\frac{dl_1}{avg\ dl}} * \log\frac{N - df + 0.5}{df + 0.5}$$

  - TF, IDF
  - Document length
- **DF-Cut**
  - Build a histogram
  - Pick the absolute *df for the % df*-cut

Other Approximation Techniques ?

Other Approximation Techniques

(2) Absolute *df*

- **Consider only terms that appear in at least *n (or %)* documents**
  - An ***absolute lower bound on df***, instead of just removing the % most-frequent terms

Other Approximation Techniques

(3) *tf*-Cut

- Consider only documents (in posting list) with tf > T ; T=1 or 2
- OR: Consider only the top N documents based on tf for each term

Other Approximation Techniques

(4) Similarity Threshold

- **Consider only partial scores > Sim_T**

Other Approximation Techniques:

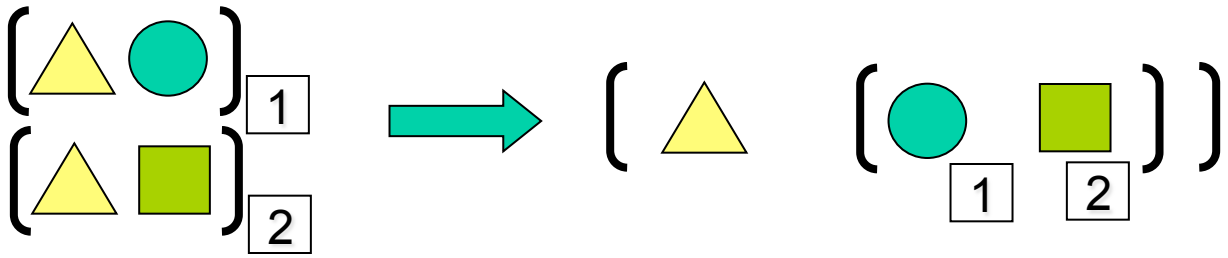
(5) Ranked List

- **Keep only the most similar N documents**
  - In the reduce phase
- **Good for ad-hoc retrieval and "more-like this" queries**

Space-Saving Tricks

(1) Stripes

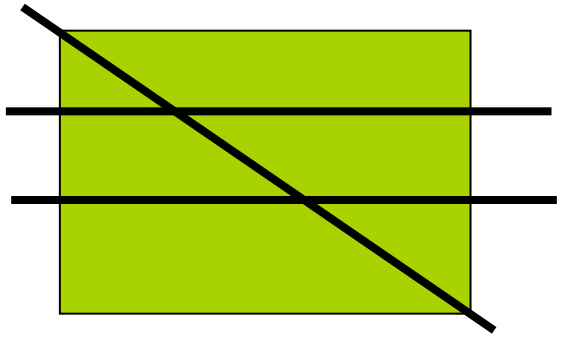
- Stripes instead of pairs
- Group by doc-id not pairs



Space-Saving Tricks

(2) Blocking

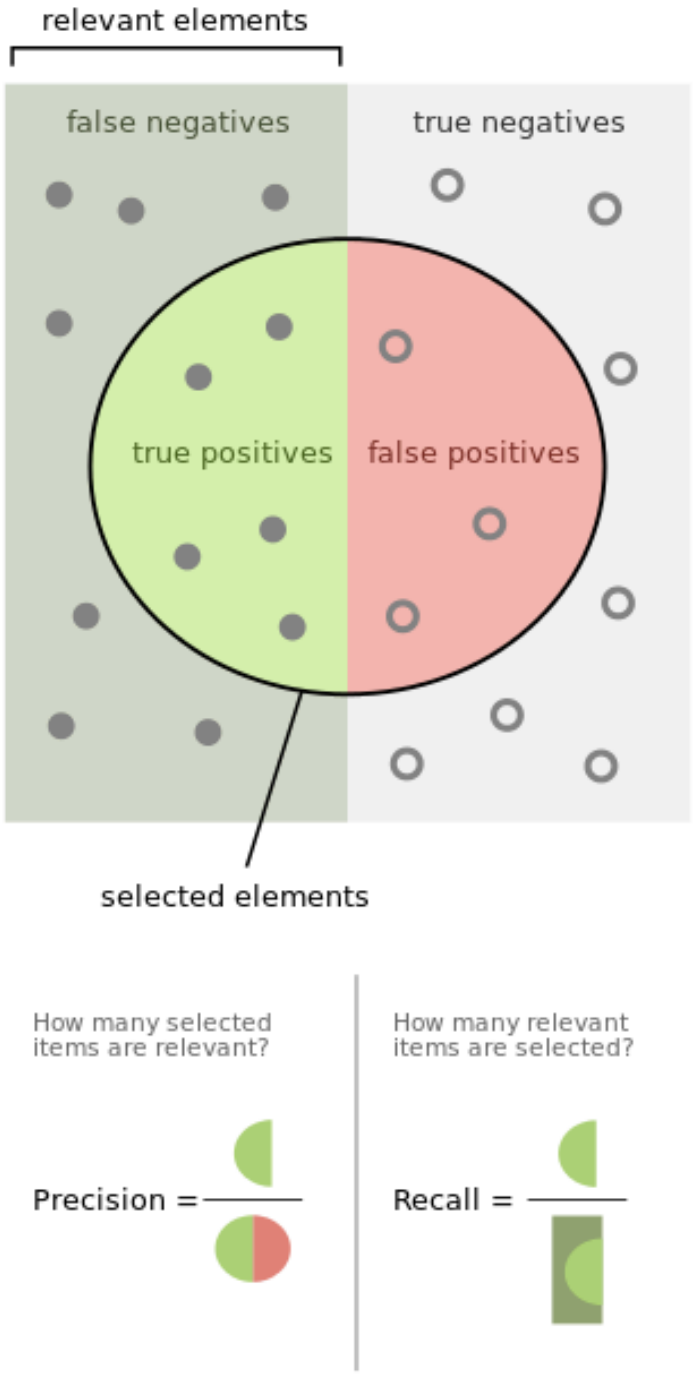
- No need to generate the whole matrix at once
- Generate different blocks of the matrix at different steps ➔ limit the max space required for intermediate results



Similarity Matrix

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW3, HW4, HW5, HW5
  - AWS: (Not necessary for week 6)
  - Synonym detection in Map-Reduce: Case studies + Metrics
  - Async lecture recap plus Q&A [Jimi]
  - MLE, MAP, Bayesian optimal classifier
  - MLE for Gaussian Mixture Models
  - EM for Gaussian Mixture Models
  - EM Algorithm for a mixture of Bernoulli distributions
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting



Precision Recall

In pattern recognition and information retrieval with binary classification, precision (also called positive predictive value) is the fraction of retrieved instances that are relevant, while recall (also known as sensitivity) is the fraction of relevant instances that are retrieved.

Both precision and recall are therefore based on an understanding and measure of relevance. Suppose a program for recognizing dogs in scenes from a video identifies 7 dogs in a scene containing 9 dogs and some cats. If 4 of the identifications are correct, but 3 are actually cats, the program's precision is 4/7 while its recall is 4/9.

In statistics, if the null hypothesis is that all and only the relevant items are retrieved, absence of type I and type II errors corresponds respectively to maximum precision (no false positive) and maximum recall (no false negative).

The above pattern recognition example contained 7 − 4 = 3 type I errors and 9 − 4 = 5 type II errors. Precision can be seen as a measure of exactness or quality, whereas recall is a measure of completeness or quantity.

https://en.wikipedia.org/wiki/Precision_and_recall

PR Examples

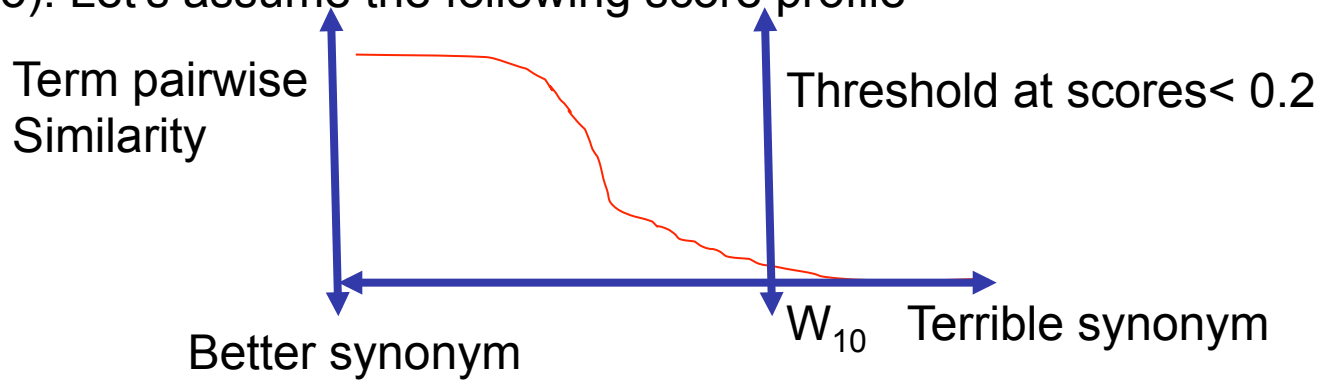
- Suppose a program for recognizing dogs in scenes from a video identifies 7 dogs in a scene containing 9 dogs and some cats. If 4 of the identifications are correct, but 3 are actually cats, the program's precision is 4/7 while its recall is 4/9.
- When a search engine returns 30 pages only 20 of which were relevant while failing to return 40 additional relevant pages, its precision is 20/30 = 2/3 while its recall is 20/60 = 1/3.

Precision/Recall Example

Ranked Synonym Results for Zebra using system A
Zebra: lion, table, cheetah, kangaroo, ……..$Word_{10,000}$

Calculate Precision and Recall
Sometimes it might be more desirable to calculate Precision@10

Assume System returns 10,000 words as candidate synonyms (most will have a score of zero or near zero). Let's assume the following score profile



Let's assume, according to wordnet, there are 20 synonyms for Zebra but only 10 occur in the words returned (above threshold), which is say 15 words.

Then precision is 10/15, while Recall is 10/20

Precision and Recall are set-based measures. So …

- **Precision and Recall are set-based measures so they can NOT discriminate very well in some scenarios**
- **E.g. Result set: Candidates ∧ True synonyms**

RESULT SET1	RESULT SET2
1st Correct Syn	1st InCorrect Syn
2nd InCorrect Syn	2nd InCorrect Syn
3rd InCorrect Syn	3rd Correct Syn
Precision@3 = 1/3	Precision@3 = 1/3

- **Introduce rank based measures such as Discounted Cumulative Gain metric (https://en.wikipedia.org/wiki/Discounted_cumulative_gain)**

Discounted Cumulative Gain metric: ranking; define DCG crudely as Rating^ (1/Ranking)
1st 5^(1/rank) #give 5* (star) for a super relevant and 1* for not relevant
2nd 1^(1/rank)
3rd 1 ^(1/rank)
DCG @ 3 for Result set 1 and result 2 will be very different

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW3, HW4, HW5, HW5
  - AWS: (Not necessary for week 6)
  - Synonym detection in Map-Reduce: Case studies + Metrics
  - Async lecture recap plus Q&A [Jimi]
  - MLE, MAP, Bayesian optimal classifier
  - MLE for Gaussian Mixture Models
  - EM for Gaussian Mixture Models
  - EM Algorithm for a mixture of Bernoulli distributions
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

Unit 6 | Distributed Supervised Machine Learning, Part 1

Hide Contents

- 6.1 Weekly Introduction (2 mins)
- 6.2 Assigned Readings
- 6.3 Supervised Machine Learning Parametric vs Non (8 mins)
- 6.4 Core Supporting Concepts Matrices Applications of Matrices (5 mins)
- 6.5 Matrices: Vector by Vector Multiplication (12 mins)
  - 6.5.1 Matrices Distributed Matrix by Vector Multiplication Part 2 (6 mins)
  - 6.5.2 Matrices: Matrix-by-Matrix Multiplication, Version 1.0 (12 mins)
  - 6.5.3 Matrices: Distributed Matrix by Vector Multiplication, Version 1.1 (6 mins)
- 6.6 Distributed Matrix Summary (3 mins)
- 6.7 Core Supporting Concepts: Optimization Theory: Gradient Descent (9 mins)
  - 6.7.1 Core Supporting Concepts: Optimization Theory: Gradient Descent Part 2 (9 mins)
- 6.8 Optimization Theory: Convex Optimization (11 mins)
- 6.9 Distributed Closed Form Linear Regression (5 mins)
  - 6.9.1 Quiz: Number of mrjobs
- 6.10 Complexity Analysis of Algorithms (3 mins)
- 6.11 Distributed-Gradient-Descent Approach to Linear Regression (7 mins)
  - 6.11.1 Quiz: LR via Distributed GD: Critique This Pseudo-Code
- 6.12 Summary of Linear Regression (3 mins)

Distributed Matrix Multiplication Summary

Distributed Matrix Multiplication is challenging!!

	Method 1.0	Method 1.1	Method 2
Number of Jobs	1	1	2
Need big memory	Yes in Reducer	N	Yes in Reducer1
Need to know dimensions of Matrix	Yes	Yes	No
Dense Emit	Yes	Yes	No

Tile-based matrix multiplication

MadLINQ: Large-Scale Distributed Matrix Computation for the Cloud, Qian et al, EuroSys 2012

Maximum vs Minimum

$$f(x)= x^3 - 12x + 1$$

First derivative
$f'(x) = 3x^2 - 12$

In convex optimisation what can we say about roots?

FOC

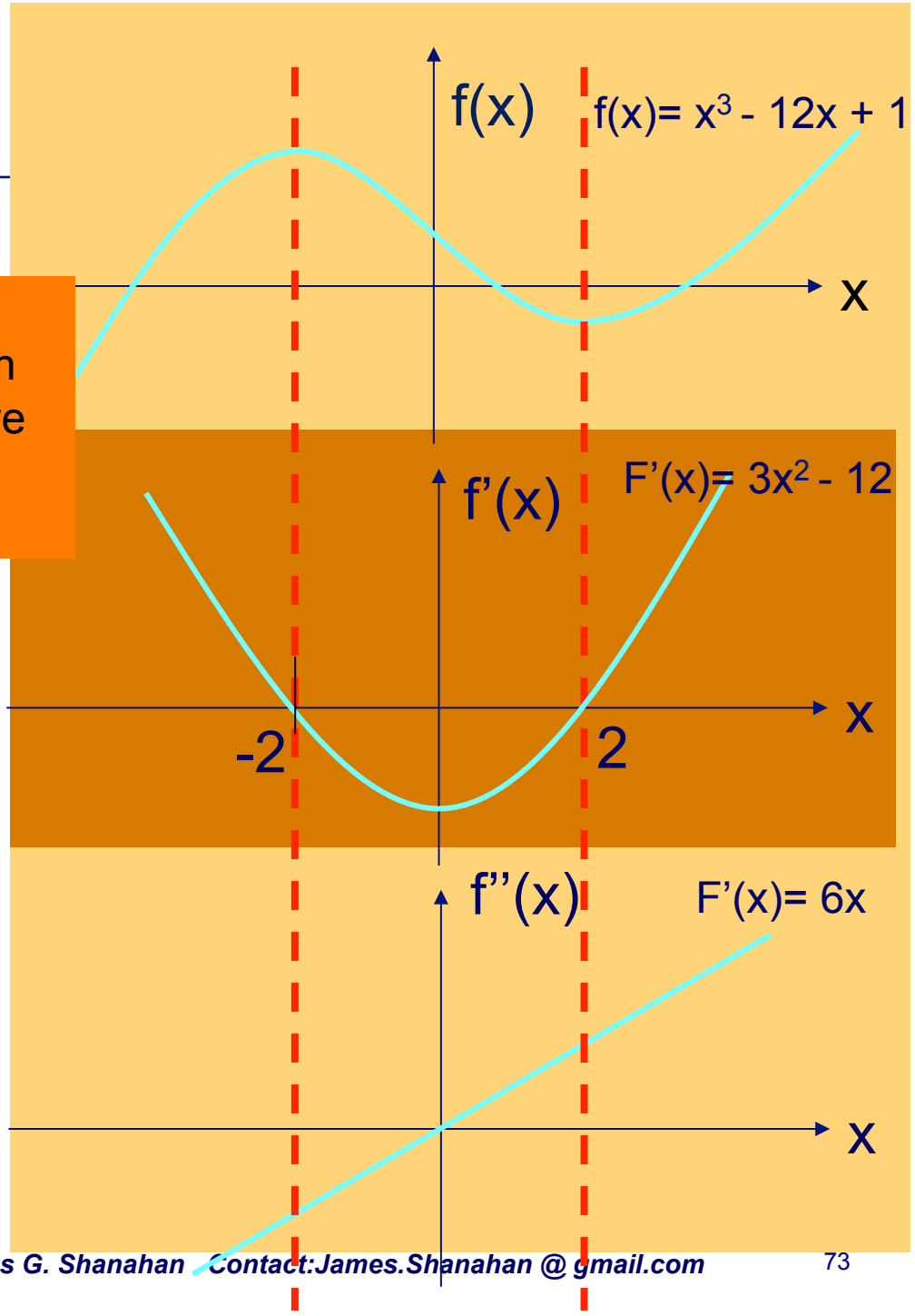
$f'(x=x^*) = 0$ at maximum and minimum
These zero points are the roots!

Second derivative $f''(x) = 6x$

Gives the "rate of change of gradient"

SOC

If $f''(x=x^*) < 0$ then maximum
If $f''(x=x^*) > 0$ then minimum
If $f''(x=x^*) == 0$ then ???



Maximum vs Minim

$f(x)= x^3 - 12$

Fir

f'(

FOC

f'(x=x*) =0 at

These zero po

Second deriva

Gives the "rate of cl

If f''(x=x*) < 0 then

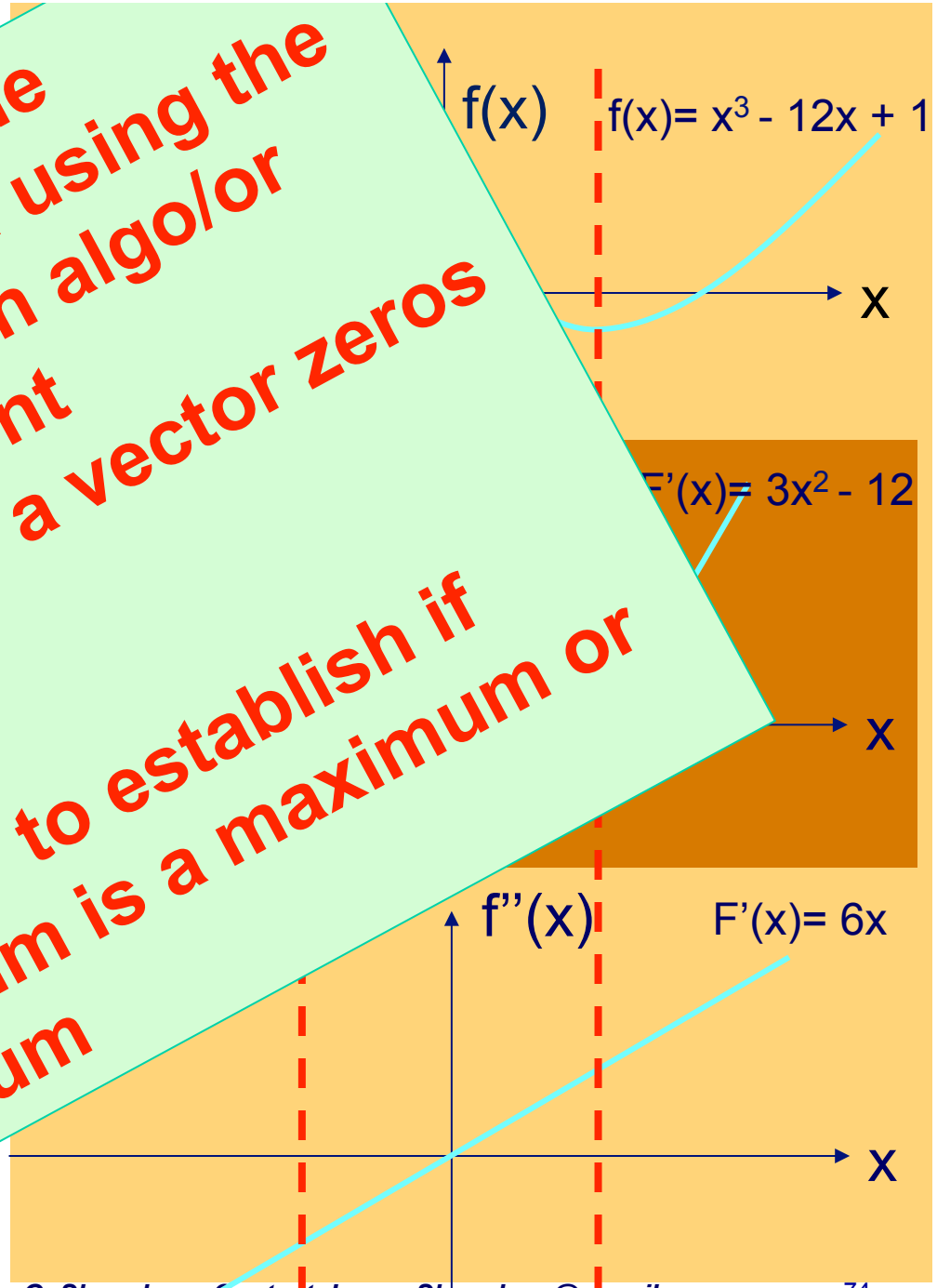
SOC If f''(x=x*) > 0 then m

If f''(x=x*) == 0 then ?

Find the roots of the Gradient function using the Newton-Rhapson algo/or gradient descent

- **FOC $\nabla(f)$= a vector zeros**

Use SOC to establish if extremum is a maximum or minimum



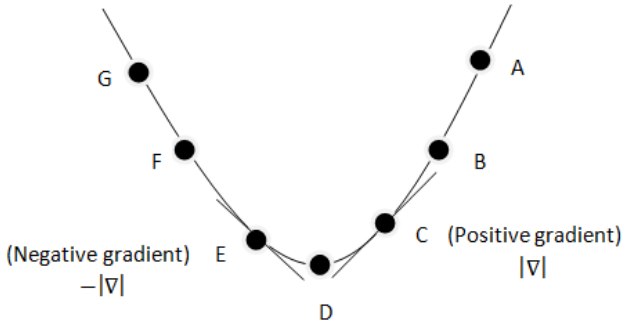
Convex Optimization Problems

More formally

Definition

An optimization problem is *convex* if its objective is a convex function, the inequality constraints f_j are convex, and the equality constraints h_j are affine

$$\begin{aligned} \underset{x}{\text{minimize}} \ & f_0(x) && \text{(Convex function)} \\ \text{s.t.} \ & f_i(x) \leq 0 && \text{(Convex sets)} \\ & h_j(x) = 0 && \text{(Affine)} \end{aligned}$$



Theorem

$\nabla f(x) = 0$ *if and only if* x *is a global minimizer of* $f(x)$.

Proof.

- $\nabla f(x) = 0$. We have

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) = f(x).$$

- $\nabla f(x) \neq 0$. There is a direction of descent.

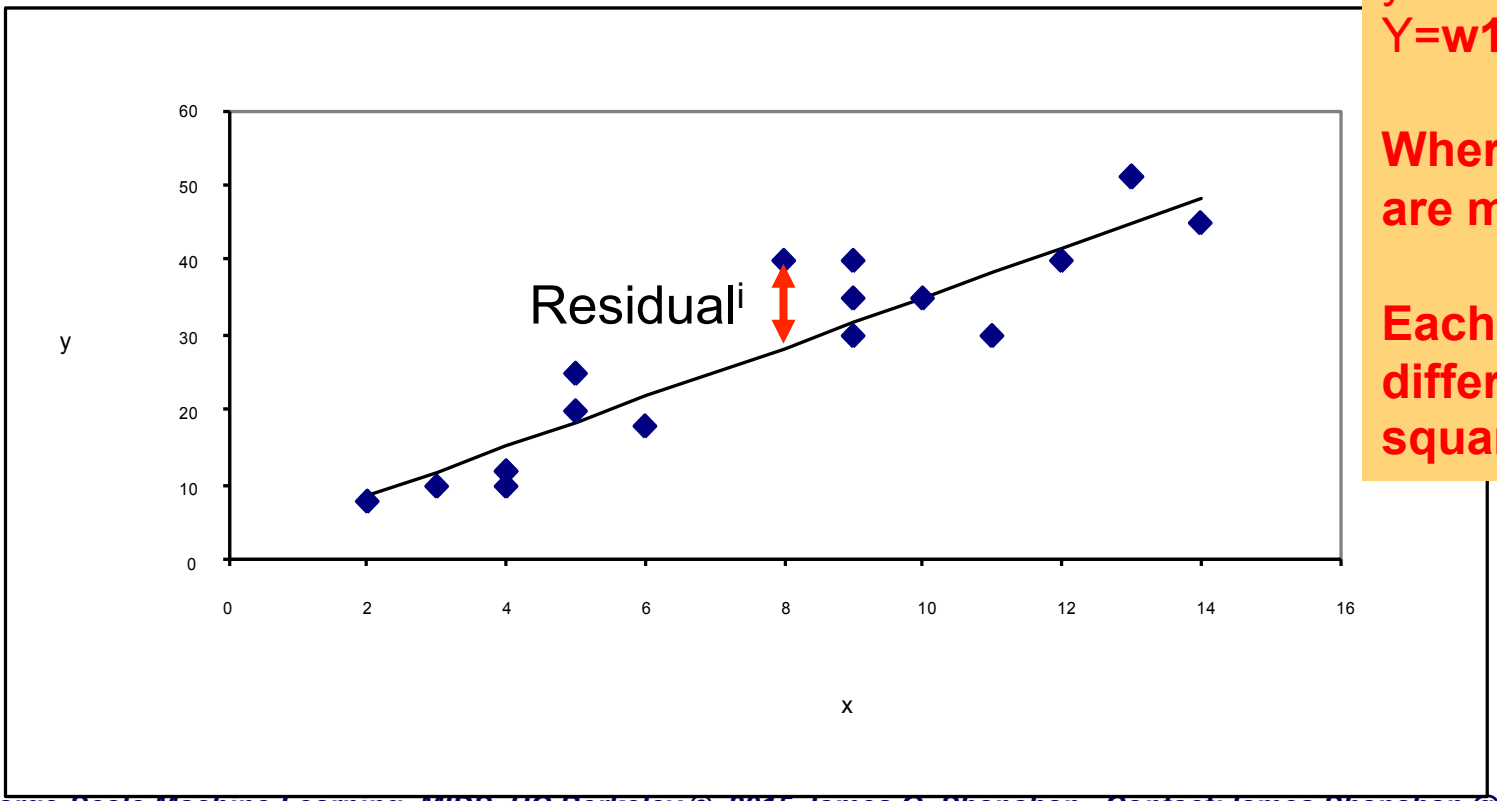
First order condition of convexity
Around a minimum, everywhere f(y)>= f(x)+NablaF(x)(y-x)
For it to be minimum if NablaF(x) is zero otherwise we violate the co

Equation of a line

$$J_q(W, X_1^m) = Minimize \sum_{i=1}^{m} (W^T X_i - y_i)^2$$

$$\text{Residual}^{\text{i}} = \left(WX^i - y^i \right)$$

$$\text{Residual}^{\text{i}} = \left(WX^i - y^i \right)^2$$



y=mx + b
Y=**w1** * x + **w0**

Where w1, w0 are model parameters

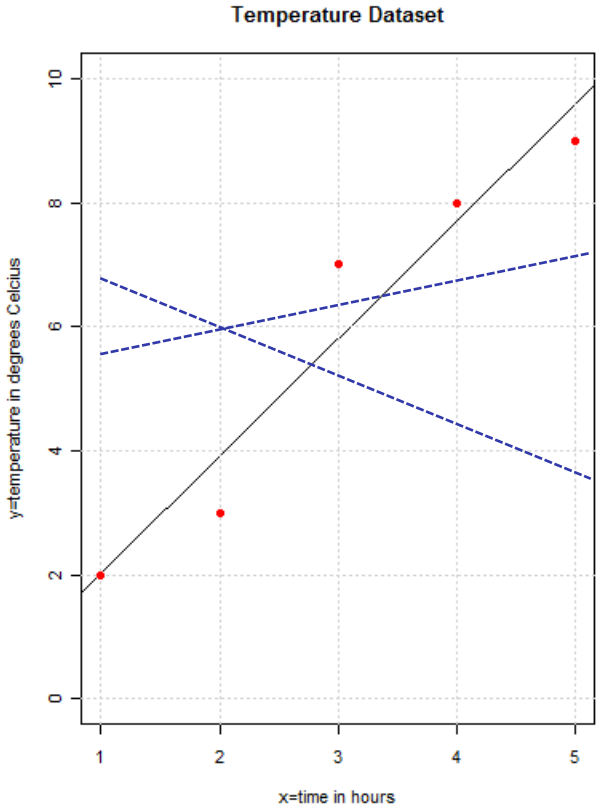
Each pair yields a different sum of squares error

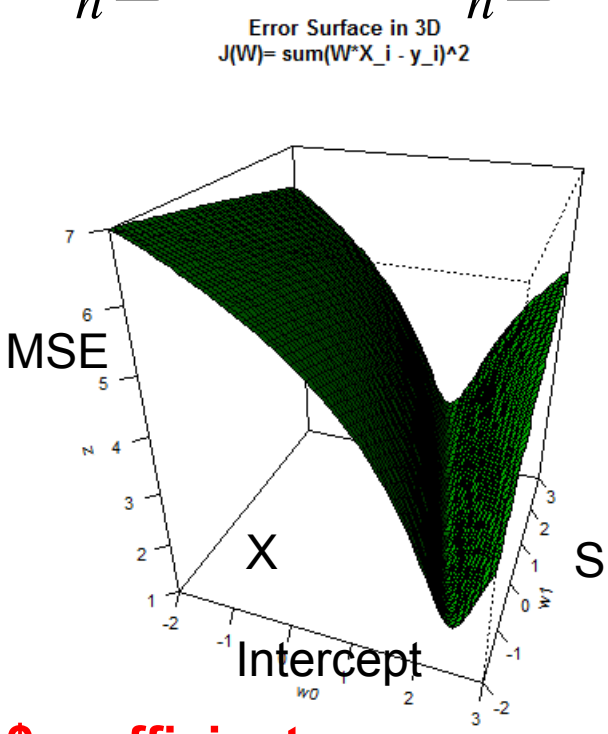
Version Space, Error surface

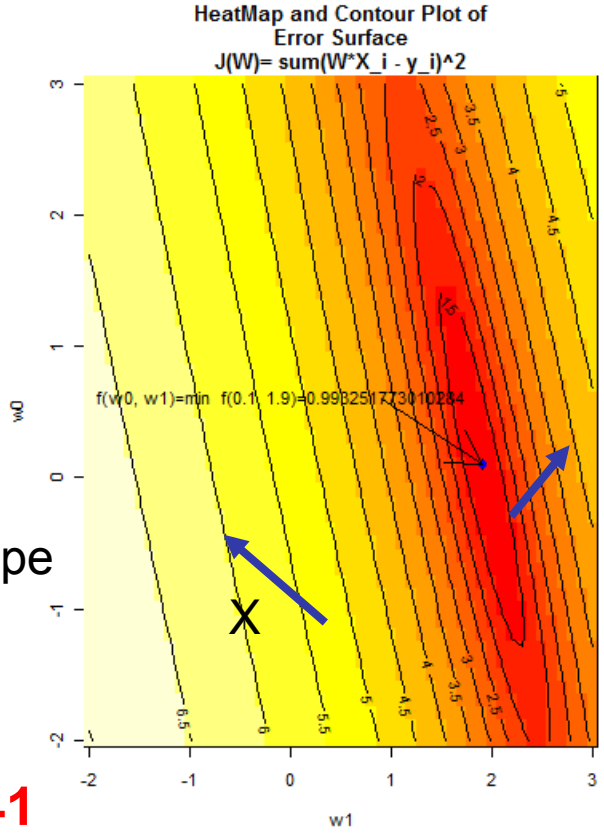
$Y=mx + b$

$Y=w_1x + w_0$

$$MSE = \frac{1}{n}\sum \text{Residual}^i = \frac{1}{n}\sum \left(WX^i - y^i\right)^2$$







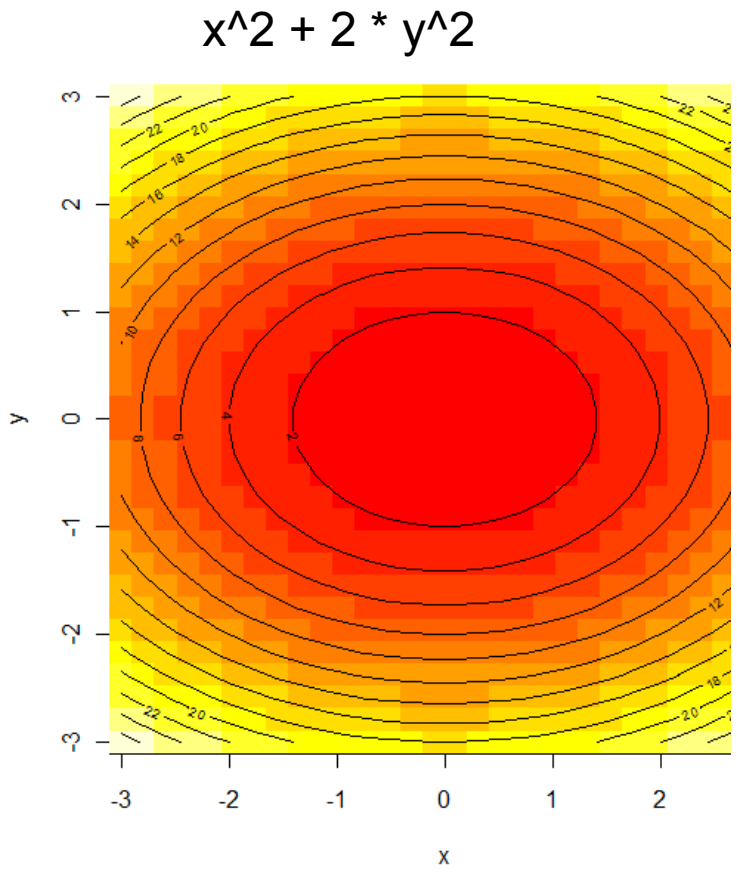
$coefficients
[1] 0.09644596, 1.90098441
$iterations
[1] 658
$SSE
[1] 2.700011

HeatMap and Contour Plot

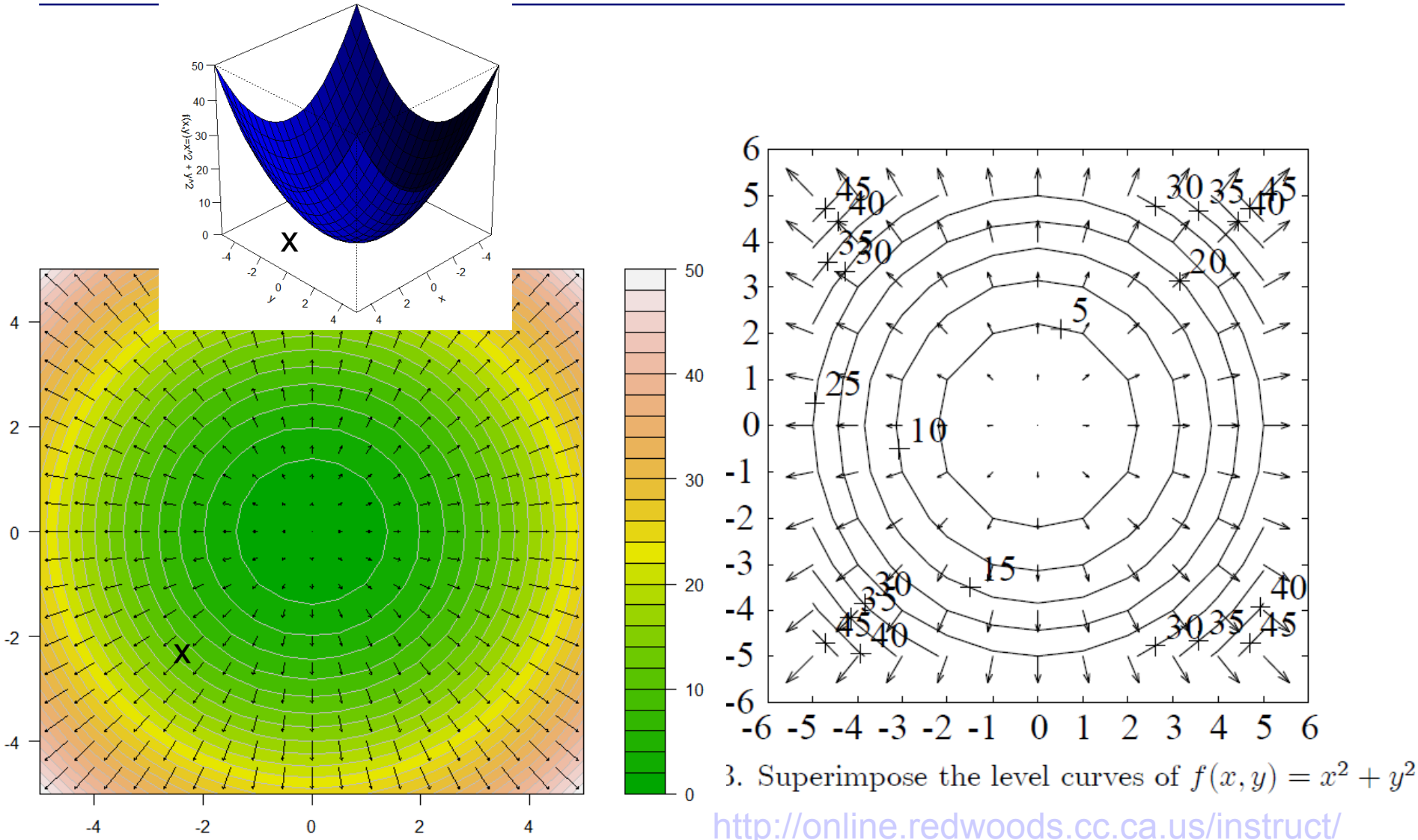
```
x <- seq(-3, 3, length= 30);    y <- x
f <- function(x,y) { x^2 + 2 * y^2 }
z <- outer(x, y, f)
```

- #new plot of isolines and heatmap
- image(x,y,z) #heat image of surface
- contour(x,y,z, add=TRUE) #add contours

See example.gradientPlots()



Gradient Vector Plots of f(x,y)=x^2+y^2

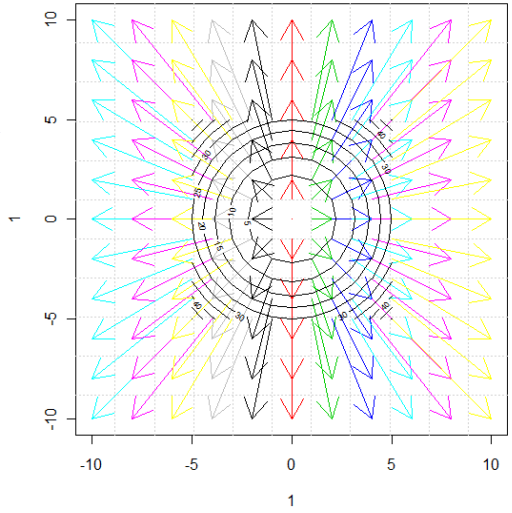


http://online.redwoods.cc.ca.us/instruct/darnold/MULTCALC/grad/grad.pdf

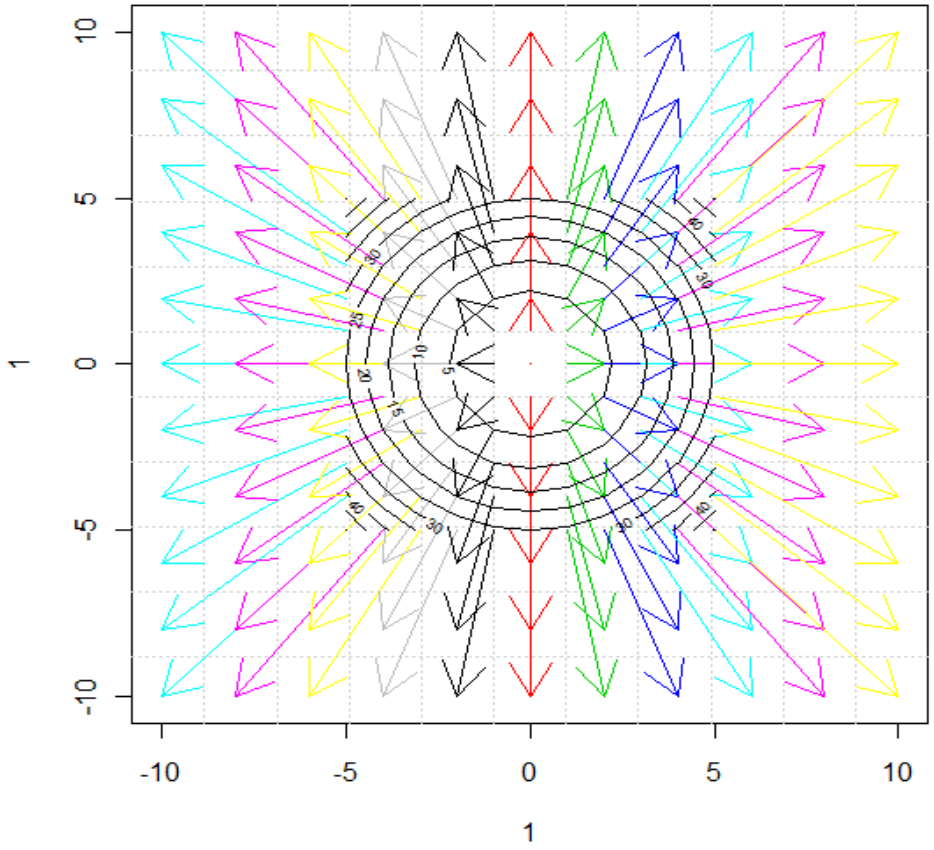
Ugly Unnormalized Gradient Plot

```
#ugly unnormalized gradient vector plot
  # plot vector plot and contour plot
  x <- -5:5
  y <- x
  f <- function(x,y) { x^2 + y^2 }
  z <- outer(x, y, f)
  u=2*x  #fprime_x(x,y) = df/dx =d(x^2 + y^2)/dx=2x
  v=2*y
  plot(1, 1, xlim=c(-10, 10), ylim=c(-10, 10), pch="")
  grid(length(x))
  for(i in x) {
    for (j in y) {
      arrows(i,j, 2*i, 2*j, col=i+10)  #(f'x(x,y), f'y(x,y), col=colour
    }
  }
  contour(x,y,z, add=TRUE) #add contours
```

In example.gradientPlots()

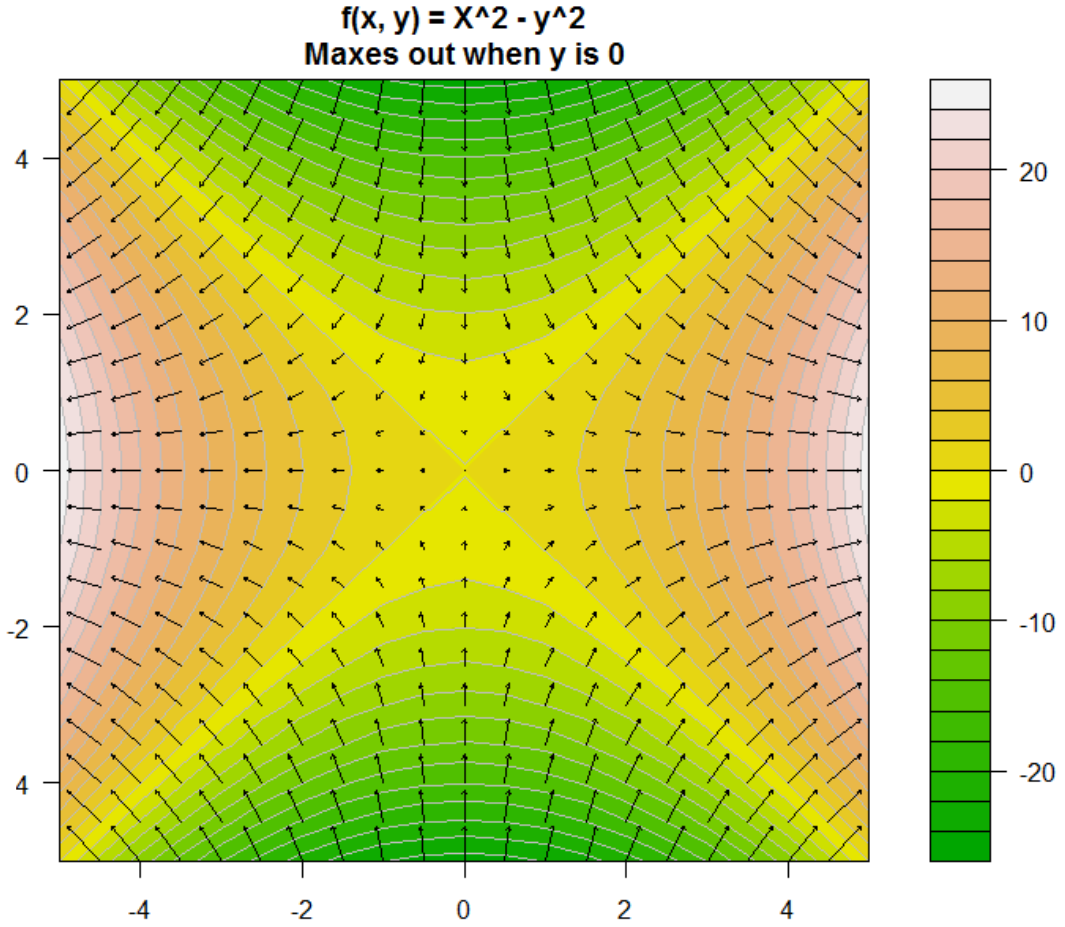


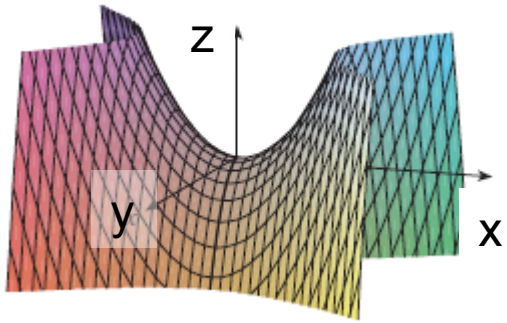
Ugly Unnormalized Gradient Plot



Gradient Vector Field for $f(x,y) = x^2 - y^2$

- **Sampled gradient vector plot superimposed on a function heat map of f(x, y) = $x^2 - y^2$**
- **Each gradient vector is plotted starting at the point**
- ***As expected, the gradient vectors point "uphill" and are perpendicular* to the level curves.**

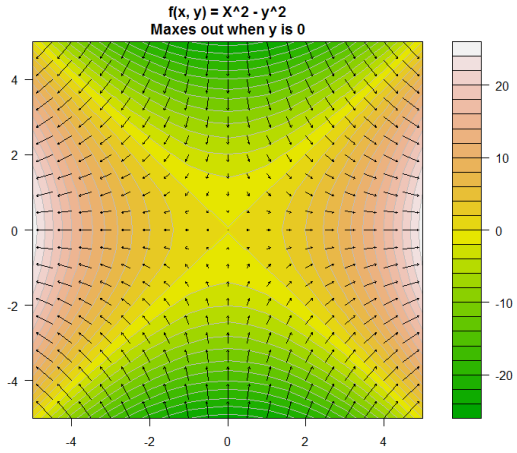




Prettified Gradient Plot

(looks like a quiver!)

```
#prettified gradient vector plot using quiver()
#f <- expression( (3*x^2 + y) * exp(-x^2-y^2))
f <- expression( (x^2) - (y^2))
#f <-expression((x^2+x^2))
x <- y <- seq(-5, 5, by=0.5)
par(mar=c(3,3,3,3))
quiver2(f,x,y, color.palette=terrain.colors,
  main="f(x, y) = X^2 - y^2\nMaxes out when y is 0")
```

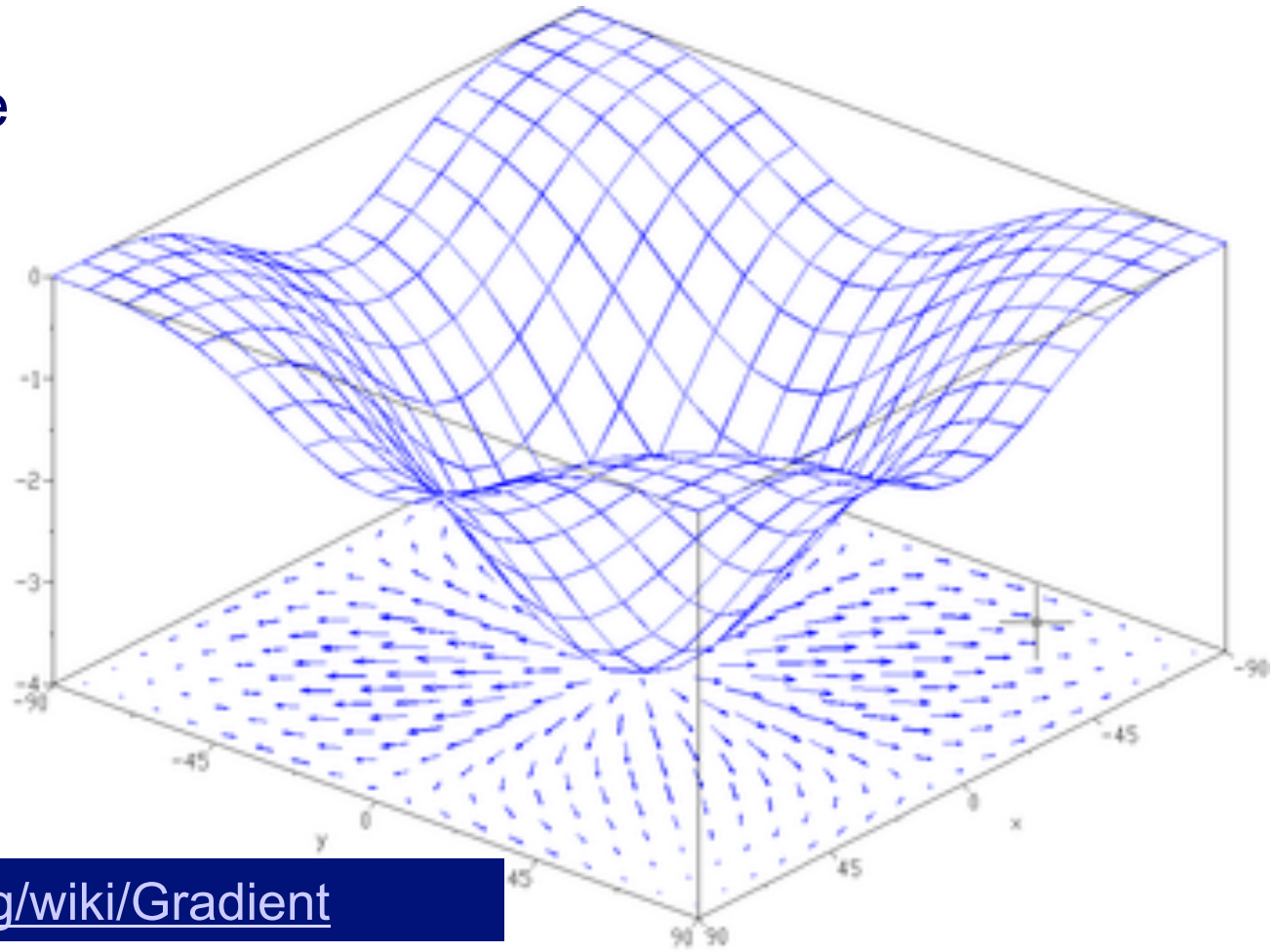


Gradient Vector at Extrema is <0, 0, ..>

- The gradient is a fancy word for derivative, or the rate of change of a function.
- It's a vector (a direction to move) that points in the direction of greatest increase of a function is zero at a local maximum or local minimum (because there is no single direction of increase); the magnitude of the vector is zero. Gradient at turning points =<0, 0, 0…,0)
- The term gradient typically refers to the derivative of vector functions, or functions of more than one variable. Yes, you can say a line has a gradient (its slope), but using the term gradient for single-variable functions is unnecessarily confusing. Keep it simple.
- http://betterexplained.com/articles/vector-calculus-understanding-the-gradient/

Gradient Example

- The gradient of the function $f(x,y) = -(\cos^2 x + \cos^2 y)^2$ depicted as a vector field on the bottom plane



http://en.wikipedia.org/wiki/Gradient

Gradient Descent for LR Price~Size Iteration 0

Eqn Line y= aX + b

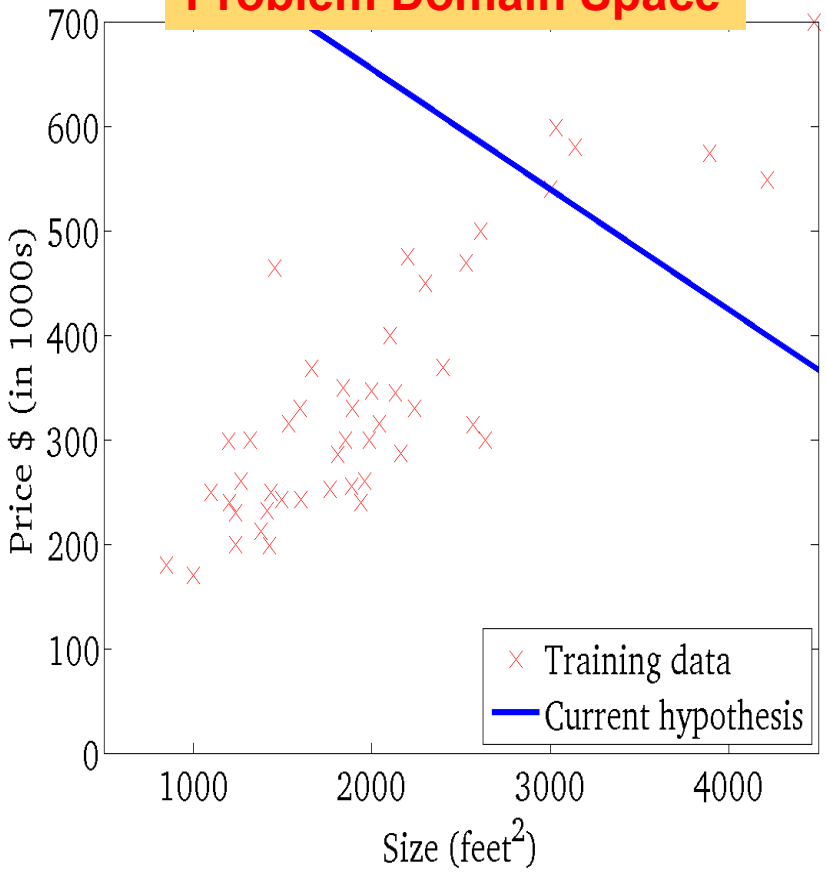
$Y = w_1 X + W_0$

$J(\theta_0, \theta_1)$ $J_q(W, X_1^L) = \frac{1}{m}\sum_{i=1}^{m}(W^T X_i - y_i)^2$

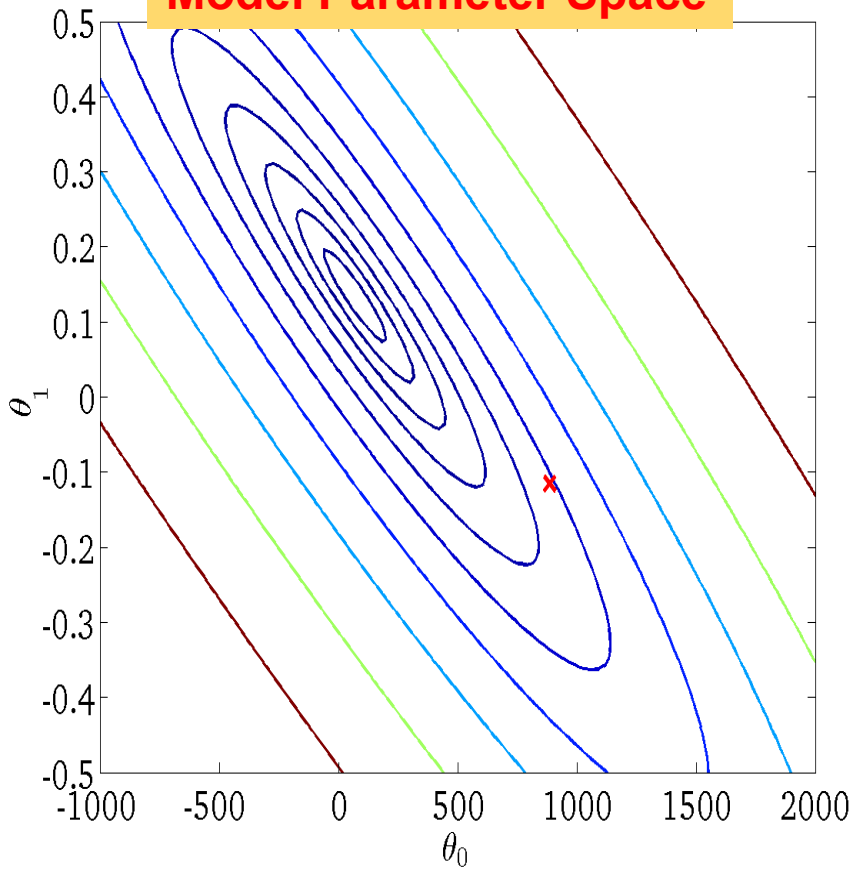
(for fixed θ_0, θ_1 this is a function of x)

(function of the parameters θ_0, θ_1)

Problem Domain Space



Model Parameter Space



Closed form solution to OLS

- **Convex optimization tells us that the first order condition**
  - $\nabla F(X) = 0$ at the global minimum and we are going to exploit that here to get a closed form solution for linear regression

Closed form solution to OLS

SSE $\mathbf{S(\theta) = \Sigma_i [y(i) - \Sigma_j \theta_j x_{ij}]^2}$

y = N x 1 vector of target values

(p+1) x 1 vector of parameter values

N x (p+1) vector of input values

$= \Sigma_i e_i^2$

$= \underline{e}' \underline{e}$

$= (y - X\theta)' (y - X\theta)$

where $\underline{e} = y - X\theta$

$S(\theta) = \Sigma e^2 = e' e = (y - X\theta)' (y - X\theta)$

$= y' y - \theta' X' y - y' X\theta + \theta' X' X\theta$

Minimize objective function and find root vector of the gradient function

$= y' y - 2\theta' X' y + \theta' X' X\theta$

Taking derivative of $S(\theta)$ with respect to the components of θ gives….

$\mathbf{dS/d\theta = -2 X' y + 2 X' X\theta}$

Set this to 0 to find the extremum (minimum) of S as a function of θ …

Normal Equations and solving for θ

Set to 0 to find the extremum (minimum) of S as a function of θ …

$\Rightarrow$ $-2\,X'\,y + 2\,X'\,X\,\theta = 0$

$\Rightarrow$ $\mathbf{X'\,X\,\theta = X'\,y}$ **(known in statistics as the Normal Equations)**

Letting $X'\,X = C$, and $X'\,y = b$,
we have $C\,\theta = b$, i.e., a set of linear equations

We could solve this directly, e.g., by matrix inversion

$$\theta = C^{-1}\,b = (X'\,X)^{-1}\,X'\,y$$

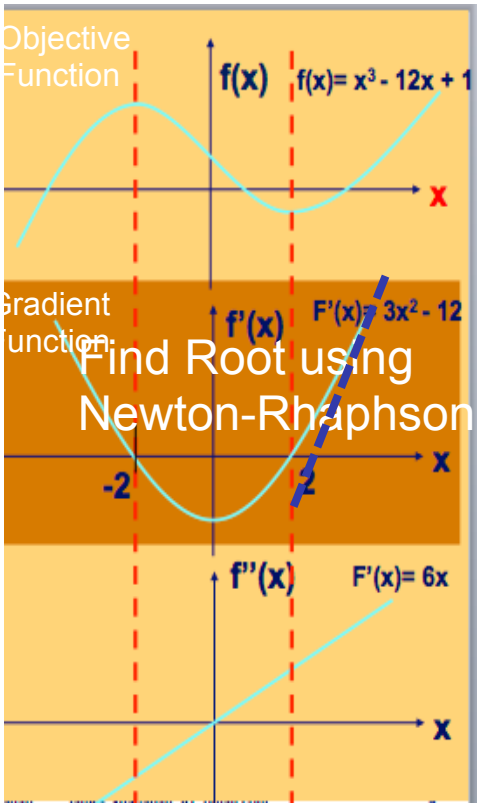
Closed form solution to OLS

Gradient Descent (a simpler root finder)

Given: Minimize f(x)

STEP 1: Find the zeros of the gradient function f'(x)

- Using Bisection method; OR Newton-Raphson; OR Gradient Descent



$$x^{i+1} = x^i - \left[f''(x^i)\right]^{-1} f'(x^i) \qquad \text{Univariate case}$$

$$x^{i+1} = x^i - \left[H(x^i)\right]^{-1} J(x^i) \qquad \text{Multivariate Case}$$

Newton-Raphson

Calculating f''(x), the Hessian H in multivariate case, and inverting it is complex so simpler algorithms have been developed such as gradient descent

$$x^{i+1} = x^i - \mathrm{a}^i f'(x^i) \qquad \text{Univariate case}$$

$$x^{i+1} = x^i - \mathrm{a}^i J(x^i) \qquad \text{Multivariate Case}$$

Gradient Descent

How large should I step in the positive gradient direction (gradient ascent)

- or in the negative gradient direction (gradient descent)

- **Convergence criteria. E.g., decision vector X does not change that much**

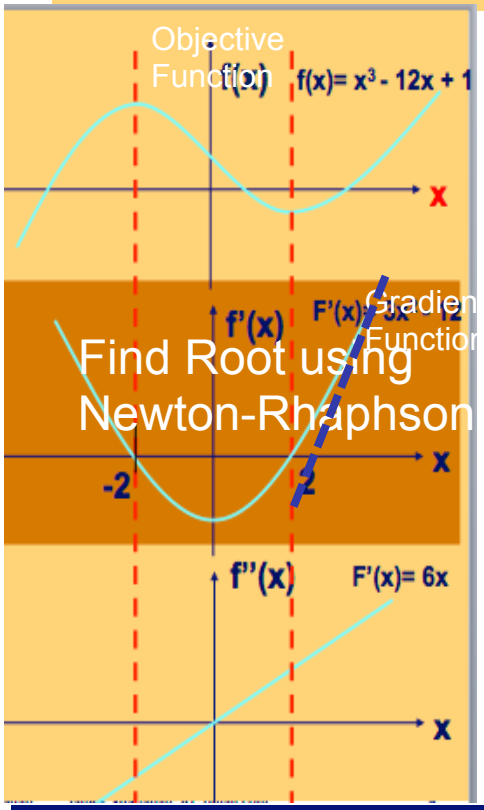
Linear Regression via Gradient Descent (a simpler root finder)

$$J_q(W, X_1^m) = Minimize \sum_{i=1}^{m} (W^T X_i - y_i)^2$$

Given: Minimize f(x)

STEP 1: Find the zeros of the gradient function f'(x)

- Using Bisection method; OR Newton-Raphson; OR Gradient Descent



$$RSS = Variance \text{ of } \varepsilon$$

$$0 = \frac{\partial \sum \hat{\varepsilon}_i^2}{\partial W} = \frac{\partial \left(\sum_{j=1}^{n} \left(X_j W - y_i \right)^2 \right)}{\partial W}$$

Gradient vector of partial derivatives

$$\nabla J(W) = \left(\sum_{j=1}^{n} \left(X_j W - y_i \right) X_j \right)$$

Pull model closer to examples with biggest residual

$$W^{t+1} = W^t - \alpha^i \left(\sum_{j=1}^{n} \left(X_j W^t - y_i \right) X_j \right)$$

Gradient Descent

For another derivation see:
http://www.stanford.edu/class/cs229/notes/cs229-notes1.pdf

OLS via Distributed Gradient Descent

- **Master/Driver Process**
- **Initialize model parameters, W= vector of zeros**

$$W = (0,0,....)$$

- **While not converged**
  - Broadcast model (e.g., weight vector) to the worker nodes
  - Mapper (MANY mappers)
    - Compute partial gradient for each training example
    - Combine in memory $\nabla f = \sum (y^i - W_i X^i) X^i$
    - Finally Yield the partial gradient
  - Reducer (single Reducer)
    - Aggregate partial gradients
    - Yield full gradient $\nabla f = \sum (y^i - W_i X^i) X^i$
  - Update weight vector $W_{i+1} = W_i + \alpha * \nabla f$
  - Check for convergence
- **End-While**

$$RSS = Variance \text{ of } \varepsilon$$

$$0 = \frac{\partial \sum \hat{\varepsilon}_i^2}{\partial W} = \frac{\partial \left(\sum_{j=1}^{n} \left(X_j W - y_i \right)^2 \right)}{\partial W}$$

$$\nabla J(W) = \left(\sum_{j=1}^{n} \left(X_j W - y_i \right) X_j \right)$$

$$W^{t+1} = W^t - \alpha^i \left(\sum_{j=1}^{n} \left(X_j W^t - y_i \right) X_j \right)$$

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW3, HW4, HW5, HW5
  - AWS: (Not necessary for week 6)
  - Synonym detection in Map-Reduce: Case studies + Metrics
  - Async lecture recap plus Q&A [Jimi]
  - MLE, MAP, Bayesian optimal classifier
  - MLE for Gaussian Mixture Models
  - EM for Gaussian Mixture Models
  - EM Algorithm for a mixture of Bernoulli distributions
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

EM Algorithm for flat clustering: Gaussian Mixture Models and Mixtures of Bernoulli Models

James G. Shanahan[1]

[1]***NativeX and iSchool, UC Berkeley, CA***

EMAIL: James_DOT_Shanahan_AT_gmail_DOT_com

- **K||**

- http://www.ojs.academypublisher.com/index.php/jetwi/article/viewFile/jetwi04015159/4277

II. Related Works

Clustering is a classic problem in machine learning and computational geometry. In the popular k-means formulation, one is given an integer k and a set of n data points $\mathbf{X} \subset \mathbf{R}^m$. k is the number of cluster centers. The goal is to choose k centers $\mathcal{C}$ to minimize the sum of the squared distances between each point and its closest center.

$$\phi = \sum_{\mathbf{x} \in \mathbf{X}} \min_{\mathbf{c} \in \mathcal{C}} \|\mathbf{x} - \mathbf{c}\|^2 \quad (1)$$

Solving this problem is NP-hard, even with just two clusters[16], however Lloyd[17] proposed a local search solution 25 years ago that is still widely used today.

In this section, we formally define the k-means clustering method, the KKZ clustering method and the k-means++ clustering method.

A. k-means clustering method

The k-means clustering method is simple and fast and locally improves the centers of mass of clusters. It works as follows.

1) Arbitrarily choose k initial centers $\mathcal{C} = \mathbf{c}_1, \cdots, \mathbf{c}_k$,
2) For each $i \in \{1, \ldots, k\}$, set the cluster $\mathbf{c}_i$ to be the set of points in $\mathbf{X}$ that are closer to $\mathbf{c}_i$ than they are to $\mathbf{c}_j$ for all $j \neq i$.
3) For each $i \in \{1, \ldots, k\}$, set $\mathbf{c}_i$ to be the center of the mass of all the points in a set C_i of cluster i: $\mathbf{c}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$.
4) Repeat steps 2) and 3) until $\mathbf{c}_i$ no longer changes.

It is standard practice to uniformly choose the initial centers at random from $\mathbf{X}$. For Step 2), the ties may be arbitrarily broken, as long as the method is consistent. Steps 2) and 3) are both guaranteed to decrease ϕ; therefore, the method makes local improvements to an arbitrary cluster until it is no longer possible to do so.

The k-means method is attractive in practice because it is simple and generally fast. Unfortunately, it is guaranteed only to find a local optimum, which can often be quite poor.

B. KKZ clustering method

The KKZ method was proposed by Katsavounidis et al. [18]. This method calculates the entire distance among the data and finds the data with a wide distance. The data are selected as the initial cluster centers. At any given time, let $D(\mathbf{x})$ denote the shortest distance from a data point $\mathbf{x}$ to the closest center we have already chosen. Then, the following clustering method is defined as the KKZ clustering method[18].

1a) Choose initial centers $\mathbf{c}_1$ and $\mathbf{c}_2$. The distance between $\mathbf{c}_1$ and $\mathbf{c}_2$ is the widest of all distance between a data point and the other data point (Figure 2).
1b) For all data, $D(\mathbf{x}_j), j \in \{1, \cdots, n\}$ are calculated (Figure 3).

Figure 2. Initial centers of KKZ method

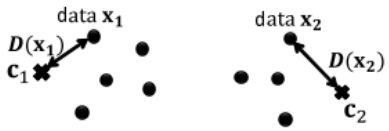


Figure 3. Distance $D(\mathbf{x})$

1c) Choose the next center $\mathbf{c}_i$, selecting $\mathbf{c}_i = \mathbf{x}' \in \mathbf{X}$ with the widest distance $D(\mathbf{x}')$ (Figure 4).
1d) Repeat step 1b) until we have chosen a total of k centers.

Steps 2)-4) proceed just like that for the standard k-means algorithm.

The KKZ method is attractive in practice because it is simple for decision of unique initial centers. However, the KKZ method sometimes find bad clusters because unfortunately it depends on outlier data points.

C. k-means++ clustering method

The k-means method begins with an arbitrary set of cluster centers. k-means++ clustering proposes specifically choosing these centers. At any given time, let $D(\mathbf{x})$ denote the shortest distance from a data point $\mathbf{x}$ to the closest center we have already chosen. Then, the following clustering method is defined as the k-means++ clustering method[19].

1a) Choose an initial center $\mathbf{c}_1$ uniformly at random from $\mathbf{X}$.
1b) For all data, $D(\mathbf{x}_j); j \in \{1, \cdots, n\}$ are calculated (Figure 3).
1c) Randomly generate a real value L satisfying the following equation.

$$0 < L \leq \sum_{\mathbf{x} \in \mathbf{X}} D(\mathbf{x}_j) \quad (2)$$

1d) Choose the next center $\mathbf{c}_i$, selecting the $\mathbf{c}_i = \mathbf{x}_j$ with satisfying the following equation (Figure 5).

$$\sum_{m=1}^{j-1} D(\mathbf{x}_m) < L \leq \sum_{m=1}^{j} D(\mathbf{x}_m) \quad (3)$$

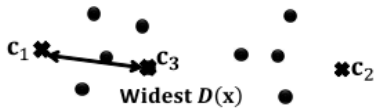


Figure 4. Next center $\mathbf{c}_i$ of KKZ method

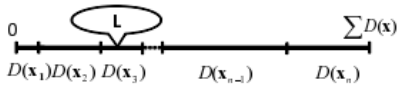


Figure 5. Next center $\mathbf{c}_i$ of k-means++ method

1e) Repeat step 1b) until we have chosen a total of k centers.

Steps 2)-4) proceed in the same way as with the standard k-means clustering method. We call the weighting used in Step 1b) simply "D^2 weighting".

III. Proposed Method

This section describes a problem with the k-means and the k-means++ clustering methods. Then, we propose a k-means combined with an Independent Component Analysis (ICA)[20], [21], [22] based seeding method.

A. Problem for k-means and k-means++ clustering methods

We have six data points, which consist of $\mathbf{x}_i$ (i =1, …, 6) and these points are divided into two clusters. Figure 6 shows these six data points.

In addition, Figure 7 shows the global optimal clustering result for these six data points. The first cluster consists of $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4, \mathbf{x}_5\}$ and the other consists of $\{\mathbf{x}_3, \mathbf{x}_6\}$. We assume that most of clustering methods can find the global optimal clusters. However, the k-means clustering method generates bad clusters if $\mathbf{x}_2$ and $\mathbf{x}_5$ are chosen as the initial $\mathbf{c}_1$ and $\mathbf{c}_2$ cluster centers. Figure 8 shows the local optimal clusters, which are bad clusters. The k-means++ clustering method was developed to avoid this bad clustering.

However, the k-means++ clustering method sometimes generates bad clusters because it depends on the choice of the initial center $\mathbf{c}_1$. The initial center $\mathbf{c}_1$ is chosen uniformly at random from $\mathbf{X}$.

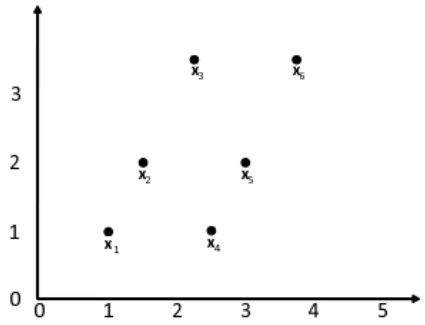


Figure 6. Given Data

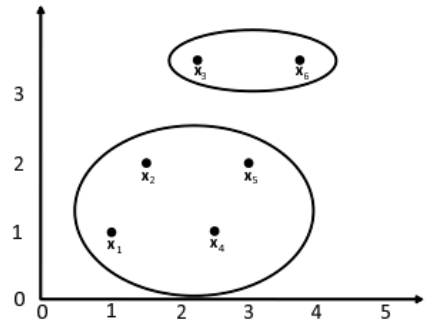


Figure 7. Global Optimal Clustering Case

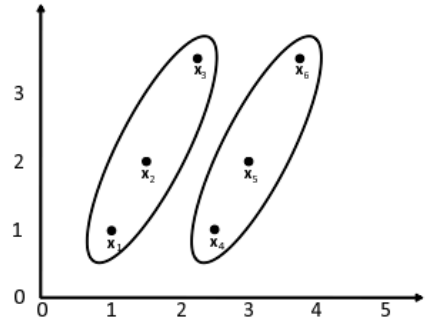


Figure 8. Local Optimal Clustering Case

B. k-means combined with ICA based seeding method

The k-means clustering method begins with an arbitrary set of cluster centers. The k-means++ clustering method begins with a small arbitrary set of cluster centers. As stated above, we propose a method for specifically choosing these centers. At any given time, we can obtain independent components (ICs) from given data $\mathbf{X}$. Then, we define the following seeding method.

1a) Extract k independent components $\mathbf{IC}_1, \cdots, \mathbf{IC}_k$ from given data $\mathbf{X}$ (Figure 9).
1b) Choose k initial centers $\mathbf{c}_i$ ($i = 1, \cdots, k$), selecting $\mathbf{c}_i = \mathbf{x}' \in \mathbf{X}$ with a minimum $\frac{\mathbf{IC}_i \cdot \mathbf{x}'}{\|\mathbf{IC}_i\| \|\mathbf{x}'\|}$ (Figure 10).

Steps 2)-4) proceed in the same way as with the standard k-means clustering method. Figure 11 shows the concept of the k-means clustering method combined with the ICA based seeding method. In the figure 11, $\mathbf{IC}_1$ and $\mathbf{IC}_2$ denote independent components. The each independent component may become an initial seed to generate the global optimal clustering case.

Live Session Outline

- **Housekeeping**
  - Please mute you microphones
  - Start RECORDING (bonus points for reminding me!)
- **Week**
  - Homework HW3, HW4, HW5, HW5
  - AWS: (Not necessary for week 6)
  - Synonym detection in Map-Reduce: Case studies + Metrics
  - Async lecture recap plus Q&A [Jimi]
  - MLE, MAP, Bayesian optimal classifier
  - MLE for Gaussian Mixture Models
  - EM for Gaussian Mixture Models
  - EM Algorithm for a mixture of Bernoulli distributions
- **Wrapup**
  - Finish RECORDING (bonus points for reminding me!)
  - Click End Meeting

EM Algorithm for flat clustering: References

- **See Section 16.5 Model-based clustering in**
  - http://nlp.stanford.edu/IR-book/pdf/16flat.pdf
  - For K-Means see 16.4 K-means
  - http://cs229.stanford.edu/notes/cs229-notes2.pdf
- **Other references**
  - What is the expectation maximization algorithm?,
    - http://ai.stanford.edu/~chuongdo/papers/em_tutorial.pdf
  - Harvard notes on EM,
  - stackoverflow on EM
- **Advanced**
  - Mixtures of Gaussians and the EM algorithm
    - http://cs229.stanford.edu/notes/cs229-notes7b.pdf
  - The EM Algorithm
    - http://cs229.stanford.edu/notes/cs229-notes8.pdf

Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a mixture of Bernoulli distributions**

MLE, MAP and Bayes Optimal Classifier

Meta concepts: about our model space

- **In the following we will introduce the general form of the Maximum Likelihood Estimation (MLE) which is a special case of the so called Maximum a Posteriori (MAP) estimation.**
- **MAP in turn is an approximation of the Bayesian prediction (Bayes Optimal Classifier).**
- **So we will first briefly explain Bayesian prediction, go on with MAP and finally end with MLE.**
  - The derivation of MLE follows Russell and Norvig's AI Book [6] (Chap. 20, pp. 712-715).

MAP Model

Models: linear regression models: θ: slope, intercept

9.6.2 Maximum a Posteriori

The *Maximum a Posteriori (MAP)* prediction is based only on one hypothesis, not on the sum of all hypotheses as in Bayesian learning. The hypothesis h_{MAP} chosen for prediction is the most probable one, given the data.

Definition 9.32 (MAP Hypothesis)

Likelihood X Prior

$$h_{MAP} = \arg\max_i P(h_i|d) \quad (9.20)$$

$$= \arg\max_i \alpha P(d|h_i) P(h_i) \quad (9.21)$$

$$= \arg\max_i P(d|h_i) P(h_i) \quad (9.22)$$

This way the original task of a large (infinite) summation for the Bayesian Learning is replaced by the task of optimization for MAP. MAP is already a reasonable method for a estimation task. It estimates the hypothesis h_{MAP}. But we will see that in most practical cases we can even make a further simplification of the estimation process. This will lead us to the Maximum Likelihood Estimation (MLE).

Maximum Likelihood Model

9.6.3 Maximum Likelihood

A further simplification of the MAP can be made by the assumption that we have the same prior probability $P(h_i)$ for all hypotheses. In applications where a hypothesis is represented by a set of parameters for a given model, we do not have a reason to prefer one single set of parameters. Thus the upper assumption is valid. The formula for the computation of the most probable hypothesis now becomes even more simple.

Definition 9.33 (ML Hypothesis)

Likelihood X Prior

$$h_{ML} = \arg\max_i P(h_i|d) \quad (9.23)$$

$$= \arg\max_i \alpha P(d|h_i)\, P(h_i) \quad (9.24)$$

$$= \arg\max_i P(d|h_i)\, P(h_i) \quad (9.25)$$

$$= \arg\max_i P(d|h_i) \quad (9.26)$$

Likelihood

This formula tells us that we can compute the optimal hypothesis h_{ML} which is the most probable one given the data d by maximizing just $P(d|h_i)$. The fact that the maximized quantity $P(d|h_i)$ is the likelihood of the data should explain the term *Maximum Likelihood*.

MAP versus MLE

9.6.2 Maximum a Posteriori

The *Maximum a Posteriori (MAP)* prediction is based only on one hypothesis, not on the sum of all hypotheses as in Bayesian learning. The hypothesis h_{MAP} chosen for prediction is the most probable one, given the data.

Definition 9.32 (MAP Hypothesis)

$$h_{MAP} = \arg\max_i P(h_i|d) \tag{9.20}$$

$$= \arg\max_i \alpha\, P(d|h_i)\, P(h_i) \tag{9.21}$$

$$= \arg\max_i P(d|h_i)\, P(h_i) \tag{9.22}$$

This way the original task of a large (infinite) summation for the Bayesian Learning is replaced by the task of optimization for MAP. MAP is already a reasonable method for a estimation task. It estimates the hypothesis h_{MAP}. But we will see that in most practical cases we can even make a further simplification of the estimation process. This will lead us to the Maximum Likelihood Estimation (MLE).

9.6.3 Maximum Likelihood

A further simplification of the MAP can be made by the assumption that we have the same prior probability $P(h_i)$ for all hypotheses. In applications where a hypothesis is represented by a set of parameters for a given model, we do not have a reason to prefer one single set of parameters. Thus the upper assumption is valid. The formula for the computation of the most probable hypothesis now becomes even more simple.

Example: MAP hypothesis

- **In classification problems what we really want to compute is the probability of a class given an example.**
- **Assume we have three hypotheses h1, h2, and h3 (only three possible models in this example;**
  - in general we will have an infinite number of models/hypotheses)
  - with the following posterior probabilities (these posterior probabilities are calculated via the likelihood of the data as presented above):
-
- **P(h1|D) = 0.4, P(h2|D) = 0.3, and P(h3|D) = 0.3.**
-
- **Here *h1* is the MAP hypothesis.**
-

Example: Bayes optimal classification rule

Bayes optimal classification is s Weighted ensemble

- **P(h1|D) = 0.4, P(h2|D) = 0.3, and P(h3|D) = 0.3.**
- **Next, assume a new example, *X1*, is presented and suppose that the individual classifiers classify the example as follows (i.e., each classifier/hypothesis/model gets to vote in a binary fashion):**
  - P(h1) = + (i.e., $Pr_{h1}(+|X1)=1$ and $Pr_{h1}(-|X1)=0$),
  - P(h2) = -
  - and P(h3) = -.
- **How can we take all hypotheses/votes into account? We can take a weighted combination based on posterior probabilities. This is known as the Bayes optimal classification rule.**
-

Example: Bayes optimal classification rule

- **Given:**
  - P(h1|D) = 0.4, P(h2|D) = 0.3, and P(h3|D) = 0.3.
  - Inferred: P(h1) = +; P(h2) = - and P(h3) = -.
- **How can we take all hypotheses/votes into account? We can take a weighted combination based on posterior probabilities. This is known as the Bayes optimal classification rule.**
-
- **Let *c* be a possible class, then the Bayes optimal probability is calculated as follows:**
-
- $P(c \mid D) = \Sigma_i \; P(c \mid h_i) \; P (h_i \mid D)$
  - i.e., a weighted sum over all possible hypothesis h_i.
- **And we want to choose the *class* that maximizes this probability:**
-
- $\text{argmax}_c \; \Sigma_i \; P(c \mid h_i) \; P (h_i \mid D)$
-

Example: Bayes optimal class is -

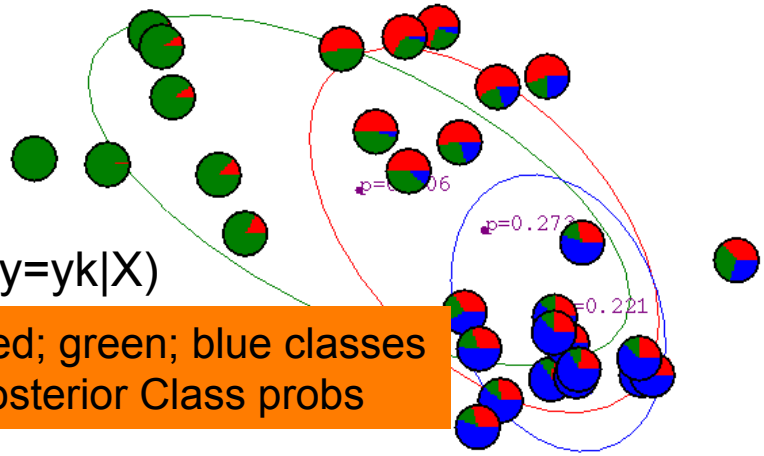
- **Given:**
  - P(h1|D) = 0.4, P(h2|D) = 0.3, and P(h3|D) = 0.3.
  - Inferred: P(h1) = +; P(h2) = - and P(h3) = -.
- **Let *c* be a possible class, then the Bayes optimal probability is calculated as follows:**
  - $P(c \mid D) = \Sigma_i \; P(c \mid h_i) \; P (h_i \mid D)$
- **And we want to choose the *class* that maximizes this probability:**
- **argmax c Σi $P(c \mid h_i) \; P (h_i \mid D)$**
-
- **P(h1|D) = 0.4 P(-|h1) = 0 P(+|h1) = 1**
- **P(h2|D) = 0.3 P(-|h2) = 1 P(+|h2) = 0**
- **P(h3|D) = 0.3 P(-|h3) = 1 P(+|h3) = 0**
-
- **And then**
  - $\Sigma i \; P(+ \mid h_i) \; P (h_i \mid D) = 0.4$
  - $\Sigma i \; P(- \mid h_i) \; P (h_i \mid D) = 0.6$

Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a mixture of Bernoulli distributions**

Naïve Bayes Classifier

$$P(Y = y_k | X_1, X_2, ..., X_N) = \frac{P(Y=y_k) P(X_1, X_2, .., X_N | Y=y_k)}{\sum_j P(Y=y_j) P(X_1, X_2, .., X_N | Y=y_j)}$$



Pr(y=yk|X)

Red; green; blue classes
Posterior Class probs

$$= \frac{P(Y=y_k) \Pi_i P(X_i | Y=y_k)}{\sum_j P(Y=y_j) \Pi_i P(X_i | Y=y_j)}$$

$$Y \leftarrow argmax_{y_k} P(Y = y_k) \Pi_i P(X_i | Y = y_k)$$

Naïve Bayes Classifier for Text

$$P(Y = y_k | X_1, X_2, ..., X_N) = \frac{P(Y=y_k) P(X_1, X_2, .., X_N | Y=y_k)}{\sum_j P(Y=y_j) P(X_1, X_2, .., X_N | Y=y_j)}$$

Y_1

Y_2

$$= \frac{P(Y=y_k) \Pi_i P(X_i | Y=y_k)}{\sum_j P(Y=y_j) \Pi_i P(X_i | Y=y_j)}$$

Pr("corporation"|Class=Business) = 1/100
10,000Words in the 10 business documents
"corporation" occurs 100 times

$$Y \leftarrow argmax_{y_k} P(Y = y_k) \Pi_i P(X_i | Y = y_k)$$

argmax_yk means find the value of yk that maximises the expression

Probability Basics

- Prior, conditional and joint probability
  - Prior probability: $P(X)$
  - Conditional probability: $P(X_1 \mid X_2), P(X_2 \mid X_1)$
  - Joint probability: $\mathbf{X} = (X_1, X_2), P(\mathbf{X}) = P(X_1, X_2)$
  - Relationship: $P(X_1, X_2) = P(X_2 \mid X_1)P(X_1) = P(X_1 \mid X_2)P(X_2)$
  - Independence: $P(X_2 \mid X_1) = P(X_2), P(X_1 \mid X_2) = P(X_1), P(X_1, X_2) = P(X_1)P(X_2)$
- Bayesian Rule

$$P(C \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C)P(C)}{P(\mathbf{X})} \qquad Posterior = \frac{Likelihood \times Prior}{Evidence}$$

Probabilistic Classification

- Establishing a probabilistic model for classification
  - Discriminative model

$$P(C \mid \mathbf{X}) \quad C = c_1, \cdots, c_L, \mathbf{X} = (X_1, \cdots, X_n)$$

  - Generative model

$$P(\mathbf{X} \mid C) \quad C = c_1, \cdots, c_L, \mathbf{X} = (X_1, \cdots, X_n)$$

- MAP classification rule
  - MAP: Maximum A Posterior
  - Assign x to c^* if $P(C = c^* \mid \mathbf{X} = \mathbf{x}) > P(C = c \mid \mathbf{X} = \mathbf{x}) \quad c \neq c^*, \; c = c_1, \cdots, c_L$
- Generative classification with the MAP rule
  - Apply Bayesian rule to convert $P(C \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid C)P(C)}{P(\mathbf{X})} \propto P(\mathbf{X} \mid C)P(C)$

Naive Bayes for Discrete-Valued Inputs

When the n input attributes X_i each take on J possible discrete values, and Y is a discrete variable taking on K possible values, then our learning task is to estimate two sets of parameters. The first is

$$\theta_{ijk} \equiv P(X_i = x_{ij} | Y = y_k) \tag{4}$$

for each input attribute X_i, each of its possible values x_{ij}, and each of the possible values y_k of Y. Note there will be nJK such parameters, and note also that only $n(J-1)K$ of these are independent, given that they must satisfy $1 = \sum_j \theta_{ijk}$ for each pair of i, k values.

In addition, we must estimate parameters that define the prior probability over Y:

$$\pi_k \equiv P(Y = y_k) \tag{5}$$

Note there are K of these parameters, $(K-1)$ of which are independent.

We can estimate these parameters using either maximum likelihood estimates (based on calculating the relative frequencies of the different events in the data), or using Bayesian MAP estimates (augmenting this observed data with prior distributions over the values of these parameters).

Maximum likelihood estimates for θ_{ijk} given a set of training examples D are given by

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij} | Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\}}{\#D\{Y = y_k\}} \quad (6)$$

where the $\#D\{x\}$ operator returns the number of elements in the set D that satisfy property x.

One danger of this maximum likelihood estimate is that it can sometimes result in θ estimates of zero, if the data does not happen to contain any training examples satisfying the condition in the numerator. To avoid this, it is common to use a "smoothed" estimate which effectively adds in a number of additional "hallucinated" examples, and which assumes these hallucinated examples are spread evenly over the possible values of X_i. This smoothed estimate is given by

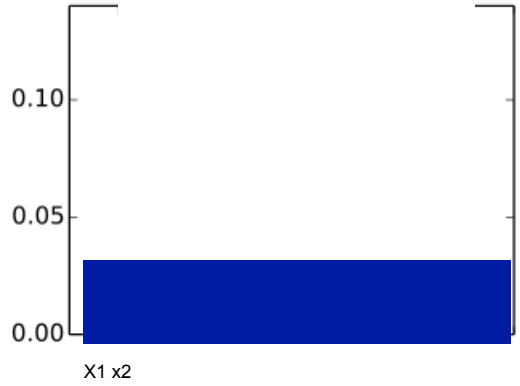
$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_{ij} | Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\} + l}{\#D\{Y = y_k\} + lJ} \quad (7)$$

where J is the number of distinct values X_i can take on, and l determines the strength of this smoothing (i.e., the number of hallucinated examples is lJ). This expression corresponds to a MAP estimate for θ_{ijk} if we assume a Dirichlet prior distribution over the θ_{ijk} parameters, with equal-valued parameters. If l is set to 1, this approach is called Laplace smoothing.

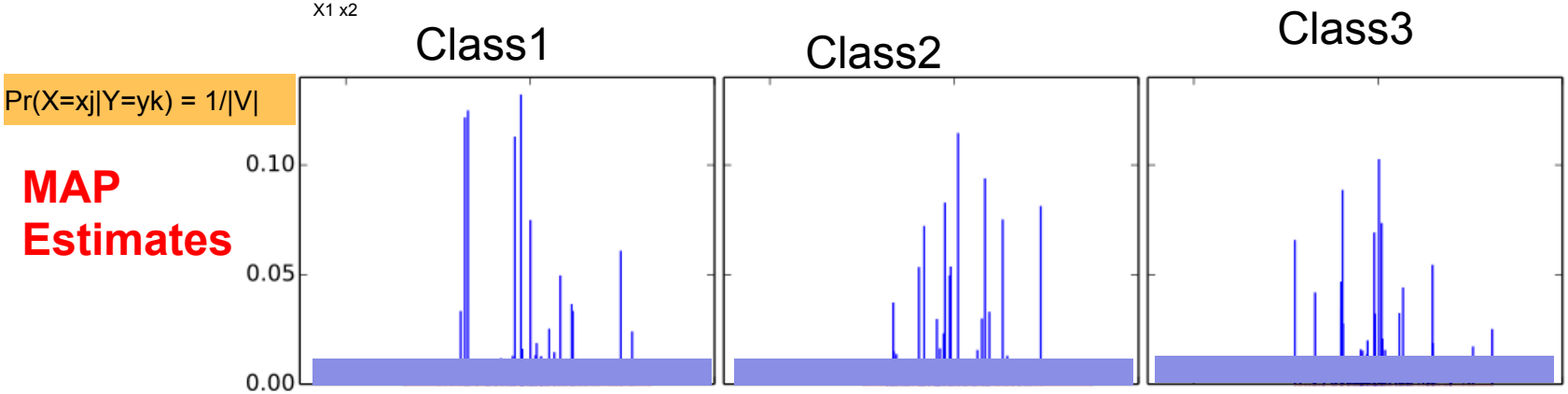
ML Estimates
Learning a NB Via Maximum Likelihood

MAP Estimates
Learning a NB Via Bayesian hierarchical model

MAP Estimates: Smoothed Probabilities

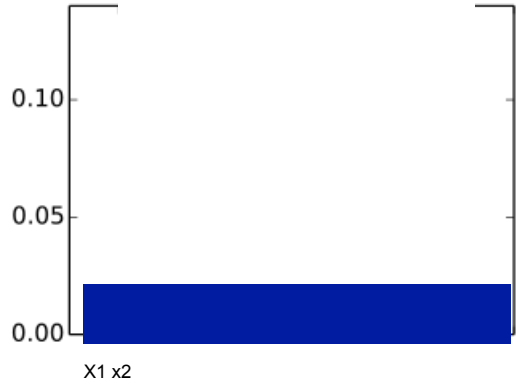


Uniform Prior: Pr(X=xj|Y=yk) = 1/|V|
+
Training data

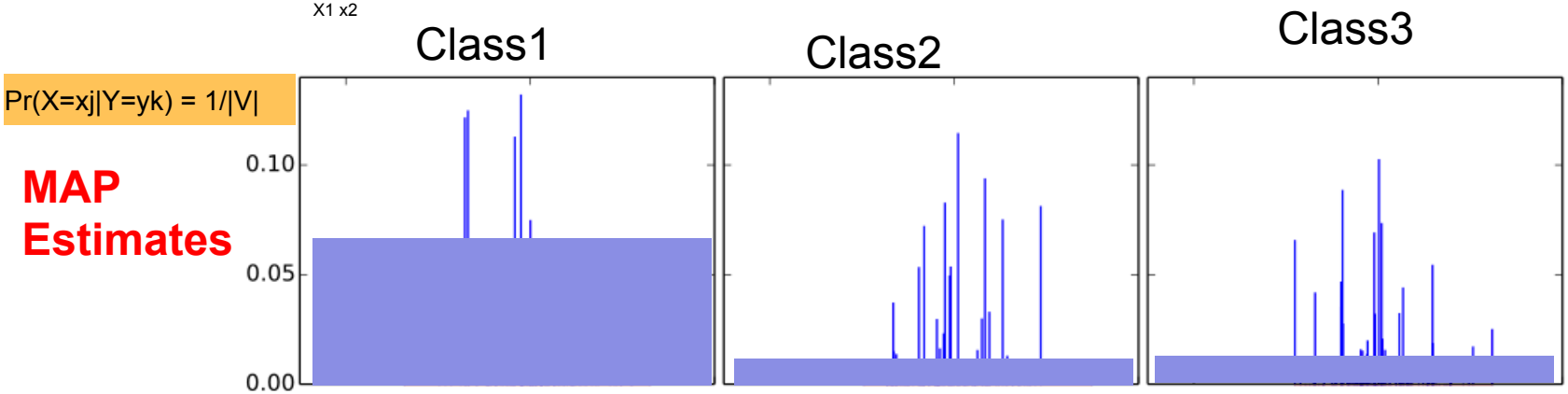


Huge Probability mass assigned to background model:
Bias versus variance?

MAP Estimates: Smoothed Probabilities

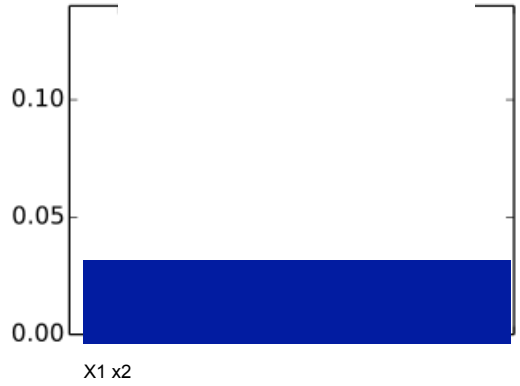


Uniform Prior: Pr(X=xj|Y=yk) = 1/|V|
+
Training data

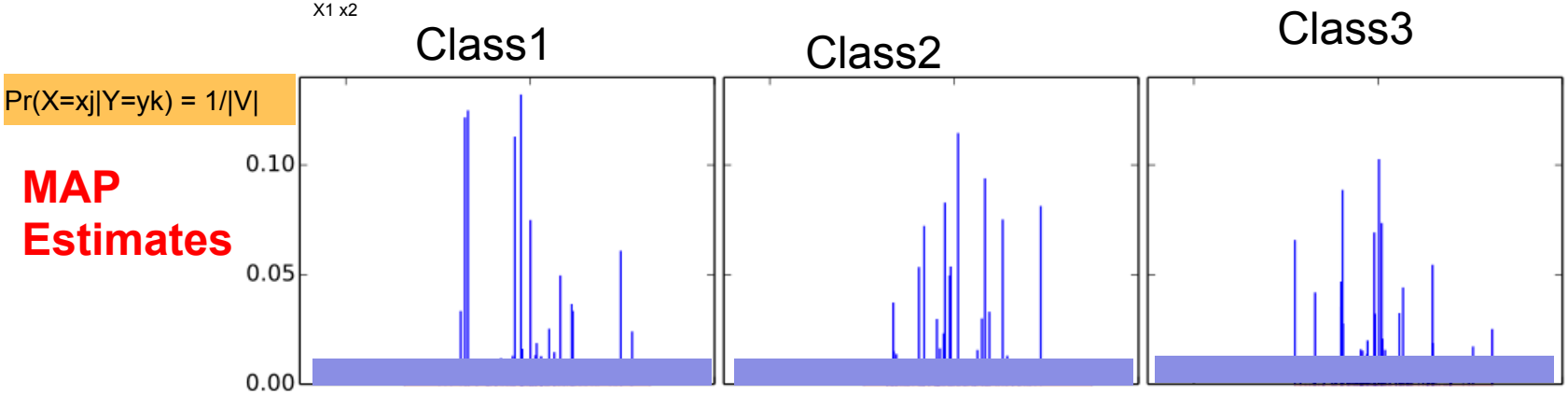


Huge Probability mass assigned to background model:
Bias versus variance?

MAP Estimates: Smoothed Probabilities



Uniform Prior: Pr(X=xj|Y=yk) = 1/|V|
+
Training data



Huge Probability mass assigned to background model:
Bias versus variance? High Bias!

Class Prior Estimates

Maximum likelihood estimates for π_k are

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|} \tag{8}$$

where $|D|$ denotes the number of elements in the training set D.

Alternatively, we can obtain a smoothed estimate, or equivalently a MAP estimate based on a Dirichlet prior over the π_k parameters assuming equal priors on each π_k, by using the following expression

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\} + l}{|D| + lK} \tag{9}$$

where K is the number of distinct values Y can take on, and l again determines the strength of the prior assumptions relative to the observed data D.

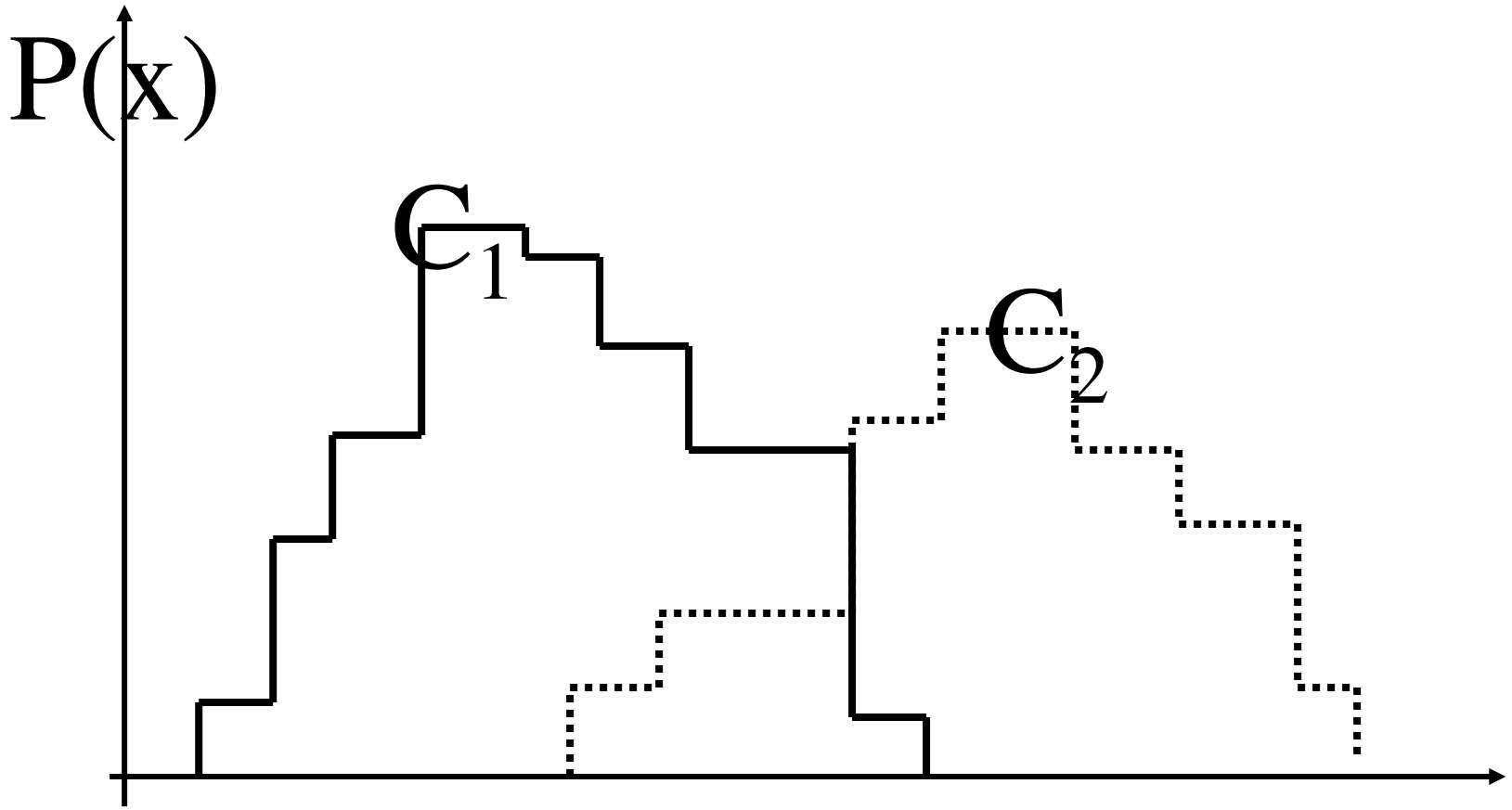
Naive Bayes for Continuous Inputs

- When the Xi are continuous we must choose some other way to represent the distributions
- P(Xi|Y)=Gaussian(Mu, Sigma).
- One common approach is to assume that for each possible discrete value yk of Y, the distribution of each continuous Xi is Gaussian, and is defined by a mean and standard deviation specific to Xi and yk.
- In order to train such a Naïve Bayes classifier we must therefore estimate the mean and standard deviation ians:

$$\mu_{ik} = E[X_i | Y = y_k]$$

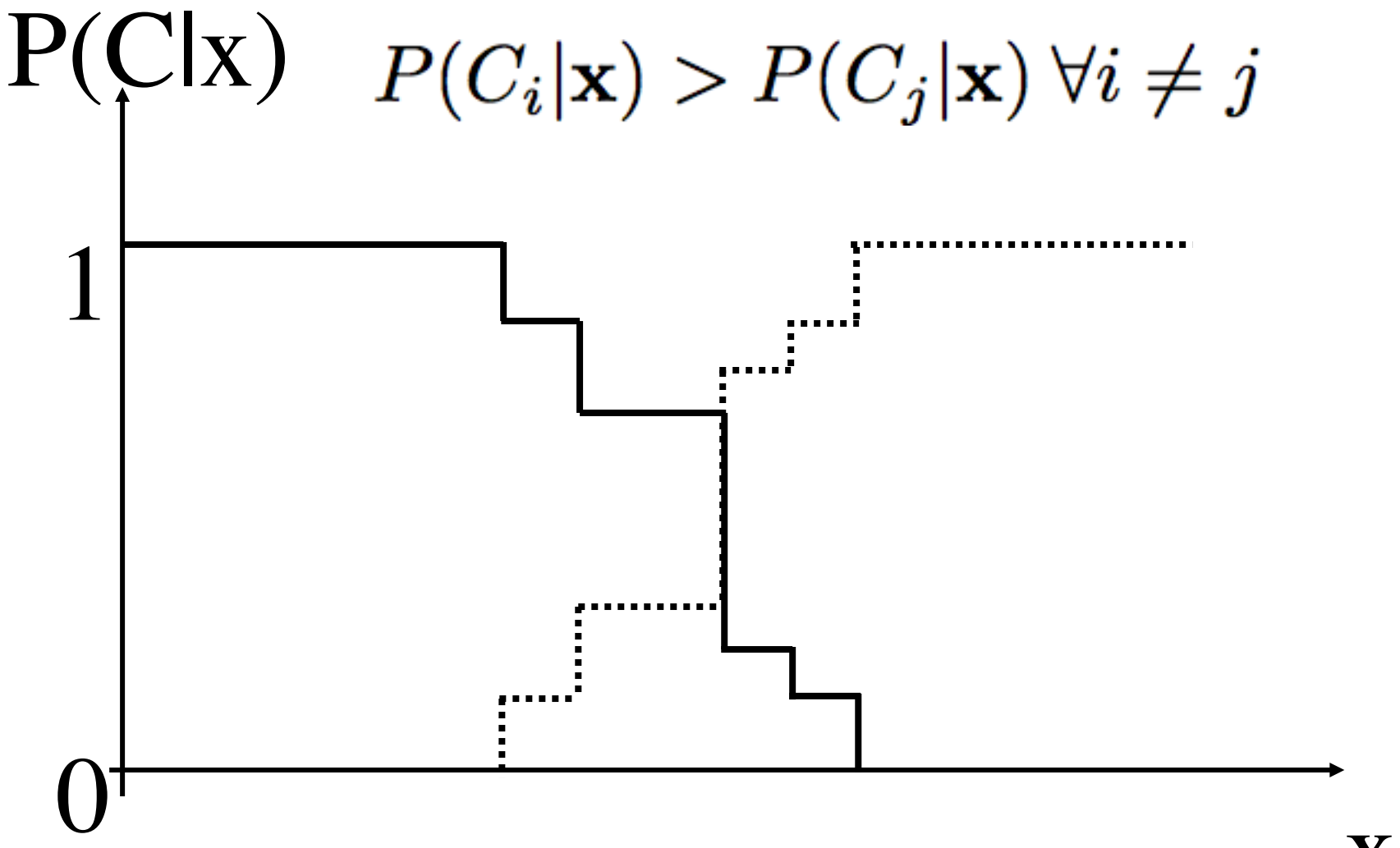
$$\sigma_{ik}^2 = E[(X_i - \mu_{ik})^2 | Y = y_k]$$

Feature Histograms



Slide by Stephen Marsland

Posterior Probability



Slide by Stephen Marsland

MLE-based Estimates

Again, we can use either maximum likelihood estimates (MLE) or maximum a posteriori (MAP) estimates for these parameters. The maximum likelihood estimator for μ_{ik} is

$$\hat{\mu}_{ik} = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j X_i^j \delta(Y^j = y_k) \tag{13}$$

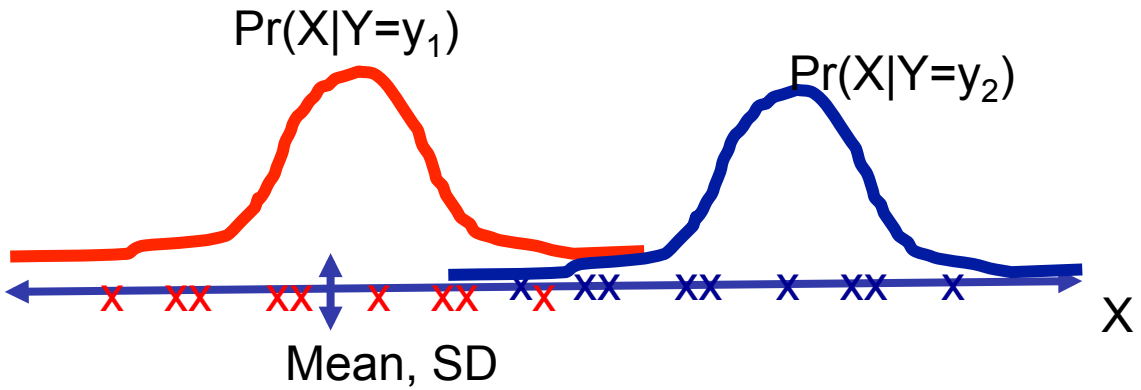
where the superscript j refers to the jth training example, and where $\delta(Y = y_k)$ is 1 if $Y = y_k$ and 0 otherwise. Note the role of δ here is to select only those training examples for which $Y = y_k$.

The maximum likelihood estimator for σ_{ik}^2 is

$$\hat{\sigma}_{ik}^2 = \frac{1}{\sum_j \delta(Y^j = y_k)} \sum_j (X_i^j - \hat{\mu}_{ik})^2 \delta(Y^j = y_k) \tag{14}$$

Estimate μ, σ from data

$$\mu_{ik} = E[X_i | Y = y_k]$$
$$\sigma_{ik}^2 = E[(X_i - \mu_{ik})^2 | Y = y_k]$$



Continuous Inputs: Pr(Y|X) ~ N(μ, σ)

If X is Normally distributed with mean μ and standard deviation σ, we write

$$\boxed{X \sim N(\mu, \sigma^2)}$$

μ and σ are the **parameters** of the distribution.

The probability density of the Normal distribution is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-(x-\mu)^2/2\sigma^2}$$

For the purposes of this course we do not need to use this expression. It is included here for future reference.

Naïve Bayes

- **Combine discrete input variables with continuous input variables?**
  - Naïve Bayes, Decision trees
- **Whereas these requires feature transformations**
  - Logistic regression: one hot-encoding

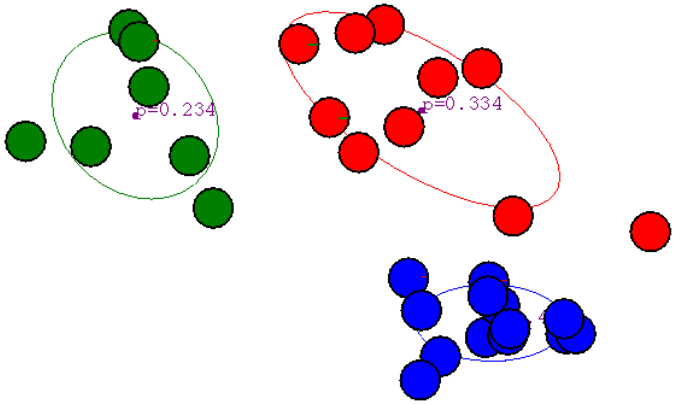
- **YES**

Flat Clustering: Hard vs. soft clustering

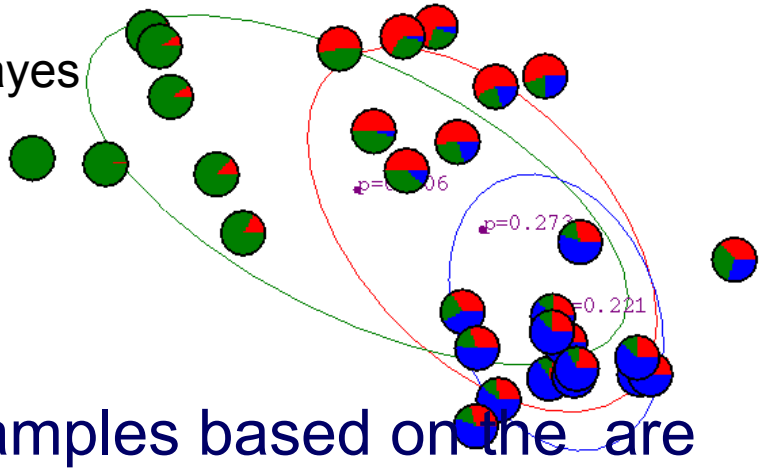
- **Hard clustering: Each example(document) belongs to exactly one cluster**
  - More common and easier to do
- **Soft clustering: An example can belong to more than one cluster.**
  - Makes more sense for applications like creating browsable hierarchies
  - You may want to put a "*pair of sneakers*" in two clusters: (i) *sports apparel* and (ii) *shoes*
  - You can only do that with a soft clustering approach.
- **See book IIR Section 16.5, 18**

Flat Clustering: Hard versus Soft

- **Hard Clustering**
  - Hard assignment; Kmeans (EM-like)
- **Soft Clustering**
  - This set of assignment probabilities (aka responsibilities) defines a soft clustering.
  - Model-based Clustering using EM and Maximum Likelihood



Naïve Bayes



  - Weighted EM-like
    - EM Centroids are weighted examples based on the are the cluster/class probabilities assignment probabilities (aka responsibilites)

EM Algorithm: distances vs Distributions

16.5 Model-based clustering

In this section, we describe a generalization of K-means, the EM algorithm. It can be applied to a larger variety of document representations and distributions than K-means.

In K-means, we attempt to find centroids that are good representatives. We can view the set of K centroids as a model that generates the data. Generating a document in this model consists of first picking a centroid at random and then adding some noise. If the noise is normally distributed, this procedure will result in clusters of spherical shape. *Model-based clustering* assumes that the data were generated by a model and tries to recover the original model from the data. The model that we recover from the data then defines clusters and an assignment of documents to clusters.

MODEL-BASED CLUSTERING

A commonly used criterion for estimating the model parameters is maximum likelihood.

Soft Clustering versus Hard Clustering

- Once we have a model Θ, we can compute an assignment probability **$P(d|\omega_k; \Theta)$ (AKA ($P(d|z_k; \Theta)$))** for each document-cluster pair.
- This set of assignment probabilities defines a soft clustering.
  - An example of a soft assignment is that a document about Chinese cars may have a fraction a membership of 0.5 in each of the two clusters China and automobiles, reflecting the fact that both topics are pertinent
- K-means and the hierarchical algorithms make fairly rigid assumptions about the data.
  - For example, clusters in K-means are assumed to be spheres.

Model-based clustering more flexible

- Model-based clustering provides a framework for incorporating our knowledge about a domain.
- **K** -means and the hierarchical algorithms make fairly rigid assumptions about the data.
  - For example, clusters in K -means are assumed to be spheres.
  - Model-based clustering offers more flexibility.
- The clustering model can be adapted to what we know about the underlying distribution of the data, be it Bernoulli, Gaussian with non-spherical variance (another model that is important in document clustering) or a member of a different family.

Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a mixture of Bernoulli distributions**

Maximum Likelihood Estimation (MLE)

- We have reduced the problem of selecting the best model to that of selecting the best parameter(s).
- E.g.,
- We want to select a parameter p which will maximize the probability that the data was generated from the model with the parameter p plugged-in.
- The parameter p is called the maximum likelihood estimator.
- The maximum of the function can be obtained by setting the derivative of the function ==0 and solving for p.

Two Important Facts

- If $A_1,\ldots,A_n$ are independent then

$$P(A_1,\ldots,A_n) = \prod_{i=1}^{n} P(A_i)$$

- The log function is monotonically increasing.
- x · y ! Log(x) · Log(y)

- Therefore if a function f(x) >= 0, achieves a maximum at x1, then log(f(x)) also achieves a maximum at x1.

MLE for binary events: solve

- **Suppose we have the following data**
  - 0,1,1,0,0,1,1,0
- **In this case it is sensible to choose the Bernoulli distribution (B(p)) as the model space.**
- **Now we want to choose the best p, i.e.,**

http://math.arizona.edu/~jwatkins/n-mle.pdf

$$P(X = x) = p^x(1-p)^{1-x}$$

Shorthand

$$\text{argmax}_p P(Data|B(p))$$

Example 2 (Bernoulli trials). *If the experiment consists of n Bernoulli trial with success probability θ, then*

$$\mathbf{L}(\theta|\mathbf{x}) = \theta^{x_1}(1-\theta)^{(1-x_1)}\cdots\theta^{x_n}(1-\theta)^{(1-x_n)} = \theta^{(x_1+\cdots+x_n)}(1-\theta)^{n-(x_1+\cdots+x_n)}.$$

$$\ln \mathbf{L}(\theta|\mathbf{x}) = \ln\theta(\sum_{i=1}^{n} x_i) + \ln(1-\theta)(n-\sum_{i=1}^{n} x_i) = n\bar{x}\ln\theta + n(1-\bar{x})\ln(1-\theta).$$

Take gradient
Set it equal to 0 and solve

$$\frac{\partial}{\partial\theta}\ln \mathbf{L}(\theta|\mathbf{x}) = n\left(\frac{\bar{x}}{\theta} - \frac{1-\bar{x}}{1-\theta}\right).$$

This equals zero when $\theta = \bar{x}$. Check that this is a maximum. Thus,

$$\hat{\theta}(\mathbf{x}) = \bar{x}.$$

$\hat{\theta}(\mathbf{x})$ is a maximum likelihood estimator for θ.

$$\hat{\theta}(\mathbf{x}) = \frac{\text{\#Successes}}{\text{\#Trials}}$$

Example of MLE

$$
\begin{aligned}
L(p) &= P(0,1,1,0,0,1,0,1|p) \\
&= P(0|p)P(1|p)\ldots P(1|p) \\
&= (1-p)p\ldots p \\
&= p^4(1-p)^4
\end{aligned}
$$

- **Now, choose p which maximizes L(p). Instead we will maximize l(p)= LogL(p)**

$$
\begin{aligned}
\ell(p) &= logL(p) = 4log(p) + 4log(1-p) \\
\frac{d\ell(p)}{dp} &= \frac{4}{p} - \frac{4}{1-p} \equiv 0 \\
\rightarrow p &= \frac{1}{2}
\end{aligned}
$$

As before, we begin with a sample $X = (X_1, \ldots, X_n)$ of random variables chosen according to one of a family of probabilities P_θ.

In addition, $\mathbf{f}(\mathbf{x}|\theta)$, $\mathbf{x} = (x_1, \ldots, x_n)$ will be used to denote the density function for the data when θ is the true state of nature.

Definition 1. *The* **likelihood function** *is the density function regarded as a function of* θ.

$$\mathbf{L}(\theta|\mathbf{x}) = \mathbf{f}(\mathbf{x}|\theta),\ \theta \in \Theta. \tag{1}$$

The **maximum likelihood estimator (MLE),**

$$\hat{\theta}(\mathbf{x}) = \arg\max_\theta \mathbf{L}(\theta|\mathbf{x}). \tag{2}$$

Example 2 (Bernoulli trials). *If the experiment consists of* n *Bernoulli trial with success probability* θ, *then*

$$\mathbf{L}(\theta|\mathbf{x}) = \theta^{x_1}(1-\theta)^{(1-x_1)} \cdots \theta^{x_n}(1-\theta)^{(1-x_n)} = \theta^{(x_1+\cdots+x_n)}(1-\theta)^{n-(x_1+\cdots+x_n)}.$$

$$\ln \mathbf{L}(\theta|\mathbf{x}) = \ln\theta(\sum_{i=1}^{n} x_i) + \ln(1-\theta)(n - \sum_{i=1}^{n} x_i) = n\bar{x}\ln\theta + n(1-\bar{x})\ln(1-\theta).$$

$$\frac{\partial}{\partial\theta}\ln \mathbf{L}(\theta|\mathbf{x}) = n\left(\frac{\bar{x}}{\theta} - \frac{1-\bar{x}}{1-\theta}\right).$$

This equals zero when $\theta = \bar{x}$. *Check that this is a maximum. Thus,*

$$\hat{\theta}(\mathbf{x}) = \bar{x}.$$

$\hat{\theta}(\mathbf{x})$ is a maximum likelihood estimator for θ.

$\hat{\theta}(\mathbf{x})$ = #Successes / #Trials

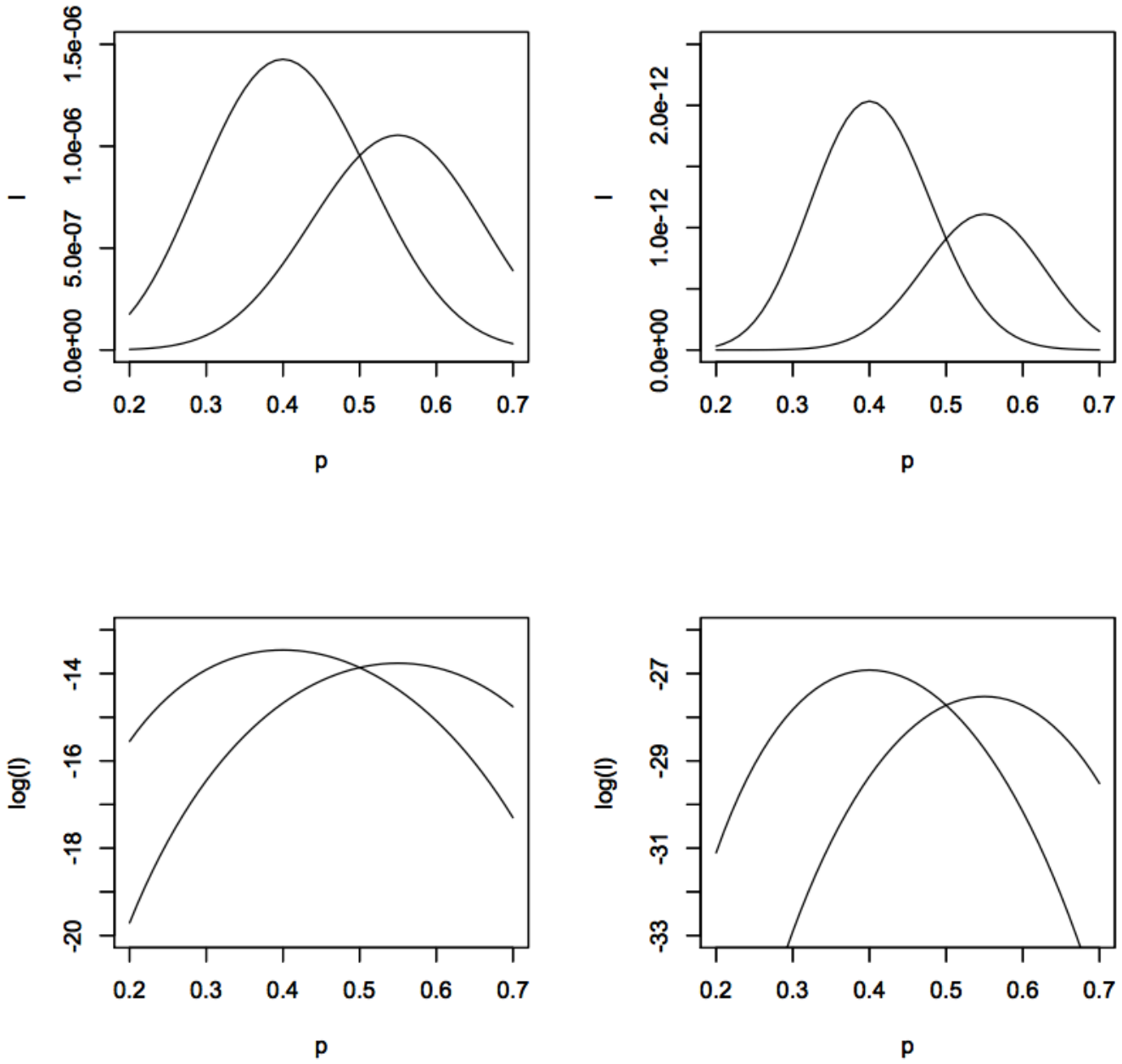
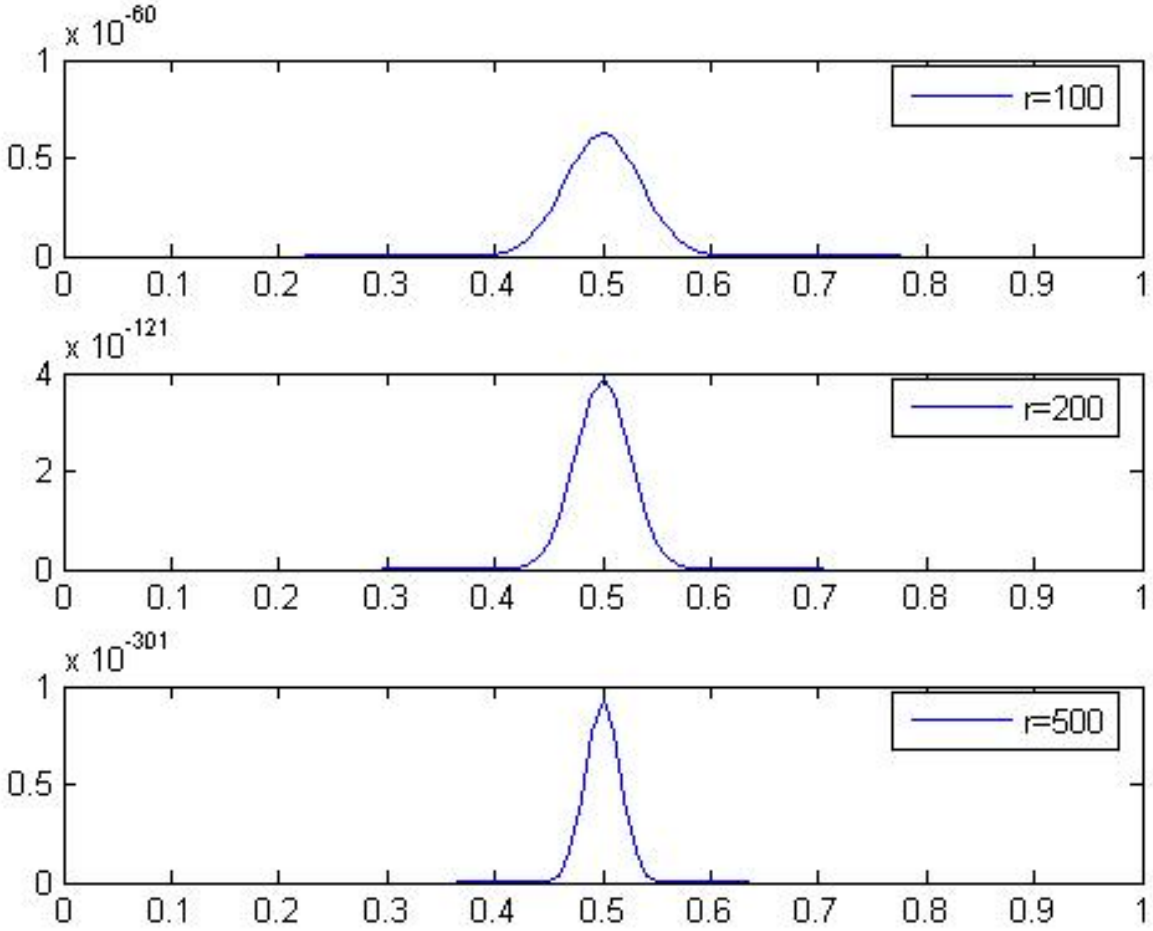


Figure 1: Likelihood function (top row) and its logarithm, the score function, (bottom row) for Bernouli trials. The left column is based on 20 trials having 8 and 11 successes. The right column is based on 40 trials having 16 and 22 successes. Notice that the maximum likelihood is approximately 10^{-6} for 20 trials and 10^{-12} for 40. Note that the peaks are more narrow for 40 trials rather than 20.

Properties of MLE

- **There are several technical properties of the estimator but lets look at the most intuitive one:**
  - As the number of data points increase we become more sure about the parameter p

Properties of MLE



r is the number of data points. As the number of data points increase the confidence of the estimator increases.

Matlab commands

- [phat,ci]=mle(Data,'distribution','Bernoulli');
- [phi,ci]=mle(Data,'distribution','Normal');

Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a mixture of Bernoulli distributions**

Gaussian Model Example

Suppose the following are marks in a course

55.5, 67, 87, 48, 63

Marks typically follow a Normal distribution whose density function is

$$N(\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2\sigma}(x-\mu)^2}$$

Now, we want to find the best μ,σ such that

$$argmax_{\mu,\sigma} p(Data|\mu, \sigma)$$

$$\hat{\mu}(\mathbf{x}) = \bar{x}$$

$$\hat{\sigma}^2(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x})^2.$$

$\hat{\mu}(\mathbf{x})\ \hat{\sigma}^2(\mathbf{x})$ **are MLE for μ, σ^2**

Derive in closed form

Multivariate Normal Distributions

1.1 The multivariate normal distribution

The multivariate normal distribution in n-dimensions, also called the multivariate Gaussian distribution, is parameterized by a **mean vector** $\mu \in \mathbb{R}^n$ and a **covariance matrix** $\Sigma \in \mathbb{R}^{n \times n}$, where $\Sigma \geq 0$ is symmetric and positive semi-definite. Also written "$\mathcal{N}(\mu, \Sigma)$", its density is given by:

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

In the equation above, "$|\Sigma|$" denotes the determinant of the matrix Σ.

For a random variable X distributed $\mathcal{N}(\mu, \Sigma)$, the mean is (unsurprisingly) given by μ:

$$\mathrm{E}[X] = \int_x x\, p(x; \mu, \Sigma) dx = \mu$$

The **covariance** of a vector-valued random variable Z is defined as $\mathrm{Cov}(Z) = \mathrm{E}[(Z - \mathrm{E}[Z])(Z - \mathrm{E}[Z])^T]$. This generalizes the notion of the variance of a

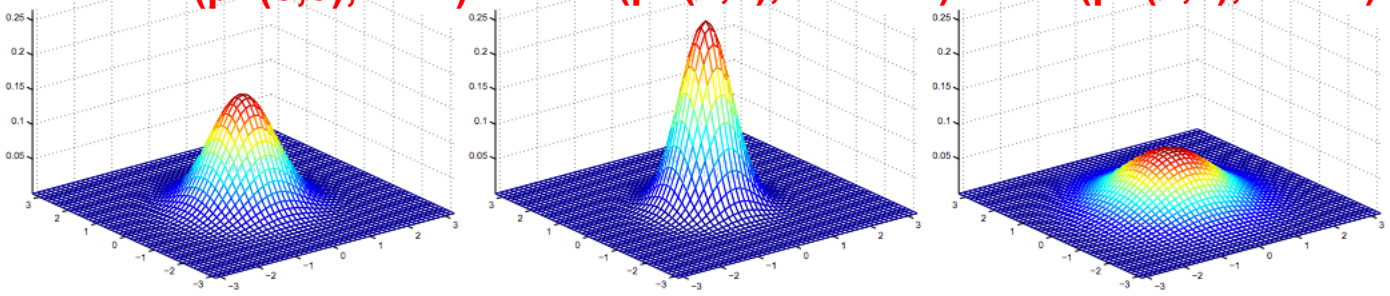
Covariance: E.g., standard normal distribution

- real-valued random variable. The covariance can also be defined as $\mathrm{Cov}(Z) = \mathrm{E}[ZZ^T] - (\mathrm{E}[Z])(\mathrm{E}[Z])^T$. (You should be able to prove to yourself that these two definitions are equivalent.) If $X \sim \mathcal{N}(\mu, \Sigma)$, then

$$\mathrm{Cov}(X) = \Sigma.$$

Here're some examples of what the density of a Gaussian distribution looks like:

X ~ N(μ=(0,0), Σ = I) X ~ N(μ=(0,0), Σ = 0.6I) X ~ N(μ=(0,0), Σ = 2I)

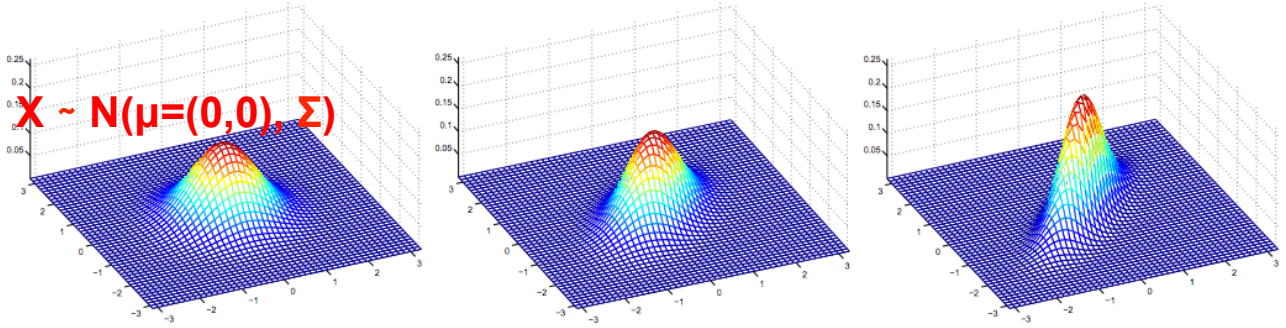


The left-most figure shows a Gaussian with mean zero (that is, the 2x1 zero-vector) and covariance matrix $\Sigma = I$ (the 2x2 identity matrix). A Gaussian with zero mean and identity covariance is also called the **standard normal distribution**. The middle figure shows the density of a Gaussian with zero mean and $\Sigma = 0.6I$; and in the rightmost figure shows one with , $\Sigma = 2I$. We see that as Σ becomes larger, the Gaussian becomes more "spread-out," and as it becomes smaller, the distribution becomes more "compressed."

Std versus varying off-diagonal elements in Σ

Let's look at some more examples.

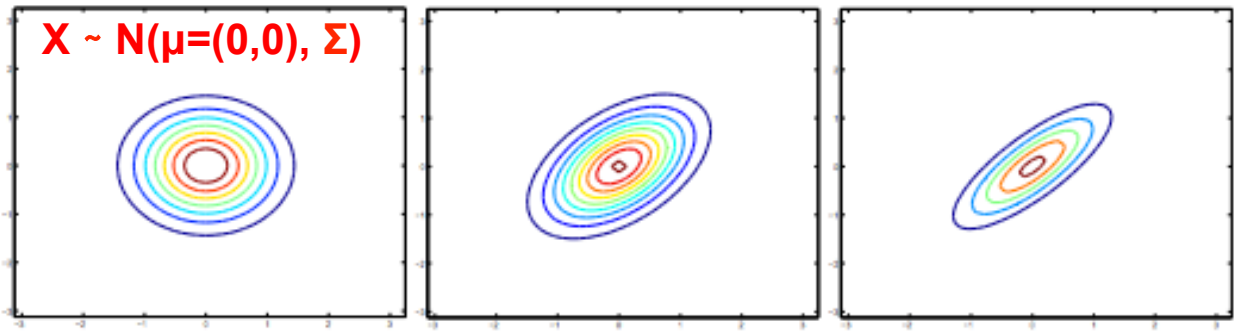
- ..



The figures above show Gaussians with mean 0, and with covariance matrices respectively

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}.$$

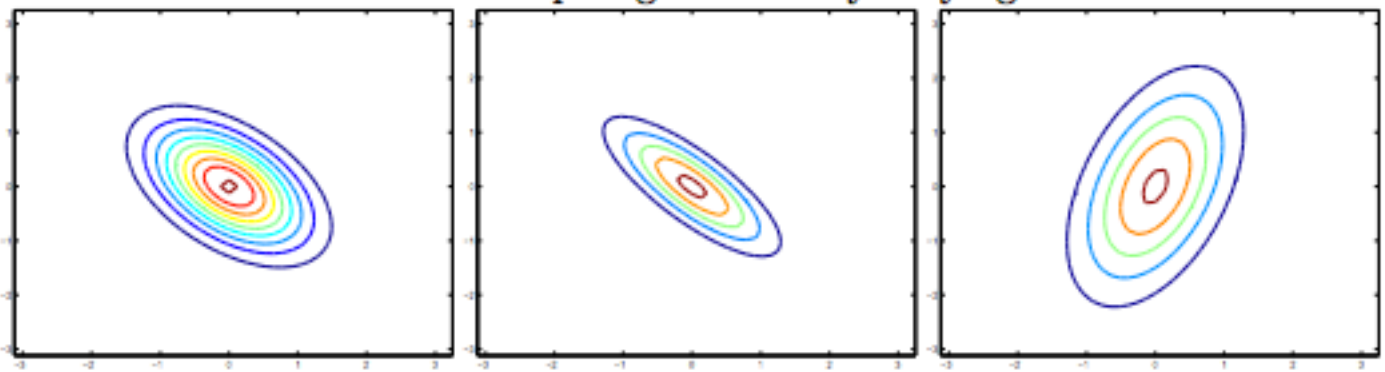
The leftmost figure shows the familiar standard normal distribution, and we see that as we increase the off-diagonal entry in Σ, the density becomes more "compressed" towards the 45° line (given by $x_1 = x_2$). We can see this more clearly when we look at the contours of the same three densities:



varying off-diagonal elements in Σ: bigger, positive versus negative

- ..

Here's one last set of examples generated by varying Σ:



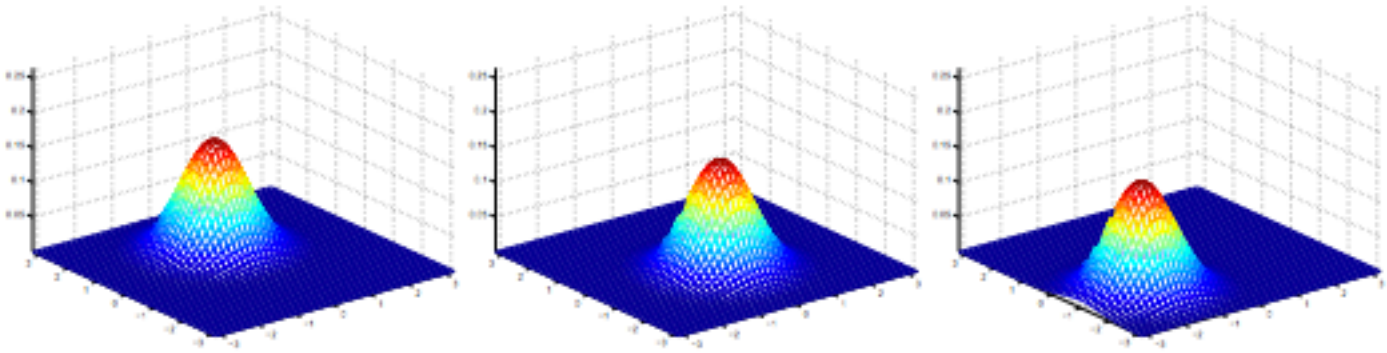
The plots above used, respectively,

$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}.$$

From the leftmost and middle figures, we see that by decreasing the diagonal elements of the covariance matrix, the density now becomes "compressed" again, but in the opposite direction. Lastly, as we vary the parameters, more generally the contours will form ellipses (the rightmost figure showing an example).

Let covariance be identity matrix and vary μ

- As our last set of examples, fixing $\Sigma = I$, by varying μ, we can also move the mean of the density around.



The figures above were generated using $\Sigma = I$, and respectively

$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad \mu = \begin{bmatrix} -0.5 \\ 0 \end{bmatrix}; \quad \mu = \begin{bmatrix} -1 \\ -1.5 \end{bmatrix}.$$

Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a mixture of Bernoulli distributions**

Gaussian Model: Estimate Parameters

Suppose the following are marks in a course

55.5, 67, 87, 48, 63

Marks typically follow a Normal distribution whose density function is

$$N(\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2\sigma}(x-\mu)^2}$$

Now, we want to find the best μ,σ such that

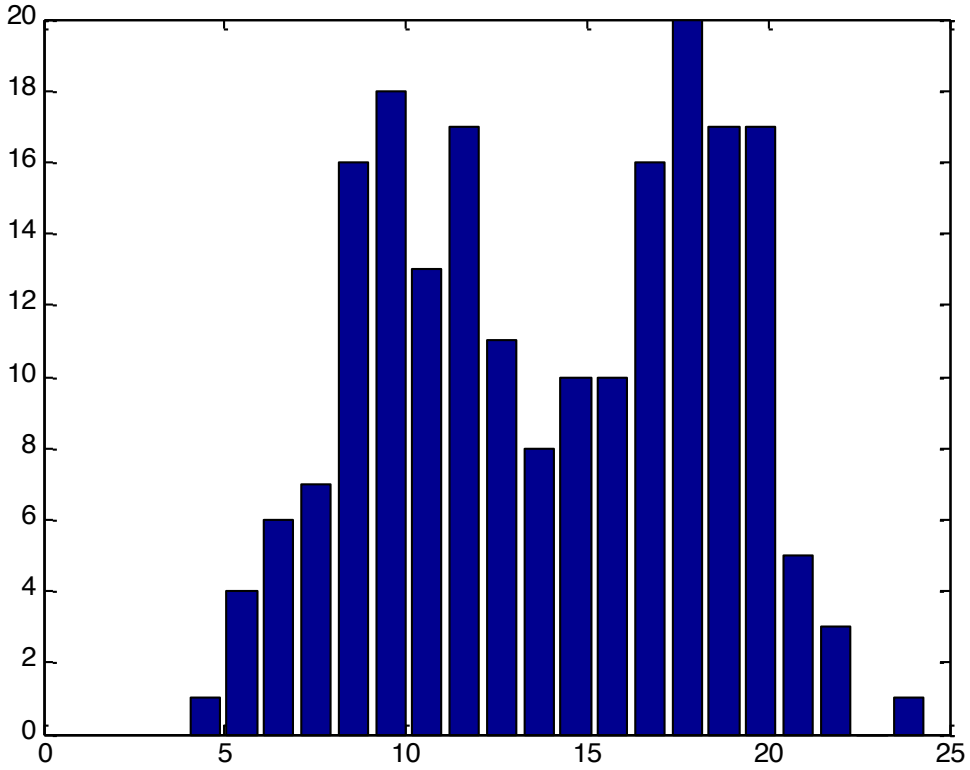
$$argmax_{\mu,\sigma} p(Data|\mu, \sigma)$$

$$\hat{\mu}(\mathbf{x}) = \bar{x}$$

$$\hat{\sigma}^2(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x})^2.$$

$\hat{\mu}(\mathbf{x})\ \hat{\sigma}^2(\mathbf{x})$ are MLE for μ, σ^2

A Mixture Model Problem



- Apparently, the dataset consists of two modes
- How can we automatically identify the two modes?

Motivation

- **Problem:**

  Describe data with Mixture Model(MM)

- **Approach:**
  - Decide on MM, e.g.
    - Gauss distribution
    - Mix of two
  - Estimate parameters

$$p(x \mid \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$p(x \mid \Theta) = \alpha_1 p_1(x \mid \mu_1, \sigma_1^2) + \alpha_2 p_2(x \mid \mu_2, \sigma_2^2)$$

$$\Theta = (\alpha_1, \alpha_2, \quad \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$$

Gaussian Mixture Model (GMM)

- **Assume that the dataset is generated by two mixed Gaussian distributions**
  - Gaussian model 1: $\theta_1 = \{\mu_1, \sigma_1; p_1\}$
  - Gaussian model 2: $\theta_2 = \{\mu_2, \sigma_2; p_2\}$
- **If we know the memberships for each bin, estimating the two Gaussian models is easy.**
- **How to estimate the two Gaussian models without knowing the memberships of bins?**

Review

- **Stochastically independent**

$$p(A,B) = p(A)\,p(B)$$

- **Bayes' rule**

$$p(X \mid \theta) = \frac{p(X,Y \mid \theta)}{p(Y \mid X,\theta)}$$

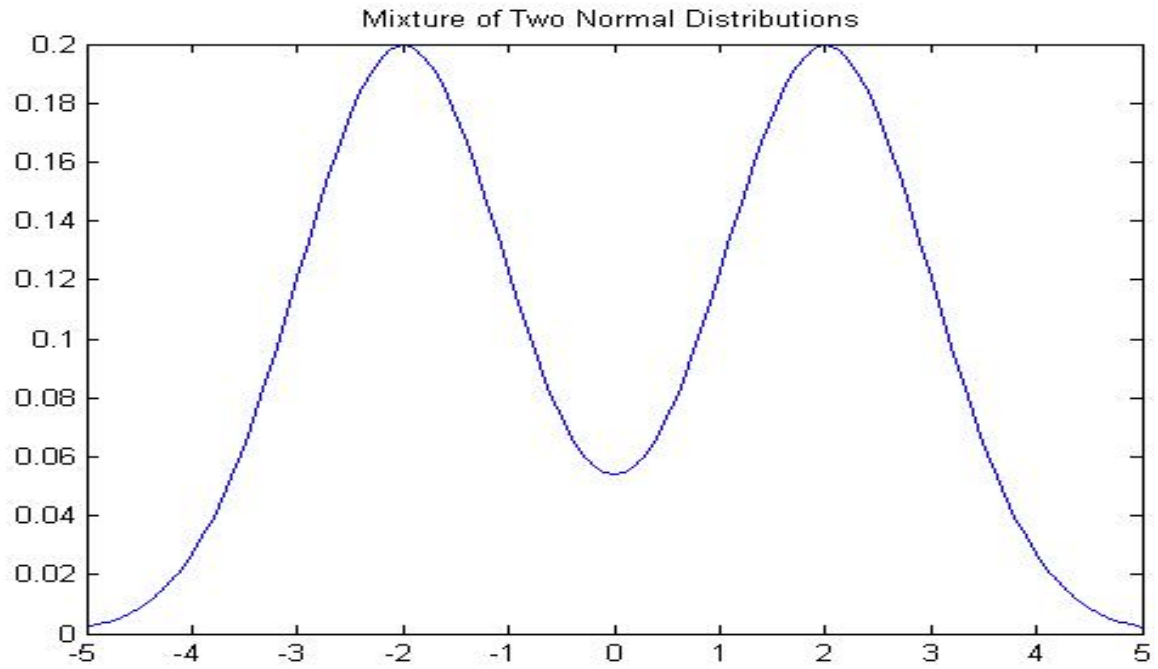
- **Logarithm**

$$\ln ab = \ln a + \ln b$$

- **Expectation**

$$E\left[Y \mid X = x\right] = \int y\, f_{Y|X}(x,y)\, dy$$

A Mixture Distribution

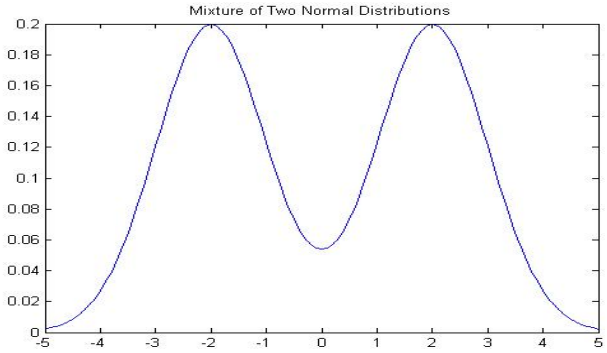


$$f(\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1, \sigma_2) = \pi_1 N(\mu_1, \sigma_1) + \pi_2 N(\mu_2, \sigma_2)$$

Mixture of Gaussian Distributions Examples

- **Suppose we have data about heights of people (in cm)**
  - 185,140,134,150,170
- **Heights follow a normal (log normal) distribution but men on average are taller than women. This suggests a mixture of two distributions**

$$f(\pi_1, \pi_2, \mu_1, \mu_2, \sigma_1, \sigma_2) = \pi_1 N(\mu_1, \sigma_1) + \pi_2 N(\mu_2, \sigma_2)$$



Recap

- Multivariate Gaussian

$$p(\mathbf{x}|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

  - By translation and rotation, it turns into multiplication of **normal distributions**
  - MLE of mean: $\hat{\mu} = \frac{\Sigma_i x_i}{N}$
  - MLE of covariance: $\hat{\Sigma} = \frac{\Sigma_i (x_i - \hat{\mu})(x_i - \hat{\mu})^T}{N}$

MLE for Mixture Distributions

- When we proceed to calculate the MLE for a mixture, the presence of the sum of the distributions prevents a "neat" factorization using the log function.
- A completely new rethink is required to estimate the parameter.
- The new rethink also provides a solution to the clustering problem.

Missing Data

- We think of clustering as a problem of estimating missing data.
- The missing data are the cluster labels.
- Clustering is only one example of a missing data problem. Several other problems can be formulated as missing data problems.

Types of classification data

OHE: One hot encoding

Data set is complete.

	x	y	z			
$\mathbf{x}^{(1)}$		2	0	1	0	$\cdots$
$\mathbf{x}^{(2)}$		3	0	0	1	$\cdots$
$\mathbf{x}^{(3)}$		1	1	0	0	$\cdots$
$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Y is the Class; Z is OHE

means: Data set is incomplete, but we complete it using a specific cost function.

Missing Labels

	x	y	z			
$\mathbf{x}^{(1)}$		?	0	1	0	$\cdots$
$\mathbf{x}^{(2)}$		?	0	0	1	$\cdots$
$\mathbf{x}^{(3)}$		?	1	0	0	$\cdots$
$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

https://en.wikipedia.org/wiki/One-hot

EM: data set is incomplete, but we complete it using posterior probabilities (a "soft" class membership).

Z = class; assume 3 clusters/classes

x	y	z			
$\mathbf{x}^{(1)}$	?	$P(z_1 = 1 \mid \mathbf{x}^{(1)}, \theta)$	$P(z_2 = 1 \mid \mathbf{x}^{(1)}, \theta)$	$P(z_3 = 1 \mid \mathbf{x}^{(1)}, \theta)$	$\cdots$
$\mathbf{x}^{(2)}$	?	$P(z_1 = 1 \mid \mathbf{x}^{(2)}, \theta)$	$P(z_2 = 1 \mid \mathbf{x}^{(2)}, \theta)$	$P(z_3 = 1 \mid \mathbf{x}^{(2)}, \theta)$	$\cdots$
$\mathbf{x}^{(3)}$	?	$P(z_1 = 1 \mid \mathbf{x}^{(3)}, \theta)$	$P(z_2 = 1 \mid \mathbf{x}^{(3)}, \theta)$	$P(z_3 = 1 \mid \mathbf{x}^{(3)}, \theta)$	$\cdots$
$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$

Missing Data Problem

- **Let D = {x(1),x(2),…x(n)} be a set of n observations.**
- **Let H = {z(1),z(2),..z(n)} be a set of n values of a hidden variable Z.**
  - z(i) corresponds to x(i)
- **Assume Z is discrete. We know how many clusters**

But $z^{(i)}$'s are not known: so use EM

- **However, in our density estimation problem, the $z^{(i)}$'s are not known.**
- **What can we do?**
- **The EM algorithm is an iterative algorithm that has two main steps.**
  - Applied to our problem, in the E-step, it tries to "guess" the values of the $z_{(i)}$'s.
  - In the M-step, it updates the parameters of our model based on our guesses. Since in the M-step we are pretending that the guesses in the first part were correct, the maximization becomes easy. Here's the algorithm:

Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a mixture of Bernoulli distributions**

EM for density Estimation

- In statistics, an expectation–maximization (EM) algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables.
- The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.

EM background

https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm

- The EM algorithm is used to find (locally) maximum likelihood parameters of a statistical model in cases where the equations cannot be solved directly.
- Typically these models involve latent variables in addition to unknown parameters and known data observations. That is, either there are missing values among the data, or the model can be formulated more simply by assuming the existence of additional unobserved data points.
- For example, a mixture model can be described more simply by assuming that each observed data point has a corresponding unobserved data point, or latent variable, specifying the mixture component that each data point belongs to.

taking the derivatives of the likelihood function does NOT work

- Finding a maximum likelihood solution typically requires taking the derivatives of the likelihood function with respect to all the unknown values — viz. the parameters and the latent variables — and simultaneously solving the resulting equations.
- In statistical models with latent variables, this usually is not possible.
- Instead, the result is typically a set of interlocking equations in which the solution to the parameters requires the values of the latent variables and vice versa, but substituting one set of equations into the other produces an unsolvable equation.

- The EM algorithm proceeds from the observation that the following is a way to solve these two sets of equations numerically.
- One can simply pick arbitrary values for one of the two sets of unknowns, use them to estimate the second set, then use these new values to find a better estimate of the first set, and then keep alternating between the two until the resulting values both converge to fixed points.
- It's not obvious that this will work at all, but in fact it can be proven that in this particular context it does, and that the derivative of the likelihood is (arbitrarily close to) zero at that point, which in turn means that the point is either a maximum or a saddle point.[12]
- In general there may be multiple maxima, and there is no guarantee that the global maximum will be found. Some likelihoods also have singularities in them, i.e. nonsensical maxima.
- For example, one of the "solutions" that may be found by EM in a mixture model involves setting one of the components to have zero variance and the mean parameter for the same component to be equal to one of the data points.

EM for density Estimation

In this set of notes, we discuss the EM (Expectation-Maximization) for density estimation.

Suppose that we are given a training set $\{x^{(1)}, \ldots, x^{(m)}\}$ as usual. Since we are in the unsupervised learning setting, these points do not come with any labels.

We wish to model the data by specifying a joint distribution $p(x^{(i)}, z^{(i)}) = p(x^{(i)}|z^{(i)})p(z^{(i)})$. Here, $z^{(i)} \sim \text{Multinomial}(\phi)$ (where $\phi_j \geq 0$, $\sum_{j=1}^{k} \phi_j = 1$, and the parameter ϕ_j gives $p(z^{(i)} = j),)$, and $x^{(i)}|z^{(i)} = j \sim \mathcal{N}(\mu_j, \Sigma_j)$. We let k denote the number of values that the $z^{(i)}$'s can take on. Thus, our model posits that each $x^{(i)}$ was generated by randomly choosing $z^{(i)}$ from $\{1, \ldots, k\}$, and then $x^{(i)}$ was drawn from one of k Gaussians depending on $z^{(i)}$. This is called the **mixture of Gaussians** model. Also, note that the $z^{(i)}$'s are **latent** random variables, meaning that they're hidden/unobserved. This is what will make our estimation problem difficult.

The parameters of our model are thus ϕ, μ and Σ. To estimate them, we can write down the likelihood of our data:

$$\begin{aligned}
\ell(\phi, \mu, \Sigma) &= \sum_{i=1}^{m} \log p(x^{(i)}; \phi, \mu, \Sigma) \\
&= \sum_{i=1}^{m} \log \sum_{z^{(i)}=1}^{k} p(x^{(i)}|z^{(i)}; \mu, \Sigma)p(z^{(i)}; \phi).
\end{aligned}$$

However, if we set to zero the derivatives of this formula with respect to the parameters and try to solve, we'll find that it is not possible to find the maximum likelihood estimates of the parameters in closed form. (Try this yourself at home.)

The random variables $z^{(i)}$ indicate which of the k Gaussians each $x^{(i)}$ had come from. Note that if we knew what the $z^{(i)}$'s were, the maximum

MLE Estimates for GMM

If we knew z

- likelihood problem would have been easy. Specifically, we could then write down the likelihood as

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log p(x^{(i)}|z^{(i)}; \mu, \Sigma) + \log p(z^{(i)}; \phi).$$

Maximizing this with respect to ϕ, μ and Σ gives the parameters:

$$\begin{aligned}
\phi_j &= \frac{1}{m}\sum_{i=1}^{m} 1\{z^{(i)} = j\}, \\
\mu_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}, \\
\Sigma_j &= \frac{\sum_{i=1}^{m} 1\{z^{(i)} = j\}(x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} 1\{z^{(i)} = j\}}.
\end{aligned}$$

But $z^{(i)}$'s are not known: so use EM

- **However, in our density estimation problem, the $z^{(i)}$'s are not known.**
- **What can we do?**
- **The EM algorithm is an iterative algorithm that has two main steps.**
  - Applied to our problem, in the E-step, it tries to "guess" the values of the $z_{(i)}$'s.
  - In the M-step, it updates the parameters of our model based on our guesses. Since in the M-step we are pretending that the guesses in the first part were correct, the maximization becomes easy. Here's the algorithm:

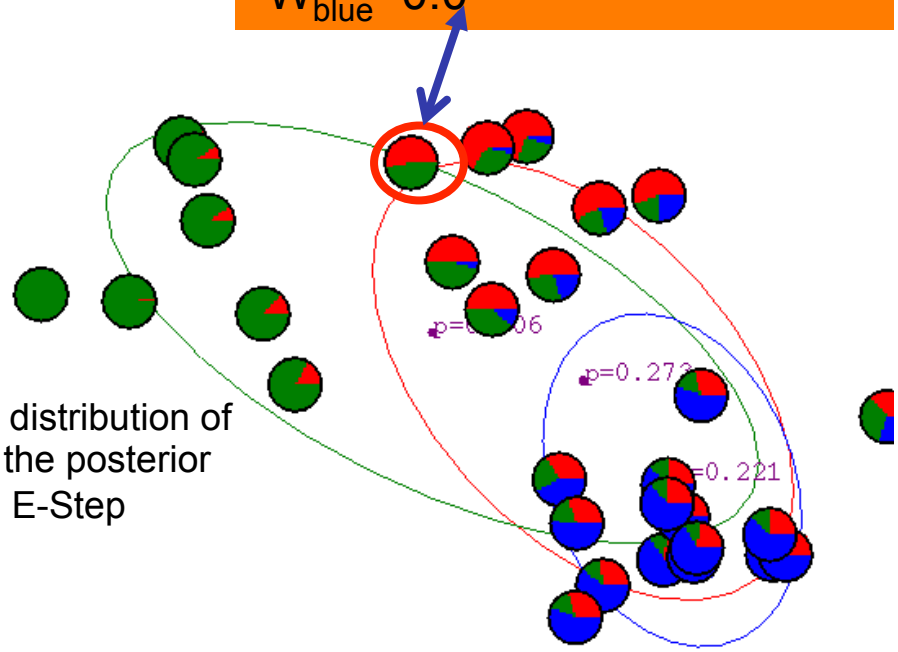
EM for GMM

Driver: Initialize μ, Σ, ϕ

- ..

E-Step

Pr(Class = Red|X) = W_{red}=0.5;
W_{green}=0.5;
W_{blue}=0.0



Repeat until convergence: {

(E-step) For each i, j, set J class; I example index

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

(M-step) Update the parameters:

phi

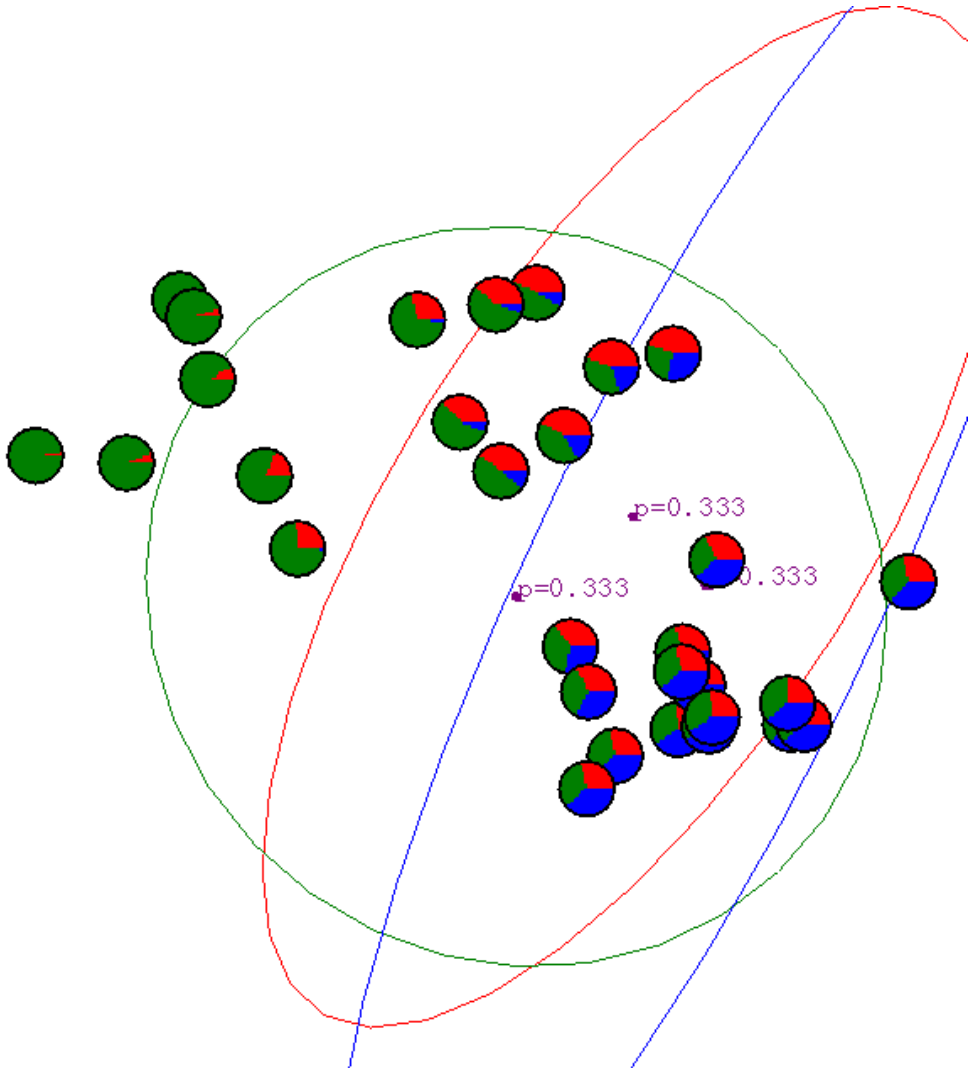
$$\phi_j := \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)},$$

Compute relative distribution of each class using the posterior probabilities from E-Step

$$\mu_j := \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}},$$

$$\Sigma_j := \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

Random init: and assignment



EM for GMM: write the mapper

Commutative and associative ops are great candidates for Mapper/combiner

Driver: Initialize μ, Σ, ϕ

- ..

Repeat until convergence: {

(E-step) For each i, j, set J class; I example index

$$w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$$

(M-step) Update the parameters:

phi

$$\phi_j := \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)},$$

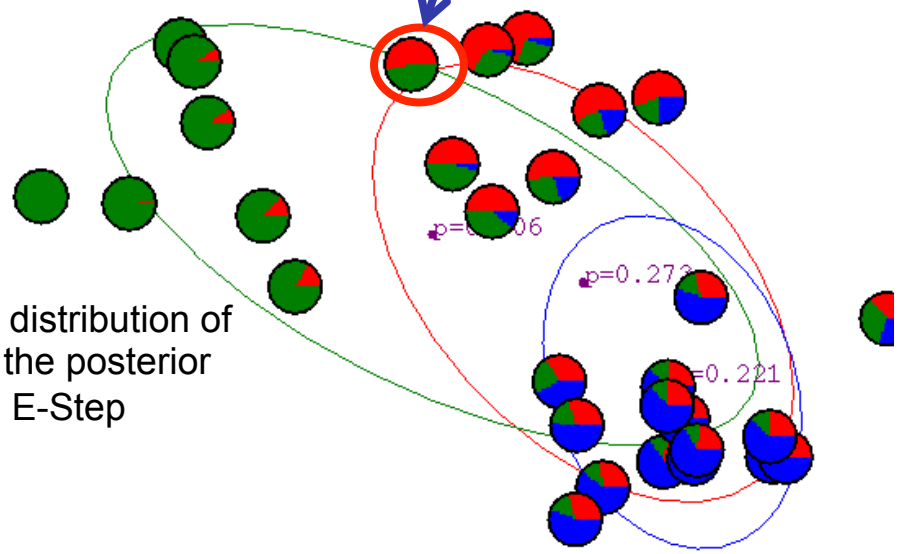
Compute relative distribution of each class using the posterior probabilities from E-Step

$$\mu_j := \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}},$$

$$\Sigma_j := \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

E-Step

Pr(Class = Red|X) = W_{red}=0.5;
W_{green}=0.5;
W_{blue}=0.0



EM for GMM: write the REDUCER

Commutative and associative, so are great candidates for Mapper/combiner

Driver: Initialize μ_j, Σ_j, ϕ_j

- ..

Repeat until convergence: {

(E-step) For each j, j, set J class, I example index

$p_j := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$

(M-step) Update the parameters:

Partition key for reducer?

Partial sums for

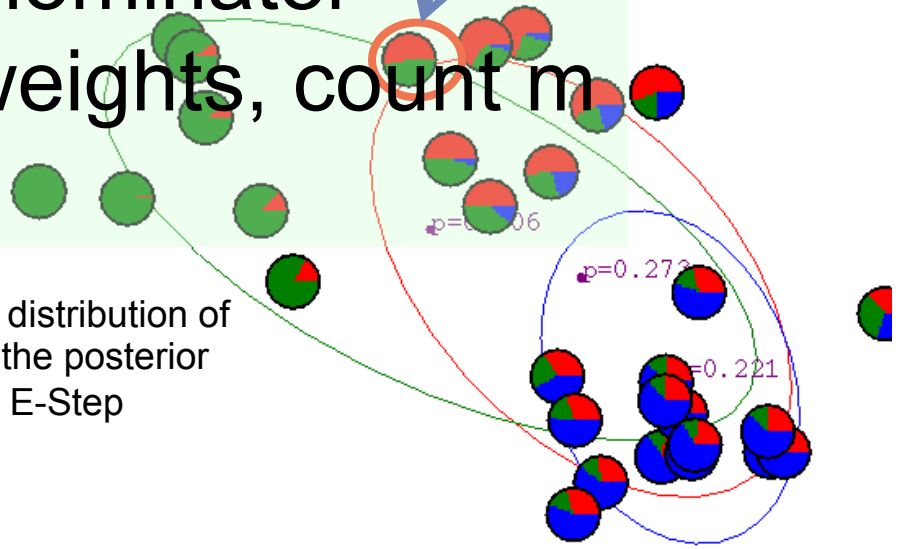
μ, numerator, denominator

Σ, numerator, denominator

ϕ partial sum of weights, count m

E-Step

Pr(Class = Red|X) = W_{red}=0.5;
W_{green}=0.5;
W_{blue}=0.0



phi $\phi_j := \frac{1}{m}\sum_{i=1}^{m} w_j^{(i)}$,

Compute relative distribution of each class using the posterior probabilities from E-Step

$$\mu_j := \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}},$$

$$\Sigma_j := \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$$

In the E-step, we calculate the posterior probability of our parameters the $z^{(i)}$'s, given the $x^{(i)}$ and using the current setting of our parameters. I.e., using Bayes rule, we obtain:

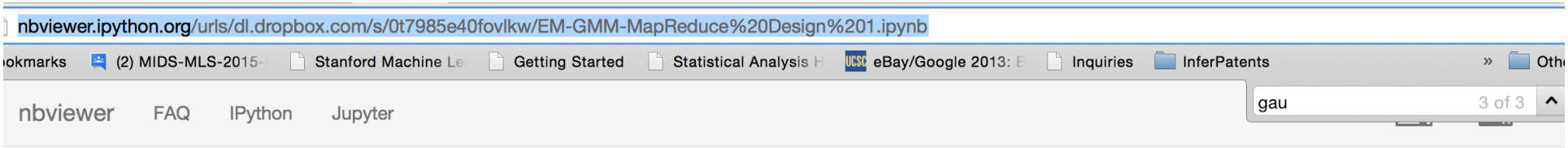
- $$p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma) = \frac{p(x^{(i)} | z^{(i)} = j; \mu, \Sigma) p(z^{(i)} = j; \phi)}{\sum_{l=1}^{k} p(x^{(i)} | z^{(i)} = l; \mu, \Sigma) p(z^{(i)} = l; \phi)}$$

Here, $p(x^{(i)} | z^{(i)} = j; \mu, \Sigma)$ is given by evaluating the density of a Gaussian with mean μ_j and covariance Σ_j at $x^{(i)}$; $p(z^{(i)} = j; \phi)$ is given by ϕ_j, and so on. The values $w_j^{(i)}$ calculated in the E-step represent our "soft" guesses[2] for the values of $z^{(i)}$.

Also, you should contrast the updates in the M-step with the formulas we had when the $z^{(i)}$'s were known exactly. They are identical, except that instead of the indicator functions "$1\{z^{(i)} = j\}$" indicating from which Gaussian each datapoint had come, we now instead have the $w_j^{(i)}$'s.

The EM-algorithm is also reminiscent of the K-means clustering algorithm, except that instead of the "hard" cluster assignments $c(i)$, we instead have the "soft" assignments $w_j^{(i)}$. Similar to K-means, it is also susceptible to local optima, so reinitializing at several different initial parameters may be a good idea.

EM Algorithm for GMM



Introduction

This is a map-reduce version of expectation maximization algo for a mixture of Gaussians model. There are two mrJob MR packages, mr_GMixEmIterate and mr_GMixEmInitialize. The driver calls the mrJob packages and manages the iteration.

E Step: Given mean vector and covariance matrix, calculate the probability of that each data point belongs to a class

$P(Cluster_k | X^i; \theta)$

$$p(\omega_k|\boldsymbol{x}^{(i)},\theta) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j \mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}$$

Estimate class assignments (responsibities) or weights Use the current model to estimate the class assignment

M Step: Given probabilities, update mean and covariance

$Centroid_k$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|\boldsymbol{x}^{(i)},\theta)\boldsymbol{x}^{(i)}$$

Centroid is just a weighted sum of soft assigned examples

$Covariance_k$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|\boldsymbol{x}^{(i)},\theta)(\boldsymbol{x}^{(i)}-\hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}^{(i)}-\hat{\boldsymbol{\mu}}_k)^T$$

Weighted covariance

$Prior_k$

$$\hat{\pi}_k = \frac{n_k}{n} \text{ where } n_k = \sum_{i=1}^{n} p(\omega_k|\boldsymbol{x}^{(i)},\theta)$$

Class prior is based on the sum of the partial weights

The EM Algorithm:

Two-Component Mixture Model

- ..

The left panel of Figure 1 shows a histogram of the 20 fictitious data points in Table 1.

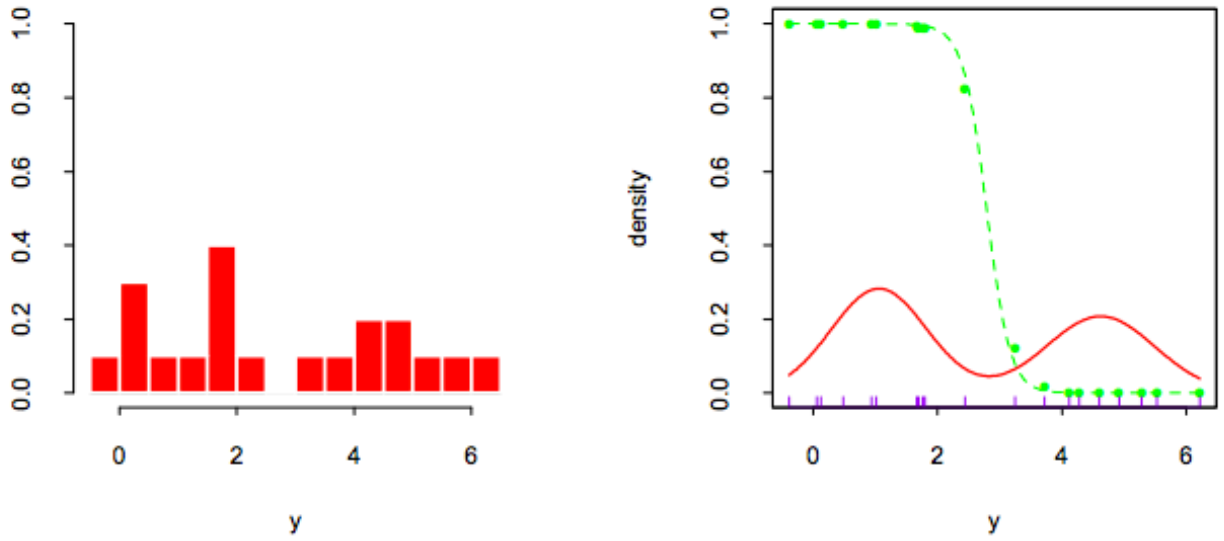


Figure 1: *Mixture example. Left panel: histogram of data. Right panel: maximum likelihood fit of Gaussian densities (solid red) and responsibility (dotted green) of the left component density for observation y, as a function of y.*

Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a mixture of Bernoulli distributions**

MRJob Notebook for EM Algo: GMM

- http://nbviewer.ipython.org/urls/dl.dropbox.com/s/0t7985e40fovlkw/EM-GMM-MapReduce%20Design%201.ipynb
- **Local copy**
  - https://www.dropbox.com/s/1y584zzplc6d4jq/EM-GMM-MapReduce-2015-09-30-Presented-in-Week5.ipynb?dl=0

Notebooks

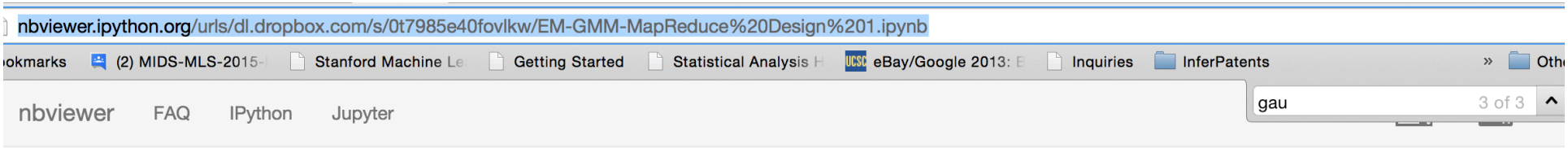
GMM:

- http://nbviewer.ipython.org/urls/dl.dropbox.com/s/0t7985e40fovlkw/EM-GMM-MapReduce%20Design%201.ipynb

Kmeans:

- http://nbviewer.ipython.org/urls/dl.dropbox.com/s/yk3mldx731q157k/EM-Kmeans-MapReduce%20Design%202.ipynb

EM Algorithm for GMM



Introduction

This is a map-reduce version of expectation maximization algo for a mixture of Gaussians model. There are two mrJob MR packages, mr_GMixEmIterate and mr_GMixEmInitialize. The driver calls the mrJob packages and manages the iteration.

E Step: Given mean vector and covariance matrix, calculate the probability of that each data point belongs to a class

$P(Cluster_k | X^i; \theta)$

$$p(\omega_k|\boldsymbol{x}^{(i)},\theta) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j \mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}$$

Estimate class assignments (responsibities) or weights Use the current model to estimate the class assignment

M Step: Given probabilities, update mean and covariance

$Centroid_k$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|\boldsymbol{x}^{(i)},\theta)\boldsymbol{x}^{(i)}$$

Centroid is just a weighted sum of soft assigned examples

$Covariance_k$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|\boldsymbol{x}^{(i)},\theta)(\boldsymbol{x}^{(i)}-\hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}^{(i)}-\hat{\boldsymbol{\mu}}_k)^T$$

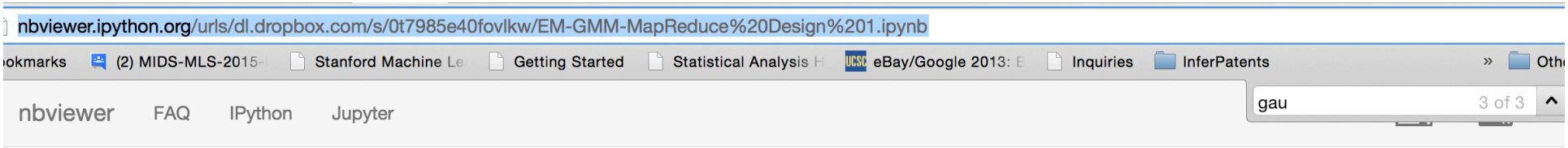
Weighted covariance

$Prior_k$

$$\hat{\pi}_k = \frac{n_k}{n} \text{ where } n_k = \sum_{i=1}^{n} p(\omega_k|\boldsymbol{x}^{(i)},\theta)$$

Class prior is based on the sum of the partial weights

EM Algorithm for GMM



Introduction

This is a map-reduce version of expectation maximization algo for a mixture of Gaussians model. There are two mrJob MR packages, mr_GMixEmIterate and mr_GMixEmInitialize. The driver calls the mrJob packages and manages the iteration.

E Step: Given mean vector and covariance matrix, calculate the probability of that each data point belongs to a class

$P(Cluster_k | X^i; \theta)$

$$p(\omega_k|\boldsymbol{x}^{(i)},\theta) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K}\pi_j \mathcal{N}(\boldsymbol{x}^{(i)}|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j)}$$

M Step: Given probabilities, update mean and covariance

$Centroid_k$

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|\boldsymbol{x}^{(i)},\theta)\boldsymbol{x}^{(i)}$$

$Covariance_k$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|\boldsymbol{x}^{(i)},\theta)(\boldsymbol{x}^{(i)}-\hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}^{(i)}-\hat{\boldsymbol{\mu}}_k)^T$$

$Prior_k$

$$\hat{\pi}_k = \frac{n_k}{n} \text{ where } n_k = \sum_{i=1}^{n} p(\omega_k|\boldsymbol{x}^{(i)},\theta)$$

Gaussian

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right).$$

- ..

In the equation above, "$|\Sigma|$" denotes the determinant of the matrix Σ.

```
%%writefile mr_GMixEmIterate.py
from mrjob.job import MRJob

from math import sqrt, exp, pow,pi
from numpy import zeros, shape, random, array, zeros_like, dot, linalg
import json

def gauss(x, mu, P_1):
    xtemp = x - mu
    n = len(x)
    p = exp(- 0.5*dot(xtemp,dot(P_1,xtemp)))
    detP = 1/linalg.det(P_1)
    p = p/(pow(2.0*pi,n/2.0)*sqrt(detP))
    return p
```

P_1 is the inverted Sigma has already been inverted in the reducer to prevent redoing it for each example. The covariance matrix needs to be inverted for Gaussian

Init EM: In-memory mapper for running totals of centroids, covariance and priors

```
class MrGMixEm(MRJob):
    DEFAULT_PROTOCOL = 'json'

    def __init__(self, *args, **kwargs):
        super(MrGMixEm, self).__init__(*args, **kwargs)

        fullPath = self.options.pathName + 'intermediateResults.txt'
        fileIn = open(fullPath)
        inputJson = fileIn.read()
        fileIn.close()
        inputList = json.loads(inputJson)
        temp = inputList[0]
        self.phi = array(temp)              #prior class probabilities
        temp = inputList[1]
        self.means = array(temp)            #current means list
        temp = inputList[2]
        self.cov_1 = array(temp)            #inverse covariance matrices for w, calc.
        #accumulate partial sums
        #sum of weights - by cluster
        self.new_phi = zeros_like(self.phi)         #partial weighted sum of weights
        self.new_means = zeros_like(self.means)
        self.new_cov = zeros_like(self.cov_1)

        self.numMappers = 1                 #number of mappers
        self.count = 0                      #passes through mapper
```

```
    def mapper(self, key, val):
        #accumulate partial sums for each mapper
        xList = json.loads(val)
        x = array(xList)
        wtVect = zeros_like(self.phi)
        for i in range(self.options.k):
            wtVect[i] = self.phi[i]*gauss(x,self.means[i],self.cov_1[i])
        wtSum = sum(wtVect)
        wtVect = wtVect/wtSum
        #accumulate to update est of probability densities.
        #increment count
        self.count += 1
        #accumulate weights for phi est
        self.new_phi = self.new_phi + wtVect
        for i in range(self.options.k):
            #accumulate weighted x's for mean calc
            self.new_means[i] = self.new_means[i] + wtVect[i]*x
            #accumulate weighted squares for cov estimate
            xmm = x - self.means[i]
            covInc = zeros_like(self.new_cov[i])

            for l in range(len(xmm)):
                for m in range(len(xmm)):
                    covInc[l][m] = xmm[l]*xmm[m]
            self.new_cov[i] = self.new_cov[i] + wtVect[i]*covInc
        #dummy yield - real output passes to mapper_final in self


    def mapper_final(self):

        out = [self.count, (self.new_phi).tolist(), (self.new_means).tolist(), (self.new_cov).toli
st()]
        jOut = json.dumps(out)

        yield 1,jOut
```

Estimate class assignments (responsibities) or weights
Use the current model to estimate class assignments

class assignments

For each cluster

wtVect is an array of priors

For each cluster

For each cluster K
update the centroid running to

Centroid$_k$ $\hat{\mu}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|x^{(i)},\theta)x^{(i)}$

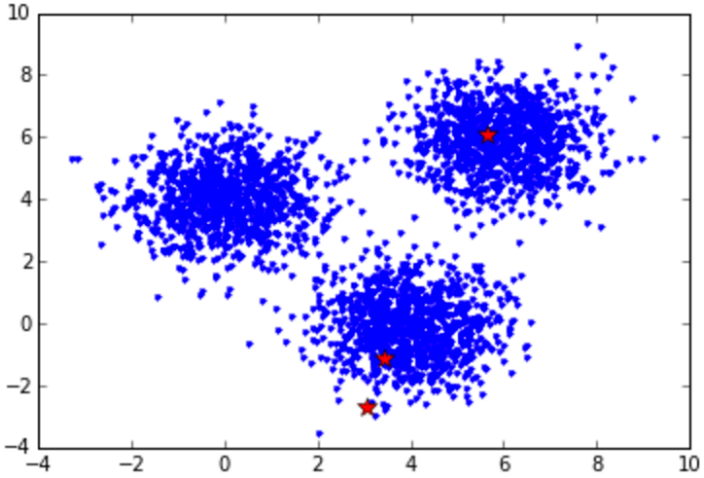
Covariance$_k$ $\hat{\Sigma}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|x^{(i)},\theta)(x^{(i)}-\hat{\mu}_k)(x^{(i)}-\hat{\mu}_k)^T$

Prior$_k$ $\hat{\pi}_k = \frac{n_k}{n}$ where $n_k = \sum_{i=1}^{n} p(\omega_k|x^{(i)},\theta)$

Up covariance running total

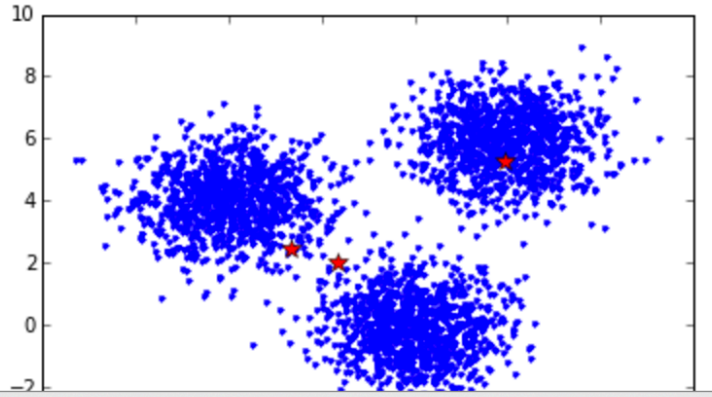
Mapper Final: count, yield running totals for priors, centroids, covariance

```
Iteration0
[[3.4409365422359786, -1.122508052281444], [3.0463635118491736, -2.6738446589131244], [5.642450693187089, 6.100299
46467955]]
```



http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/0t7985e40fovlkw/EM-GMM-MapReduce%20Design%201.ipynb

```
Iteration1
[[2.3403393295629256, 1.999145534329882], [1.3615073875003847, 2.4641575753284246], [5.941841130708148, 5.27309144
8075368]]
```



```
    def reducer(self, key, xs): A single F
        #accumulate partial sums
        first = True
        #accumulate partial sums
        #xs us a list of paritial stats, including count, phi, mean, and covariance.
        #Each stats is k-length array, storing info for k components
        for val in xs:
            if first:
                temp = json.loads(val)
                #totCount, totPhi, totMeans, and totCov are all arrays
                totCount = temp[0]
                totPhi = array(temp[1])
                totMeans = array(temp[2])
                totCov = array(temp[3])
                first = False
            else:
                temp = json.loads(val)
                #cumulative sum of four arrays
                totCount = totCount + temp[0]
                totPhi = totPhi + array(temp[1])
                totMeans = totMeans + array(temp[2])
                totCov = totCov + array(temp[3])
        #finish calculation of new probability parameters. array divided by array
        newPhi = totPhi/totCount
        #initialize these to something handy to get the right size arrays
        newMeans = totMeans
        newCov_1 = totCov
        for i in range(self.options.k):
            newMeans[i,:] = totMeans[i,:]/totPhi[i]
            tempCov = totCov[i,:,:]/totPhi[i]
            #almost done.  just need to invert the cov matrix.  invert here to save doing a matrix
inversion

            #with every input data point.
            newCov_1[i,:,:] = linalg.inv(tempCov)

        outputList = [newPhi.tolist(), newMeans.tolist(), newCov_1.tolist()]
        jsonOut = json.dumps(outputList)

        #write new parameters to file
        fullPath = self.options.pathName + 'intermediateResults.txt'
        fileOut = open(fullPath,'w')
        fileOut.write(jsonOut)
        fileOut.close()
```

Mapper Final: count, yield running totals for priors, centroids

Reducer: count, yield running totals for priors, centroids, covariance

```
(self.new_phi).tolist(), (self.new_means).tolist(), (self.new_c
```

Centroid$_k$

$$\hat{\mu}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|x^{(i)},\theta)x^{(i)}$$

Covariance$_k$

$$\hat{\Sigma}_k = \frac{1}{n_k}\sum_{i=1}^{n} p(\omega_k|x^{(i)},\theta)(x^{(i)}-\hat{\mu}_k)(x^{(i)}-\hat{\mu}_k)^T$$

Prior$_k$

$$\hat{\pi}_k = \frac{n_k}{n} \text{ where } n_k = \sum_{i=1}^{n} p(\omega_k|x^{(i)},\theta)$$

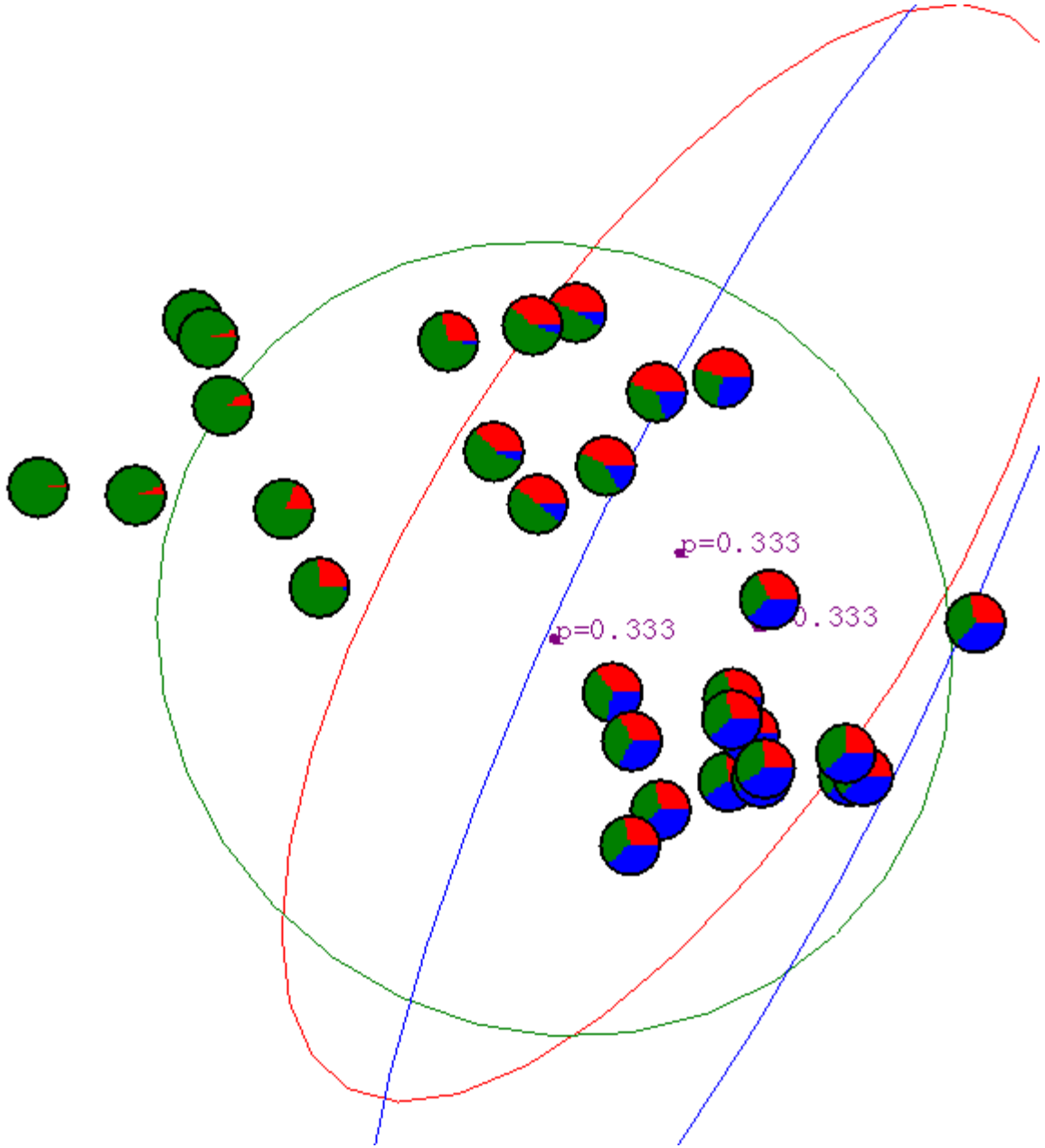
Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a mixture of Bernoulli distributions**

Animation: EM algorithm for a GMM

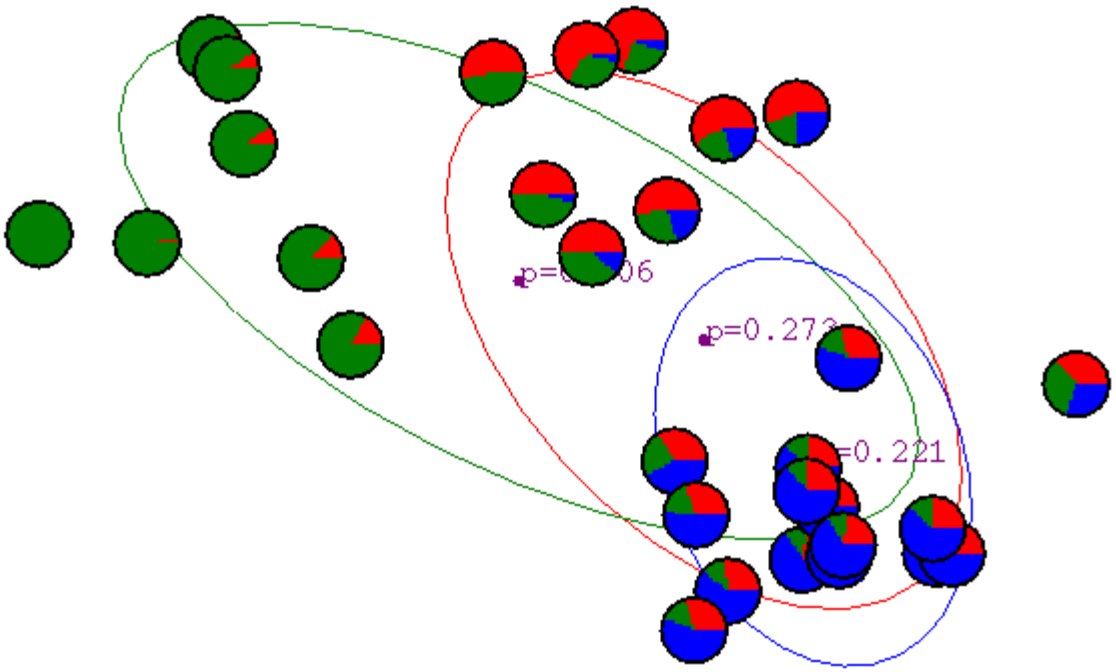
- **Running the EM algorithm to derive the Gaussian Mixture Model (GMM) from a set of training data**

Gaussian Mixture Example: Start

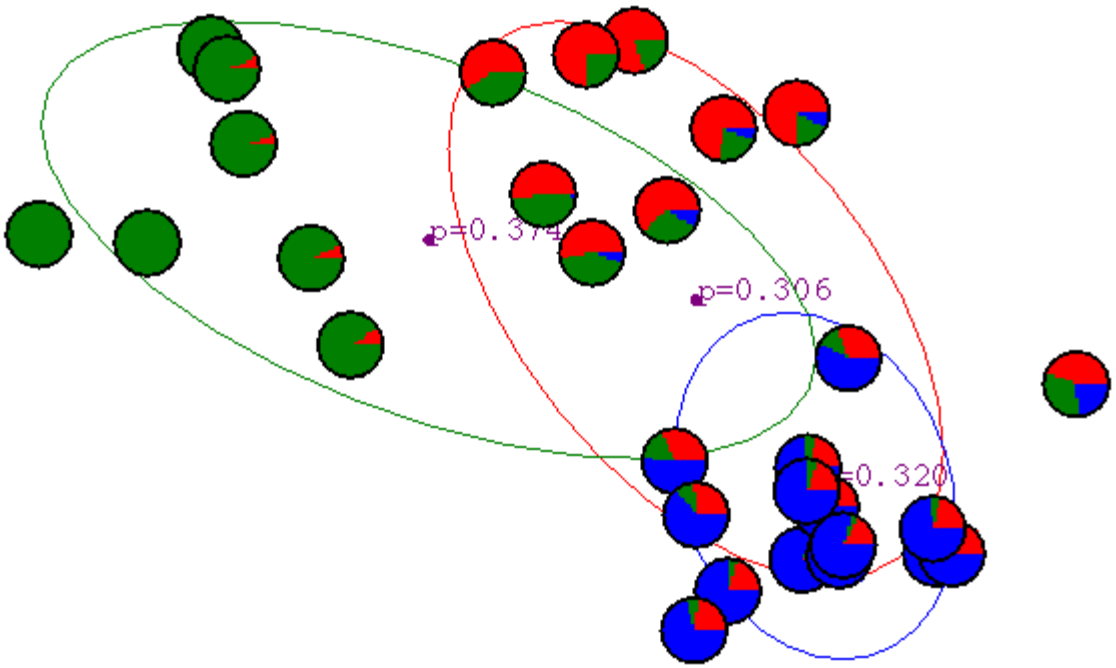


Based on slides from Rong Jin, Andrew Blake, Bill Freeman, Ya Chang and Mathias Kölsch and Andrew W. Moore

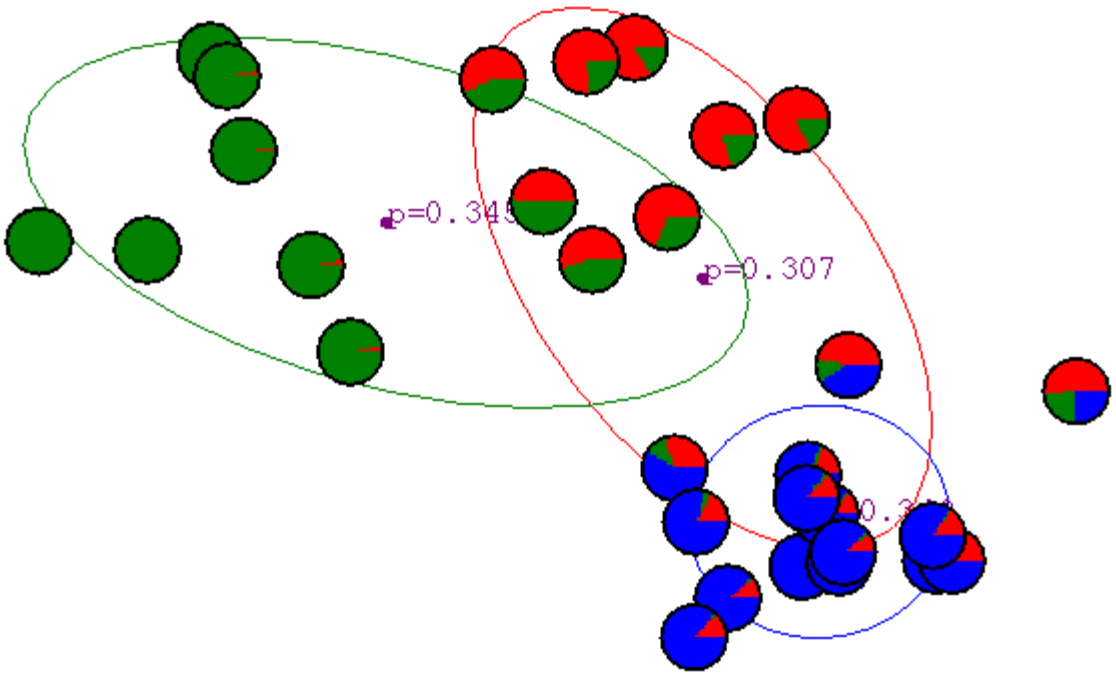
After first iteration



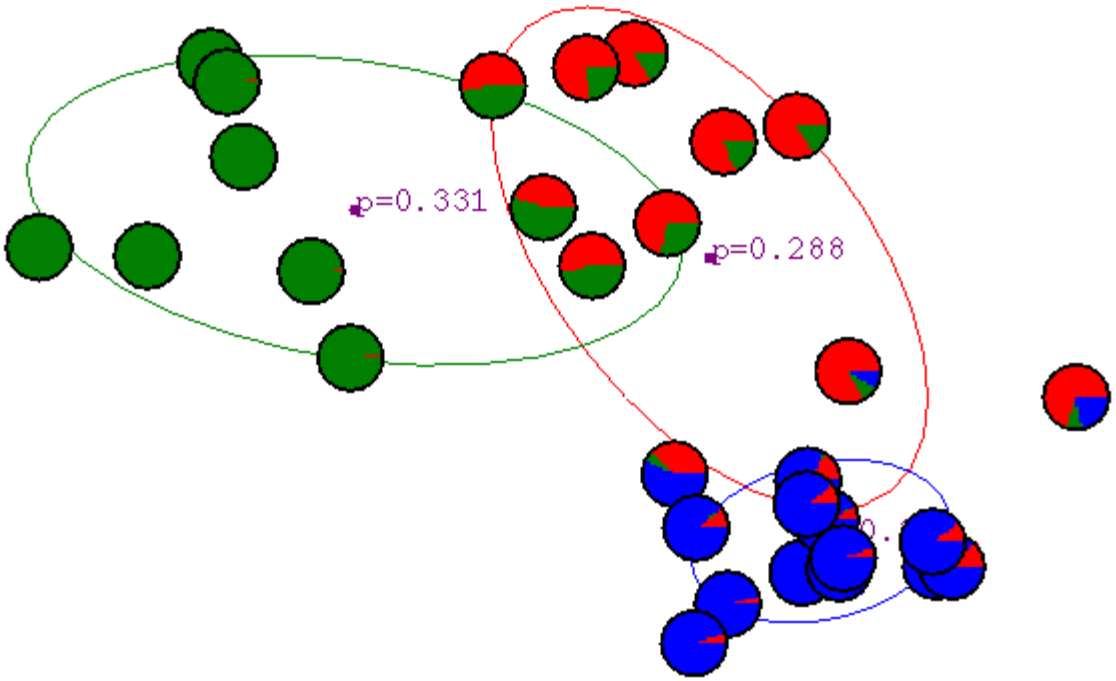
After 2nd iteration



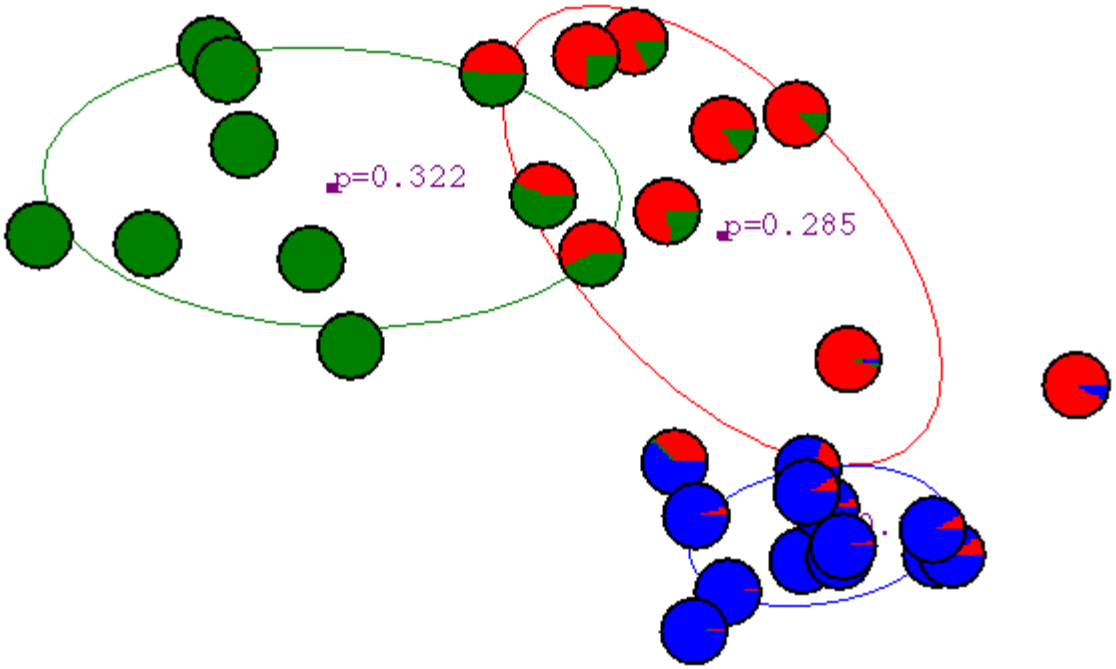
After 3rd iteration



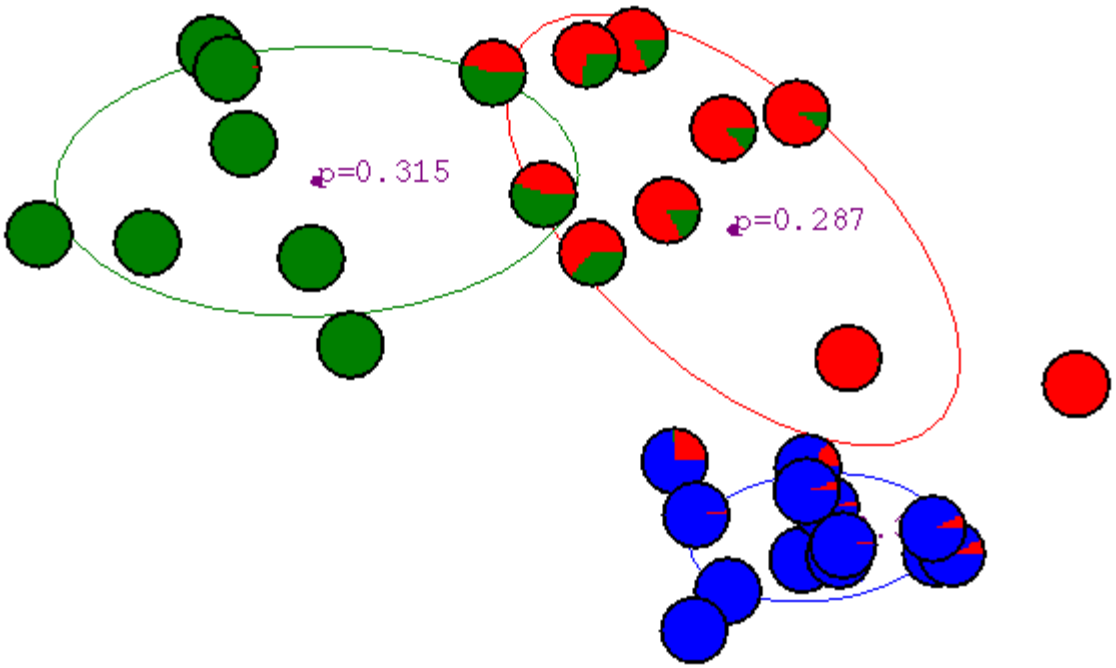
After 4th iteration



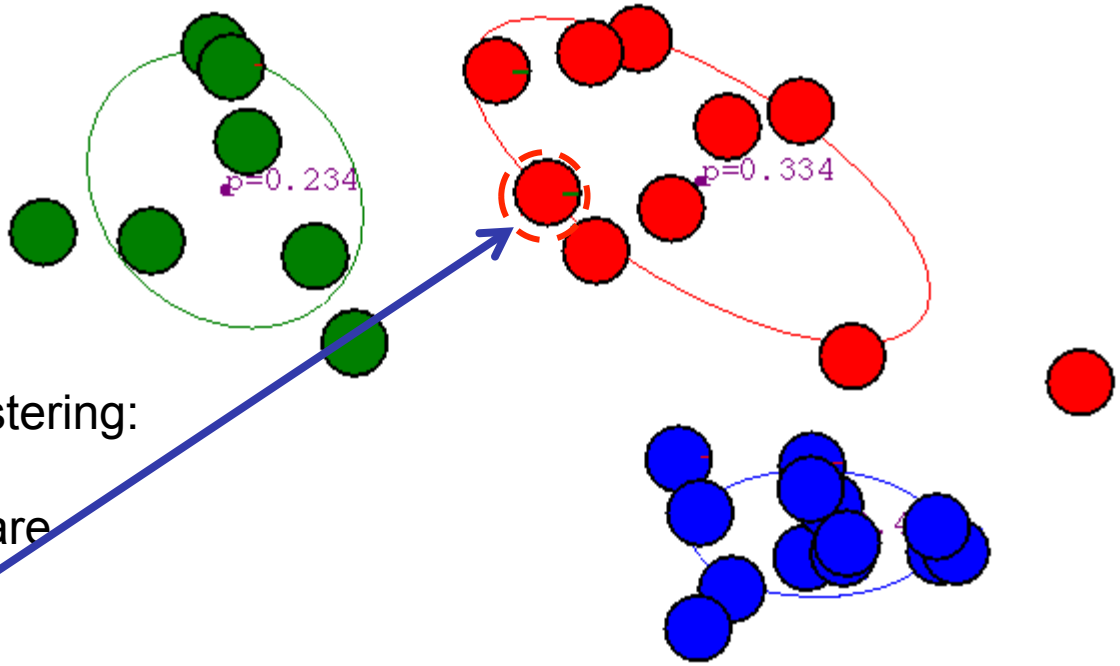
After 5th iteration



After 6th iteration



After 20th iteration

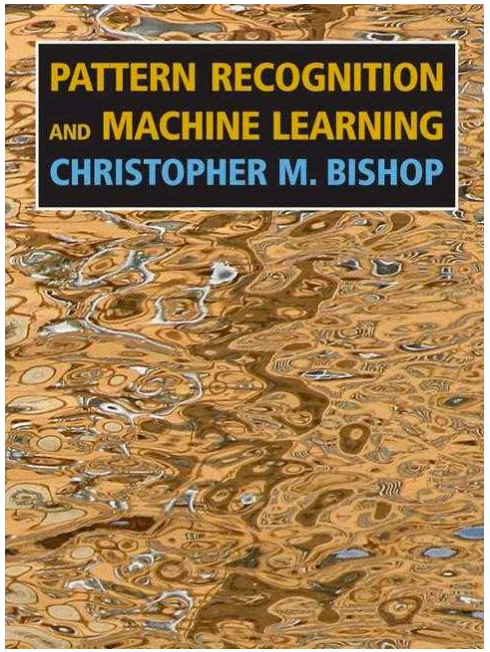


After 20 iterations of clustering: EM seems to yield hard assignments. But there are examples here with soft assignments. E.g.,

Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a mixture of Bernoulli distributions**

Plate-based presentation of K-Means and EM algorithms



9.1 K-means Clustering (2/3)

- Two-stage optimization
  - In the 1st stage: minimizing J with respect to the r_{nk}, keeping the μ_k fixed

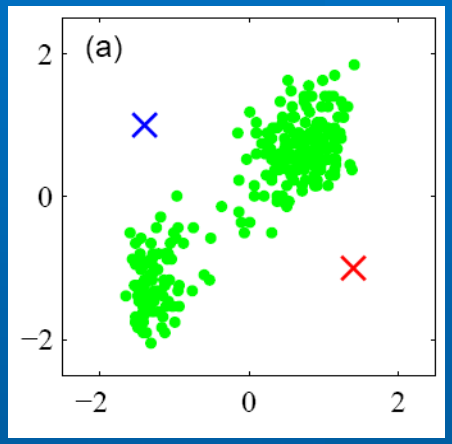
$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \| \boldsymbol{x}_n - \mu_j \|^2 \\ 0 & otherwise \end{cases}$$

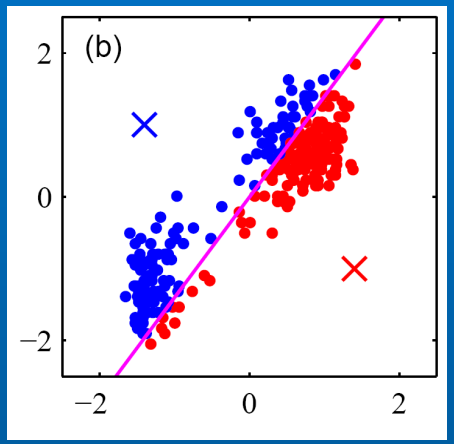
  - In the 2nd stage: minimizing J with respect to the μ_k, keeping r_{nk} fixed

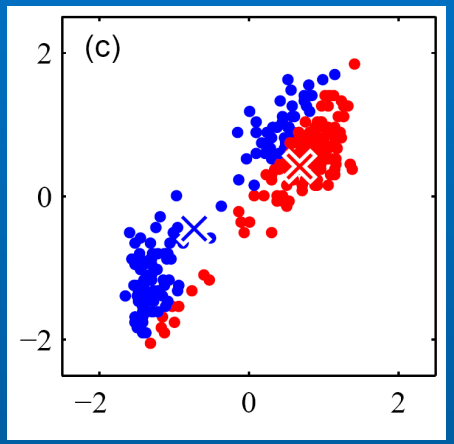
$$\mu_k = \frac{\sum_n r_{nk} \boldsymbol{x}_n}{\sum_n r_{nk}}$$

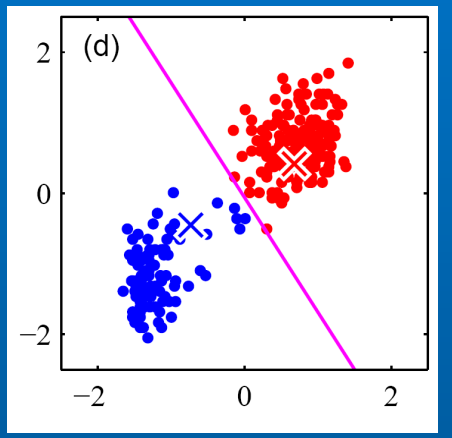
The mean of all of the data points assigned to cluster k

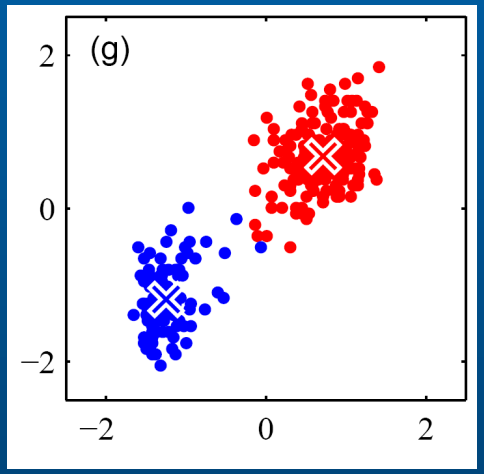
9.1 K-means Clustering (3/3)

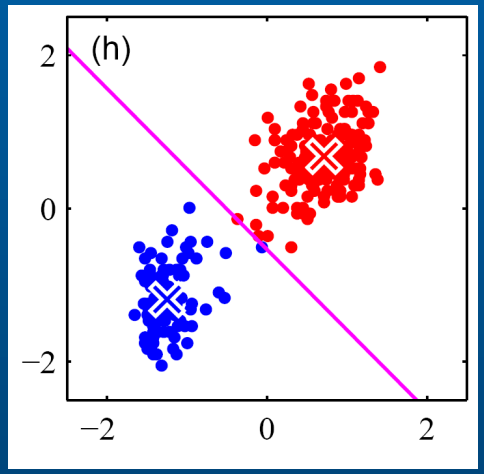












9.2 Mixtures of Gaussians (1/3)

A formulation of Gaussian mixtures in terms of discrete *latent* variables

- Gaussian mixture distribution can be written as a linear superposition of Gaussian

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathrm{N}(\boldsymbol{x}|\mu_k, \Sigma_k) \quad \ldots(*)$$

- An equivalent formulation of the Gaussian mixture inv olving an explicit latent variable
  - Graphical representation of a mixture model
    - **A** binary random variable **z** having a 1-of-K representation

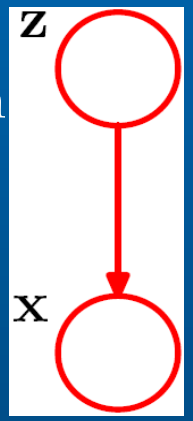
$$p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k=1}^{K} N(\boldsymbol{x}|\mu_k, \Sigma_k)^{z_k}$$

$$p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z})$$

$$p(z_k = 1) = \pi_k \qquad \sum_{k=1}^{K} \pi_k = 1$$

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{z}} p(\boldsymbol{z})p(\boldsymbol{x}|\boldsymbol{z}) = \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}|\mu_k, \Sigma_k)$$



    - The marginal distribution of **x** is a Gaussian mixture of the form (*) (→ for every observed data point $\mathbf{x}_n$, there is a corresponding latent variable $\mathbf{z}_n$)

$$p(\boldsymbol{x}) = \Sigma_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z})$$

9.2 Mixtures of Gaussians (2/3)

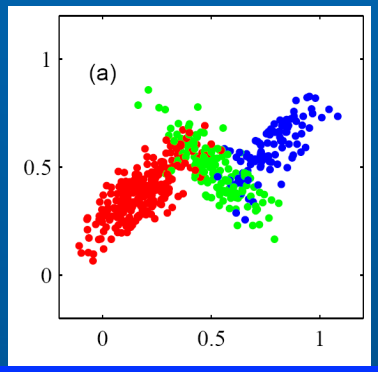
$$\gamma(z_k) \equiv p(z_k = 1 | \boldsymbol{x}) = \frac{p(z_k = 1) p(\boldsymbol{x} | z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1) p(\boldsymbol{x} | z_j = 1)}$$

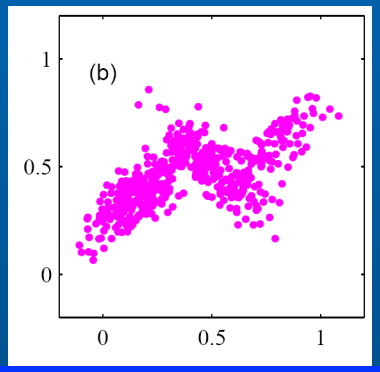
$$= \frac{\pi_k N(\boldsymbol{x} | \mu_k, \sum_k)}{\sum_{j=1}^{K} \pi_j N(\boldsymbol{x} | \mu_j, \sum_j)}$$

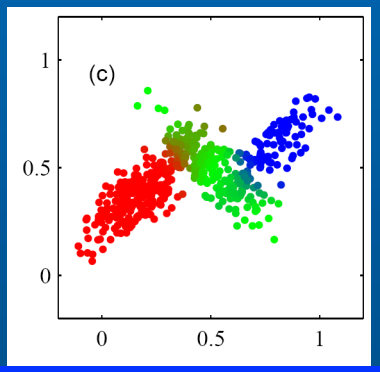
- $\gamma(z_k)$ can also be viewed as the responsibility that co mponent k takes for explaining the observation $\mathbf{x}$

9.2 Mixtures of Gaussians (3/3)

- Generating random samples distributed according to the Gaus sian mixture model
  - Generating a value for **z**, which denoted as $\hat{z}$ from the margina l distribution p(**z**) and then generate a value for **x** from the condi tional distribution $p(\boldsymbol{x}|\hat{z})$



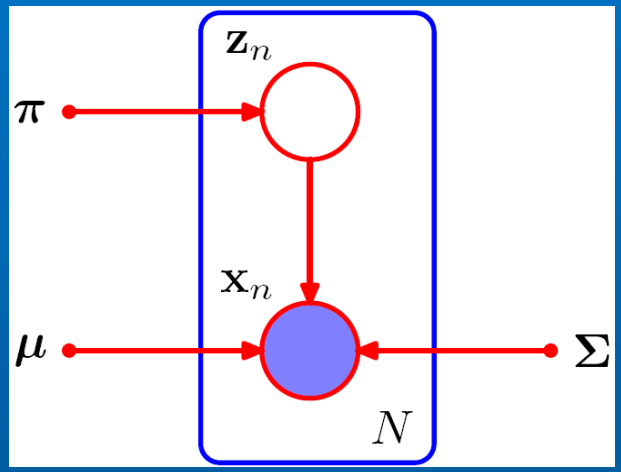




a. The three states of **z**, corresponding to the three components of the mixture, are depic ted in red, green, blue
b. The corresponding samples from the marginal distribution p(**x**)
c. The same samples in which the colors represent the value of the responsibilities $\gamma(z_{nk})$ associated with data point
   - ✓ Illustrating the responsibilities by evaluating the posterior probability for each compo nent in the mixture distribution which this data set was generated

9.2.1 Maximum likelihood (1/3)

- Graphical representation of a Gaussian mixture model for a set of N i.i.d. data points $\{x_n\}$, with corresponding latent points $\{z_n\}$



- The log of the likelihood function

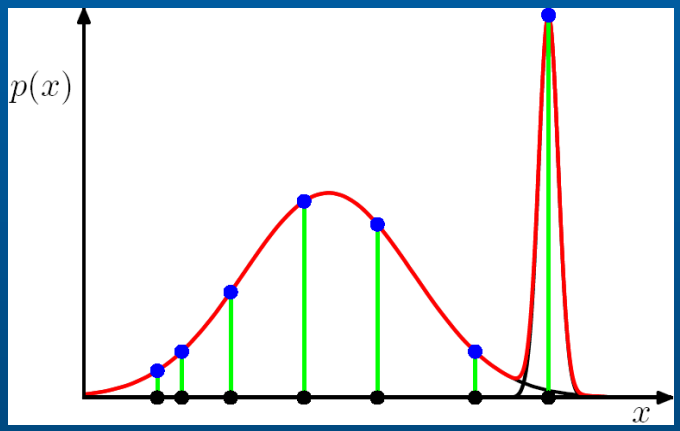
$$\ln p(\boldsymbol{X}|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}_n|\mu_k,\Sigma_k) \right\} \quad \ldots.(*1)$$

9.2.1 Maximum likelihood (2/3)

- For simplicity, consider a Gaussian mixture whose components hav e covariance matrices given by $\Sigma_k = \sigma_k^2 I$
  - Suppose that one of the components of the mixture model has its mean μ_j exactly equal to one of the data points so that $\mu_j = \mathbf{x}_n$
  - This data point will contribute a term in the likelihood function of the f orm

$$N(\boldsymbol{x}_n | \boldsymbol{x}_n, \sigma_j^2 I) = \frac{1}{(2\pi)^{1/2}} \frac{1}{\sigma_j}$$

  - Once there are at least two components in the mixture, one of the components can have a finite variance and therefore assign finite probability to all of the data points while the other component can shrink onto one specific data point and thereby contribute an ever increasing additive value to the log likelihood → over-fitting problem



Overfittings when one of the Gaussian components collapses onto a specific data point

$D = \{x_1, x_2, x_3\},\ p(x \mid z_1 = 1) = N(x \mid x_1, \sigma_1),\ p(x \mid z_2 = 1) = N(x \mid \mu_2, \sigma_2)$

(single Gaussian case)

$$L = \left(\frac{1}{\sigma_1}\right)^3 \exp\left(-\frac{(x_1 - x_1)^2 + (x_2 - x_1)^2 + (x_3 - x_1)^2}{2\sigma_1^2}\right)$$

$$= \left(\frac{1}{\sigma_1}\right)^2 \exp\left(-\frac{(x_2 - x_1)^2 + (x_3 - x_1)^2}{2\sigma_1^2}\right) \cdot \left(\frac{1}{\sigma_1}\right)$$

$$\lim_{\sigma_1 \to 0} L = 0$$

(mixture case)

$$L = \left(\frac{1}{2\sigma_1}\exp\left(-\frac{(x_1 - x_1)^2}{2\sigma_1^2}\right) + \frac{1}{2\sigma_2}\exp\left(-\frac{(x_1 - \mu_2)^2}{2\sigma_2^2}\right)\right) \cdot$$

$$\left(\frac{1}{2\sigma_1}\exp\left(-\frac{(x_2 - x_1)^2}{2\sigma_1^2}\right) + \frac{1}{2\sigma_2}\exp\left(-\frac{(x_2 - \mu_2)^2}{2\sigma_2^2}\right)\right) \cdot \left(\frac{1}{2\sigma_1}\exp\left(-\frac{(x_3 - x_1)^2}{2\sigma_1^2}\right) + \frac{1}{2\sigma_2}\exp\left(-\frac{(x_3 - \mu_2)^2}{2\sigma_2^2}\right)\right)$$

$$\lim_{\sigma_1 \to 0} L = \infty$$

9.2.1 Maximum likelihood (3/3)

- Over-fitting problem
  - Example of the over-fitting in a maximum likelihood approach
  - This problem does not occur in the case of Bayesian approach
  - In applying maximum likelihood to a Gaussian mixture models , there should be steps to avoid finding such pathological soluti ons and instead seek local minima of the likelihood function tha t are well behaved
- *Identifiability* problem
  - A K-component mixture will have a total of K! equivalent soluti ons corresponding to the K! ways of assigning K sets of paramet ers to K components
- Difficulty of maximizing the log likelihood function → the presence of the summation over k that appears inside the logarithm gives no closed form solution as in the single case

9.2.2 EM for Gaussian mixtures (1/4)

I. Assign some initial values for the means, covariances, and mixing coefficients

II. *Expectation* or E step
- Using the current value for the parameters to evaluate the posterior probabilities or *responsibilities*

III. *Maximization* or M step
- Using the result of II to re-estimate the means, covariances, and mixing coefficients

❖ It is common to run the K-means algorithm in order to find a suitable initial values
- The covariance matrices → the sample covariances of the clusters found by the K-means algorithm
- Mixing coefficients → the fractions of data points assigned to the respective clusters

9.2.2 EM for Gaussian mixtures (2/4)

- Given a Gaussian mixture model, the goal is to maximize the likeli hood function with respect to the parameters
  1. Initialize the means μ_k, covariance Σ_k and mixing coefficients π_k
  2. E step

$$\gamma(z_{nk}) = \frac{\pi_k N(\boldsymbol{x}_n|\mu_k,\Sigma_k)}{\sum_{j=1}^{K}\pi_j N(\boldsymbol{x}_n|\mu_j,\Sigma_j)}$$

  3. M step

$$\mu_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\,\boldsymbol{x}_n$$

$$\Sigma_k^{new} = \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\boldsymbol{x}_n-\mu_k^{new})(\boldsymbol{x}_n-\mu_k^{new})^T$$

$$\pi_k^{new} = \frac{N_k}{N} \qquad N_k = \sum_{n=1}^{N}\gamma(z_{nk})$$

  4. Evaluate the log likelihood

$$\ln p(\boldsymbol{X}|\mu,\Sigma,\Pi) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k N(\boldsymbol{x}_n|\mu_k,\Sigma_k)\right\}$$

9.2.2 EM for Gaussian mixtures (3/4)

$$\ln p(\boldsymbol{X}|\pi,\mu,\Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}_n|\mu_k,\Sigma_k) \right\} \quad \ldots(*2)$$

- Setting the derivatives of (*2) with respect to the means of the Gaussian components to zero →

$$0 = -\sum_{n=1}^{N} \frac{\pi_k n(\boldsymbol{x}_n|\mu_k,\Sigma_k)}{\Sigma_j \pi_j N(\boldsymbol{x}_n|\mu_j,\Sigma_j)} \Sigma_k(\boldsymbol{x}_n - \mu_k)$$

$$\frac{\pi_k n(\boldsymbol{x}_n|\mu_k,\Sigma_k)}{\Sigma_j \pi_j N(\boldsymbol{x}_n|\mu_j,\Sigma_j)}$$ ←Responsibility $\gamma(z_{nk})$

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\, \boldsymbol{x}_n$$

A weighted mean of all of the points in the data set

- Setting the derivatives of (*2) with respect to the covariance o f the Gaussian components to zero →

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\boldsymbol{x}_n - \mu_k^{new})(\boldsymbol{x}_n - \mu_k^{new})^T$$

  - Each data point weighted by the corresponding posterior probability
  - The denominator given by the effective # of points associated with t he corresponding component

Soft Flat Clustering via EM: Outline

- **MLE, MAP and Bayes Optimal Classifier**
- **Flat Clustering (as opposed to hierarchical)**
  - Soft versus Hard
- **Maximum Likelihood Estimation (MLE)**
  - Bernoulli distributions
  - Gaussian Distributions
- **MLE for Gaussian Mixture Models**
- **EM for Gaussian Mixture Models**
  - MRJob Notebook for EM Algorithm for GMMs
  - Animation: EM algorithm for a GMM
  - Plate-based presentation of K-Means and EM algorithms
- **EM Algorithm for a Bernoulli Mixture Models**

ML: Select select the parameters Θ that maximize the log-likelihood of generating the data D:

More generally, the maximum likelihood criterion is to select the parameters Θ that maximize the log-likelihood of generating the data D:

$$\Theta = \arg\max_{\Theta} L(D|\Theta) = \arg\max_{\Theta} \log \prod_{n=1}^{N} P(d_n|\Theta) = \arg\max_{\Theta} \sum_{n=1}^{N} \log P(d_n|\Theta)$$

$L(D|\Theta)$ is the objective function that measures the goodness of the clustering. Given two clusterings with the same number of clusters, we prefer the one with higher $L(D|\Theta)$.

In text clustering, we chose the class that maximizes the likelihood of generating a particular document.

Here, we choose the clustering Q that maximizes the likelihood of generating a given set of documents

EM Algorithm for a mixture of Bernoulli distributions

A commonly used algorithm for model-based clustering is the *Expectation-Maximization algorithm* or *EM algorithm*. EM clustering is an iterative algorithm that maximizes $L(D|\Theta)$. EM can be applied to many different types of probabilistic modeling. We will work with a mixture of multivariate Bernoulli distributions here, the distribution we know from Section 11.3 (page 222) and Section 13.3 (page 263):

See Section 16.5 in The IR Book

$$P(d|\omega_k;\Theta) = \left(\prod_{t_m \in d} q_{mk}\right)\left(\prod_{t_m \notin d}(1-q_{mk})\right)$$

where $\Theta = \{\Theta_1, \ldots, \Theta_K\}$, $\Theta_k = (\alpha_k, q_{1k}, \ldots, q_{Mk})$, and $q_{mk} = P(U_m = 1|\omega_k)$ are the parameters of the model.[3] $P(U_m = 1|\omega_k)$ is the probability that a document from cluster ω_k contains term t_m. The probability α_k is the prior of cluster ω_k: the probability that a document d is in ω_k if we have no information about d.

The mixture model then is:

$$P(d|\Theta) = \sum_{k=1}^{K} \alpha_k \left(\prod_{t_m \in d} q_{mk}\right)\left(\prod_{t_m \notin d}(1-q_{mk})\right)$$

Use EM as our model depends on unobserved latent variables

- **In statistics, an expectation–maximization (EM) algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables.**
- **The EM iteration alternates between performing an expectation**
- **(E) step,**
  - which creates a function for the expectation of the log-likelihoodevaluated using the current estimate for the parameters, and a maximization
- **(M) step,**
  - which computes parameters maximizing the expected log-likelihood found on the *E* step.
- **These parameter-estimates are then used to determine the distribution of the latent variables in the next E step.**

Use EM to estimate latent variables

- The EM algorithm is used to find the maximum likelihood parameters of a statistical model in cases where the equations cannot be solved directly. Typically these models involve latent variables in addition to unknown parameters and known data observations. That is, either there are missing values among the data, or the model can be formulated more simply by assuming the existence of additional unobserved data points. For example, a mixture model can be described more simply by assuming that each observed data point has a corresponding unobserved data point, or latent variable, specifying the mixture component that each data point belongs to.
- Finding a maximum likelihood solution typically requires taking the derivatives of the likelihood function with respect to all the unknown values — viz. the parameters and the latent variables — and simultaneously solving the resulting equations. In statistical models with latent variables, this usually is not possible. Instead, the result is typically a set of interlocking equations in which the solution to the parameters requires the values of the latent variables and vice versa, but substituting one set of equations into the other produces an unsolvable equation.
- The EM algorithm proceeds from the observation that the following is a way to solve these two sets of equations numerically. One can simply pick arbitrary values for one of the two sets of unknowns, use them to estimate the second set, then use these new values to find a better estimate of the first set, and then keep alternating between the two until the resulting values both converge to fixed points. It's not obvious that this will work at all, but in fact it can be proven that in this particular context it does, and that the derivative of the likelihood is (arbitrarily close to) zero at that point, which in turn means that the point is either a maximum or a saddle point.[*citation needed*] In general there may be multiple maxima, and there is no guarantee that the global maximum will be found. Some likelihoods also have singularities in them, i.e. nonsensical maxima. For example, one of the "solutions" that may be found by EM in a mixture model involves setting one of the components to have zero variance and the mean parameter for the same component to be equal to one of the data points.
-

EM Algorithm for a mixture of Bernoulli distributions

EXPECTATION-MAXIMIZATION ALGORITHM

A commonly used algorithm for model-based clustering is the *Expectation-Maximization algorithm* or *EM algorithm*. EM clustering is an iterative algorithm that maximizes $L(D|\Theta)$. EM can be applied to many different types of probabilistic modeling. We will work with a mixture of multivariate Bernoulli distributions here, the distribution we know from Section 11.3 (page 222) and Section 13.3 (page 263):

W_k is the cluster

$$P(d|\omega_k;\Theta) = \left(\prod_{t_m \in d} q_{mk}\right)\left(\prod_{t_m \notin d}(1-q_{mk})\right) \tag{16.14}$$

where $\Theta = \{\Theta_1, \ldots, \Theta_K\}$, $\Theta_k = (\alpha_k, q_{1k}, \ldots, q_{Mk})$, and $q_{mk} = P(U_m = 1|\omega_k)$ are the parameters of the model.[3] $P(U_m = 1|\omega_k)$ is the probability that a document from cluster ω_k contains term t_m. The probability α_k is the prior of cluster ω_k: the probability that a document d is in ω_k if we have no information about d.

The mixture model then is:

$$P(d|\Theta) = \sum_{k=1}^{K} \alpha_k \left(\prod_{t_m \in d} q_{mk}\right)\left(\prod_{t_m \notin d}(1-q_{mk})\right) \tag{16.15}$$

Example: The EM clustering algorithm

- .

(a)

docID	document text	docID	document text
1	hot chocolate cocoa beans	7	sweet sugar
2	cocoa ghana africa	8	sugar cane brazil
3	beans harvest ghana	9	sweet sugar beet
4	cocoa butter	10	sweet cake icing
5	butter truffles	11	cake black forest
6	sweet chocolate		

E-Step

M-Step

Sugar in class 2 only

Parameter	Iteration of clustering							
	0	1	2	3	4	5	15	25
α_1	0.50	0.45	0.53	0.57	0.58	0.54	0.45	
$r_{1,1}$		1.00	1.00	1.00	1.00	1.00	1.00	1.00
$r_{2,1}$		0.50	0.79	0.99	1.00	1.00	1.00	1.00
$r_{3,1}$		0.50	0.84	1.00	1.00	1.00	1.00	1.00
$r_{4,1}$		0.50	0.75	0.94	1.00	1.00	1.00	1.00
$r_{5,1}$		0.50						
$r_{6,1}$	1.00	1.00						
$r_{7,1}$	0.00	0.00						
$r_{8,1}$		0.00						
$r_{9,1}$		0.00						
$r_{10,1}$		0.50						
$r_{11,1}$		0.50						
$q_{\text{africa},1}$	0.000	0.100	0.134	0.158	0.158	0.169	0.200	
$q_{\text{africa},2}$	0.000	0.083	0.042	0.001	0.000	0.000	0.000	
$q_{\text{brazil},1}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
$q_{\text{brazil},2}$	0.000	0.167	0.195	0.213	0.214	0.196	0.167	
$q_{\text{cocoa},1}$	0.000	0.400	0.432	0.465	0.474	0.508	0.600	
$q_{\text{cocoa},2}$	0.000	0.167	0.090	0.014	0.001	0.000	0.000	
$q_{\text{sugar},1}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
$q_{\text{sugar},2}$	1.000	0.500	0.585	0.640	0.642	0.589	0.500	
$q_{\text{sweet},1}$	1.000	0.300	0.238	0.180	0.159	0.153	0.000	
$q_{\text{sweet},2}$	1.000	0.417	0.507	0.610	0.640	0.608	0.667	

(a)

docID	document text	docID	document
1	hot chocolate cocoa beans	7	sweet suga
2	cocoa ghana africa	8	sugar cane
3	beans harvest ghana	9	sweet suga
4	cocoa butter	10	sweet cake
5	butter truffles	11	cake black
6	sweet chocolate		

Class Priors

E-Step

M-Step

Sugar in class 2 only

Parameter	Iteration of clustering							
	0	1	2	3	4	5	15	25
α_1	0.50	0.45	0.53	0.57	0.58	0.54	0.45	
$r_{1,1}$		1.00	1.00	1.00	1.00	1.00	1.00	1.00
$r_{2,1}$		0.50	0.					
$r_{3,1}$		0.50	0.					
$r_{4,1}$		0.50	0.					
$r_{5,1}$		0.50	0.					
$r_{6,1}$	1.00	1.00	1.					
$r_{7,1}$	0.00	0.00	0.					
$r_{8,1}$		0.00	0.00	0.00	0.00	0.00	0.00	0.00
$r_{9,1}$		0.00	0.00	0.00	0.00	0.00	0.00	0.00
$r_{10,1}$		0.50	0.40	0.14	0.01	0.00	0.00	0.00
$r_{11,1}$		0.50	0.57	0.58	0.41	0.07	0.00	0.00
$q_{\text{africa},1}$	0.000	0.100	0.134	0.158	0.158	0.169	0.200	
$q_{\text{africa},2}$	0.000	0.083	0.042	0.001	0.000	0.000	0.000	
$q_{\text{brazil},1}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
$q_{\text{brazil},2}$	0.000	0.167	0.195	0.213	0.214	0.196	0.167	
$q_{\text{cocoa},1}$	0.000	0.400	0.432	0.465	0.474	0.508	0.600	
$q_{\text{cocoa},2}$	0.000	0.167	0.090	0.014	0.001	0.000	0.000	
$q_{\text{sugar},1}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
$q_{\text{sugar},2}$	1.000	0.500	0.585	0.640	0.642	0.589	0.500	
$q_{\text{sweet},1}$	1.000	0.300	0.238	0.180	0.159	0.153	0.000	
$q_{\text{sweet},2}$	1.000	0.417	0.507	0.610	0.640	0.608	0.667	

(a)

docID	document text	docID	document text
1	hot chocolate cocoa beans	7	sweet sugar
2	cocoa ghana africa	8	sugar cane brazil
3	beans harvest ghana	9	sweet sugar beet
4	cocoa butter	10	sweet cake icing
5	butter truffles	11	cake black forest
6	sweet chocolate		

E-Step

M-Step

Parameter	Iteration of clustering							
	0	1	2	3	4	5	15	25
α_1	0.50	0.45	0.53	0.57	0.58	0.54	0.45	
$r_{1,1}$		1.00	1.00	1.00	1.00	1.00	1.00	1.00
$r_{2,1}$		0.50	0.7					
$r_{3,1}$		0.50	0.					
$r_{4,1}$		0.50	0.					
$r_{5,1}$		0.50	0.					
$r_{6,1}$	1.00	1.00	1.					
$r_{7,1}$	0.00	0.00	0.					
$r_{10,1}$		0.50	0.40	0.14	0.01	0.00	0.00	0.00
$r_{11,1}$		0.50	0.57	0.58	0.41	0.07	0.00	0.00
$q_{\text{africa},1}$	0.000	0.100	0.134	0.158	0.158	0.169	0.200	
$q_{\text{africa},2}$	0.000	0.083	0.042	0.001	0.000	0.000	0.000	
$q_{\text{brazil},1}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
$q_{\text{brazil},2}$	0.000	0.167	0.195	0.213	0.214	0.196	0.167	
$q_{\text{cocoa},1}$	0.000	0.400	0.432	0.465	0.474	0.508	0.600	
$q_{\text{cocoa},2}$	0.000	0.167	0.090	0.014	0.001	0.000	0.000	
$q_{\text{sugar},1}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
$q_{\text{sugar},2}$	1.000	0.500	0.585	0.640	0.642	0.589	0.500	
$q_{\text{sweet},1}$	1.000	0.300	0.238	0.180	0.159	0.153	0.000	
$q_{\text{sweet},2}$	1.000	0.417	0.507	0.610	0.640	0.608	0.667	

(a)

docID	document text	docID	document text
1	hot chocolate cocoa beans	7	sweet sugar
2	cocoa ghana africa	8	sugar cane brazil
3	beans harvest ghana	9	sweet sugar beet
4	cocoa butter	10	sweet cake icing
5	butter truffles	11	cake black forest
6	sweet chocolate		

Soft cluster assigments for each document

Sugar in class 2 only

E-Step

M-Step

Parameter	Iteration of clustering							
	0	1	2	3	4	5	15	25
α_1	0.50	0.45	0.53	0.57	0.58	0.54	0.45	
$r_{1,1}$		1.00	1.					
$r_{2,1}$		0.50	0.					
$r_{3,1}$		0.50	0.					
$r_{4,1}$		0.50	0.					
$r_{5,1}$		0.50	0.52	0.66	0.91	1.00	1.00	1.00
$r_{6,}$								
$r_{7,}$								
$r_{8,}$								
$r_{9,}$								
r_{10}								
r_{11}								
q_{af}								
q_{af}								
q_{br}								
q_{br}								
q_{cc}								
q_{cc}								
$q_{\text{sugar},1}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
$q_{\text{sugar},2}$	1.000	0.500	0.585	0.640	0.642	0.589	0.500	
$q_{\text{sweet},1}$	1.000	0.300	0.238	0.180	0.159	0.153	0.000	
$q_{\text{sweet},2}$	1.000	0.417	0.507	0.610	0.640	0.608	0.667	

nent text

1	hot chocolate cocoa beans	7	sweet sugar
2	cocoa ghana africa	8	sugar cane brazil
3	beans harvest ghana	9	sweet sugar beet
4	cocoa butter	10	sweet cake icing
5	butter truffles	11	cake black forest
6	sweet chocolate		

Word Class conditionals Pr(Word=sugar|Class=2) = $q_{sugar,2}$
Sugar in Doc7 and we only have one Doc in class 2

➔ Iteration1:

$q_{sugar,2}$=1

1/1 #of docs in class 2 with Sweet / # of docs in class 2

➔ ?Smoothing

➔ M-Step is exactly the **Bernoulli Naive Bayes**

Sugar in class 2 only

EM for a mixture of Gaussian Distributions

- Maximum Likelihood MLE
  - Pick Θ to maximize P(traindata | Θ)
- Handle partially observed data
  - What if traindata is not only partially observed with Z hidden and X observed ?
  - Now pick Θ to maximize P(X, Z | Θ) but we don't know what Z is.
  - We can only maximize $E_{\text{over } Z}(P(X, Z \mid \Theta))$. That is, pick Θ to maximize $E_{P(Z|X,\Theta)}(P(X, Z \mid \Theta))$
- Algorithm: Expectation Maximization
  - Iterate until converge
    - Compute $P(Z|X,\Theta)$
    - Compute Θ' that maximize $E_{\text{over } Z}(P(X, Z \mid \Theta'))$
    - Set Θ to Θ'

Z is the cluster variable
μ,σ are cluster parameters

Initialization via KMeans

- Finding good seeds is even more critical for EM than for **K** -means. EM is prone to get stuck in local optima if the seeds are not chosen well. This is a general problem that also occurs in other applications of EM.4
- Therefore, as with **K** -means, the initial assignment of documents to clusters is often computed by a different algorithm.
- For example, a hard **K** -means clustering may provide the initial assignment, which EM can then "soften up."

- In this model, we generate a document by first picking a cluster k with probability α_k and then generating the terms of the document according to the parameters q_{mk} .
- Recall that the document representation of the multivariate Bernoulli is a vector of M Boolean values (and not a real-valued vector).

EM Algo: infer the parameters of the clustering from the data

- How do we choose parameters Θ thatmaximize **L(D|Θ)** ?
- EM is similar to **K** -means in that it alternates between an **expectation step** , corresponding to reassignment, and a **maximization step** , corresponding to recomputation of the parameters of the model.
- The parameters of **K** -means are the centroids, while the parameters of the instance of EM in this section are the α_k and q_{mk} .

Expectation Maximization Algorithm

- A commonly used algorithm for model-based clustering is the Expectation Maximization algorithm or EM algorithm

- EM clustering is an iterative algorithm that maximizes $L(D|\Theta)$
- EM can be applied to many different types of probabilistic modeling.
- We will work with a mixture of multivariate Bernoulli distributions here,

Multivariate Bernoulli: Estimates

- ***Model 1*: Multivariate Bernoulli**
  - One feature X_w for each word in dictionary
  - X_w = true in document d if w appears in d
  - Naive Bayes assumption:
    - Given the document's topic, appearance of one word in the document tells us nothing about chances that another word appears
- **Bernoulli model estimates the P(t|c) as the fraction of documents of class c that contain the term t.**

Parameter estimation

- **Multivariate Bernoulli model:**

$$\hat{P}(X_w = t \mid c_j) = \text{fraction of documents of topic } c_j \text{ in which word } w \text{ appears}$$

- **Multinomial model:**

$$\hat{P}(X_i = w \mid c_j) = \frac{\text{fraction of times in which word } w \text{ appears across all documents of topic } c_j}{}$$

  - Can create a mega-document for topic *j* by concatenating all documents on this topic
  - Use frequency of *w* in mega-document

Classification

- **Multinomial vs Multivariate Bernoulli?**
- **Multinomial model is almost always more effective in text applications!**
- **See *IR Book* sections 13.2 and 13.3 for worked examples with each model**

	docID	words in document	in $c = China$?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Vocabulary = {Chinese, Beijing, Shanghai, Macao, Tokyo}

Example 13.2: Applying the Bernoulli model to the example in Table 13.1, we have the same estimates for the priors as before: $\hat{P}(c) = 3/4$, $\hat{P}(\overline{c}) = 1/4$. The conditional probabilities are:

$$\begin{aligned}
\hat{P}(\text{Chinese}|c) &= (3+1)/(3+2) = 4/5\\
\hat{P}(\text{Japan}|c) = \hat{P}(\text{Tokyo}|c) &= (0+1)/(3+2) = 1/5\\
\hat{P}(\text{Beijing}|c) = \hat{P}(\text{Macao}|c) = \hat{P}(\text{Shanghai}|c) &= (1+1)/(3+2) = 2/5\\
\hat{P}(\text{Chinese}|\overline{c}) &= (1+1)/(1+2) = 2/3\\
\hat{P}(\text{Japan}|\overline{c}) = \hat{P}(\text{Tokyo}|\overline{c}) &= (1+1)/(1+2) = 2/3\\
\hat{P}(\text{Beijing}|\overline{c}) = \hat{P}(\text{Macao}|\overline{c}) = \hat{P}(\text{Shanghai}|\overline{c}) &= (0+1)/(1+2) = 1/3
\end{aligned}$$

The denominators are $(3+2)$ and $(1+2)$ because there are three documents in c and one document in $\overline{c}$ and because the constant B in Equation (13.7) is 2 – there are two cases to consider for each term, occurrence and nonoccurrence.

The scores of the test document for the two classes are

$$\begin{aligned}
\hat{P}(c|d_5) &\propto \hat{P}(c)\cdot\hat{P}(\text{Chinese}|c)\cdot\hat{P}(\text{Japan}|c)\cdot\hat{P}(\text{Tokyo}|c)\\
&\quad\cdot(1-\hat{P}(\text{Beijing}|c))\cdot(1-\hat{P}(\text{Shanghai}|c))\cdot(1-\hat{P}(\text{Macao}|c))\\
&= 3/4\cdot 4/5\cdot 1/5\cdot 1/5\cdot(1-2/5)\cdot(1-2/5)\cdot(1-2/5)\\
&\approx 0.005
\end{aligned}$$

and, analogously,

$$\begin{aligned}
\hat{P}(\overline{c}|d_5) &\propto 1/4\cdot 2/3\cdot 2/3\cdot 2/3\cdot(1-1/3)\cdot(1-1/3)\cdot(1-1/3)\\
&\approx 0.022
\end{aligned}$$

Thus, the classifier assigns the test document to $\overline{c}$ = *not-China*. When looking only at binary occurrence and not at term frequency, Japan and Tokyo are indicators for $\overline{c}$ ($2/3 > 1/5$) and the conditional probabilities of Chinese for c and $\overline{c}$ are not different enough (4/5 vs. 2/3) to affect the classification decision.

	docID	words in document	in c = *China*?
training set	1	Chinese Beijing Chinese	yes
	2	Chinese Chinese Shanghai	yes
	3	Chinese Macao	yes
	4	Tokyo Japan Chinese	no
test set	5	Chinese Chinese Chinese Tokyo Japan	?

Bernoulli NB: Smoothing is based number of classes

- **6 terms in Vocabulary 6 terms in calculation of P(C|X)**
- **Count single occurrence of term, .e.g., Chinese is just counted once**

▶ **Table 13.3** Multinomial versus Bernoulli model.

	multinomial model	Bernoulli model
event model	generation of token	generation of document
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
document representation	$d = \langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle, t_k \in V$	$d = \langle e_1, \ldots, e_i, \ldots, e_M \rangle,$ $e_i \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t\|c)$	$\hat{P}(U_i = e\|c)$
decision rule: maximize	$\hat{P}(c) \prod_{1 \le k \le n_d} \hat{P}(X = t_k\|c)$	$\hat{P}(c) \prod_{t_i \in V} \hat{P}(U_i = e_i\|c)$
multiple occurrences	taken into account	ignored
length of docs	can handle longer docs	works best for short docs
# features	can handle more	works best with fewer
estimate for term the	$\hat{P}(X = \text{the}\|c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1\|c) \approx 1.0$

EM Bernoulli using MRJob

- **Implement!!**

- **End of Lectures**