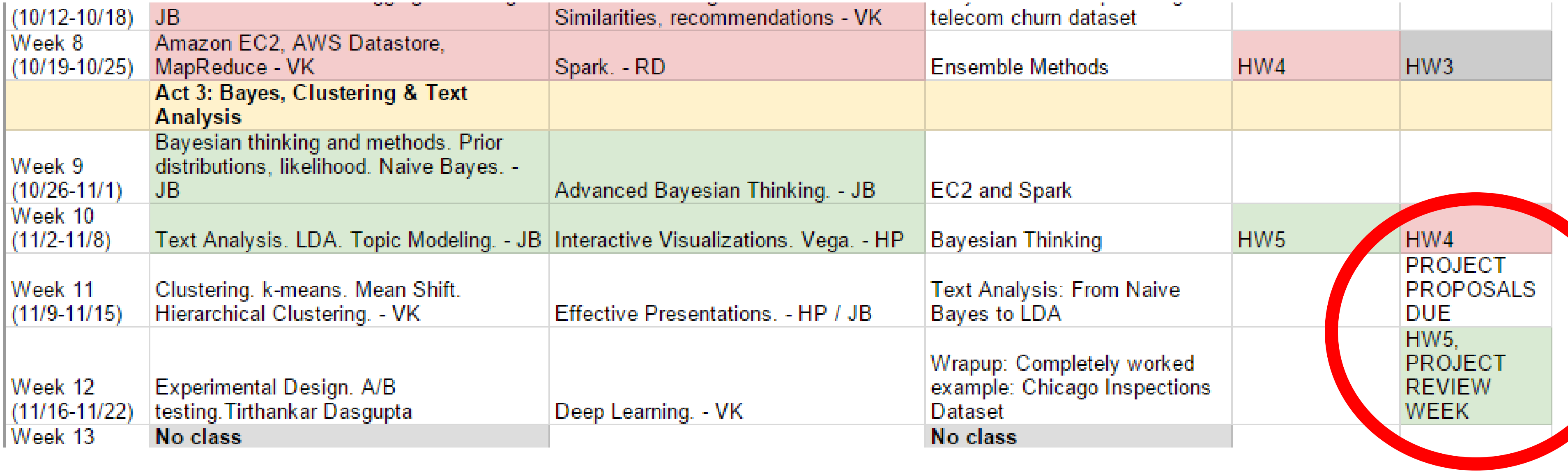
CS109 – Data Science

Joe Blitzstein, Hanspeter Pfister, Verena Kaynig-Fittkau

vkaynig@seas.harvard.edu
staff@cs109.org

Announcements

- Grades for HW2 are getting out tonight
- Final Projects:
  - 3-4 persons per team



(10/12-10/18)	JB	Similarities, recommendations - VK	telecom churn dataset		
Week 8 (10/19-10/25)	Amazon EC2, AWS Datastore, MapReduce - VK	Spark. - RD	Ensemble Methods	HW4	HW3
	Act 3: Bayes, Clustering & Text Analysis				
Week 9 (10/26-11/1)	Bayesian thinking and methods. Prior distributions, likelihood. Naive Bayes. - JB	Advanced Bayesian Thinking. - JB	EC2 and Spark		
Week 10 (11/2-11/8)	Text Analysis. LDA. Topic Modeling. - JB	Interactive Visualizations. Vega. - HP	Bayesian Thinking	HW5	HW4
Week 11 (11/9-11/15)	Clustering. k-means. Mean Shift. Hierarchical Clustering. - VK	Effective Presentations. - HP / JB	Text Analysis: From Naive Bayes to LDA		PROJECT PROPOSALS DUE
Week 12 (11/16-11/22)	Experimental Design. A/B testing.Tirthankar Dasgupta	Deep Learning. - VK	Wrapup: Completely worked example: Chicago Inspections Dataset		HW5, PROJECT REVIEW WEEK
Week 13	**No class**		**No class**		

Proposal: final project aspect —> Review is meeting with TF who likes project

Next Topics

- ML best practices
  - imbalanced data
  - missing values
- Recommender systems
  - collaborative filtering
  - content-based filtering
- Map Reduce

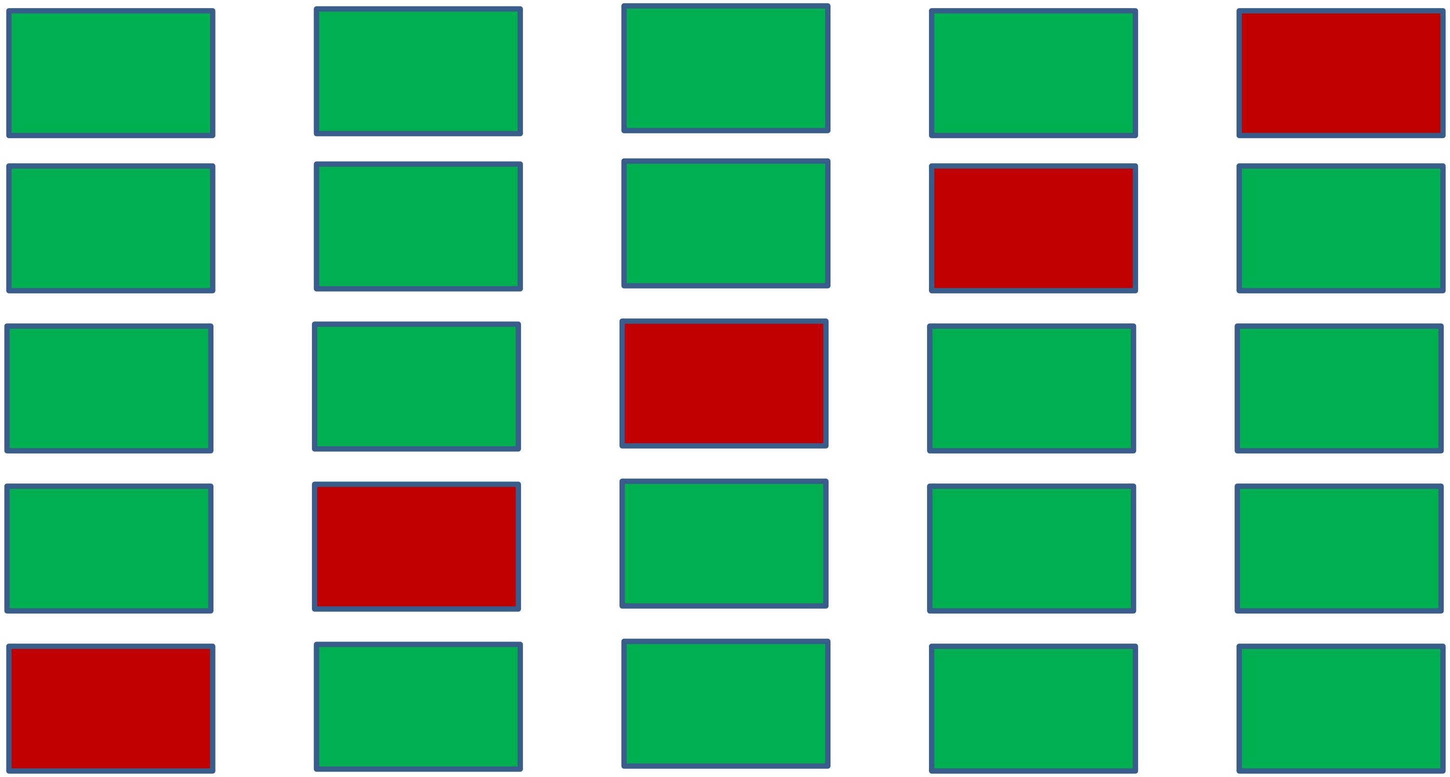
Cross Validation

🟩	🟨	🟥
training data	validation data	test data

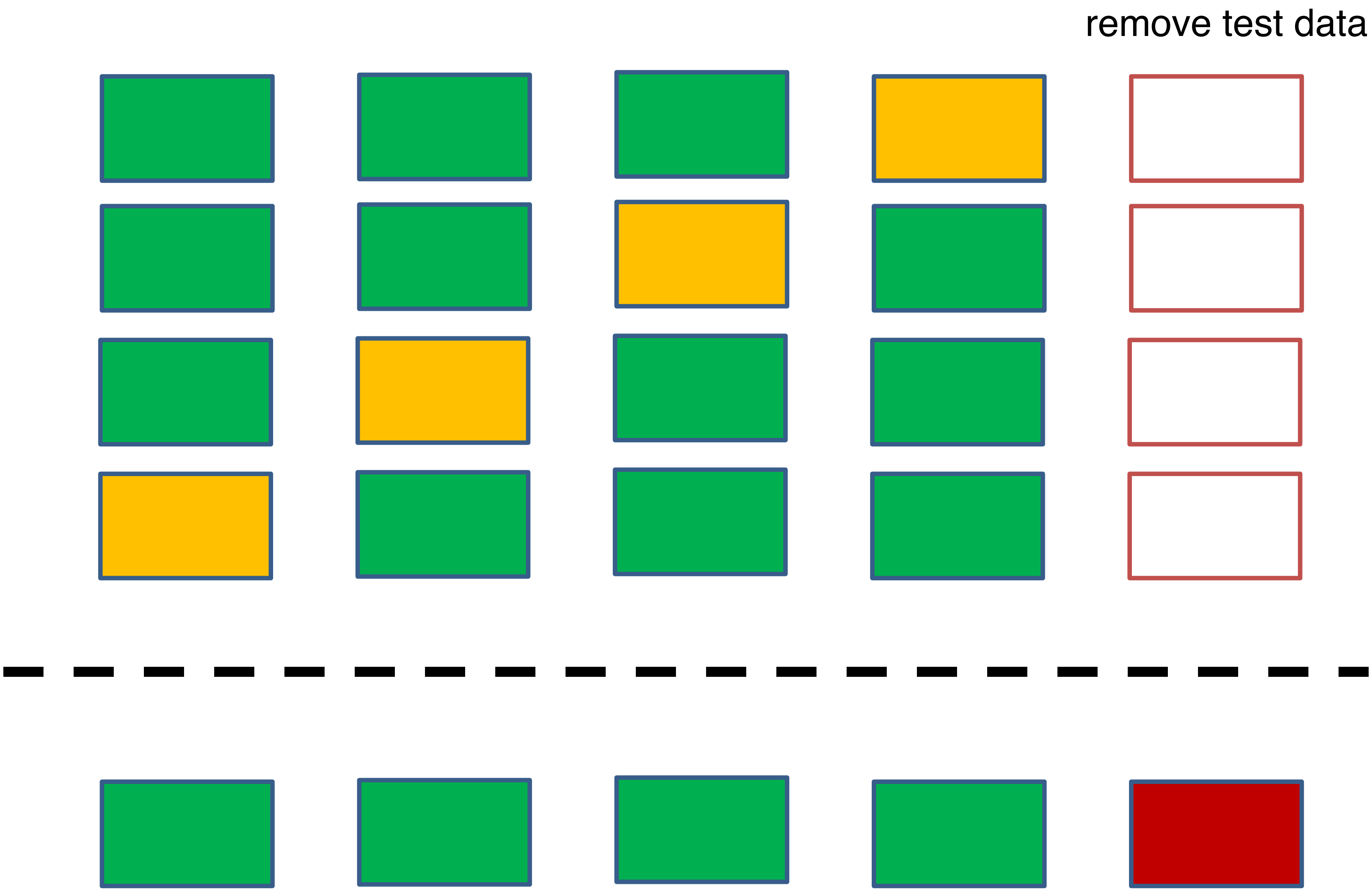
- Training data: train classifier
- Validation data: estimate hyper parameters
- Test data: estimate performance

- Be mindful of validation and test set, validation set might refer to test set in some papers.

5 – Fold Cross Validation



5 – Fold Cross Validation



remove test data

Last Step of Each Fold

1. Take best parameters 🟨 from validation set.
2. Train on training data and validation data together 🟩 🟨
3. Test performance on test data 🟥

   Only in last step do you touch test data.

This is the **final** result of your method.

Things to Keep in Mind

- How do you aggregate the parameters?
- What if the hyperparameters are all over the place?
- What if the hyperparameters are at the border of your grid search window?

Test beyond the window then.

This is wrong: used whole data, including test data for feature selection.

Scenario - 1

- 1. Screen the predictors: find a subset of “good” predictors that show fairly strong (univariate) correlation with the class labels
- 2. Using just this subset of predictors, build a multivariate classifier.
- 3. Use cross-validation to estimate the unknown tuning parameters and to estimate the prediction error of the final model.

Hastie-Tibsherani_Friedman, “The Elements of Statistical Learning”

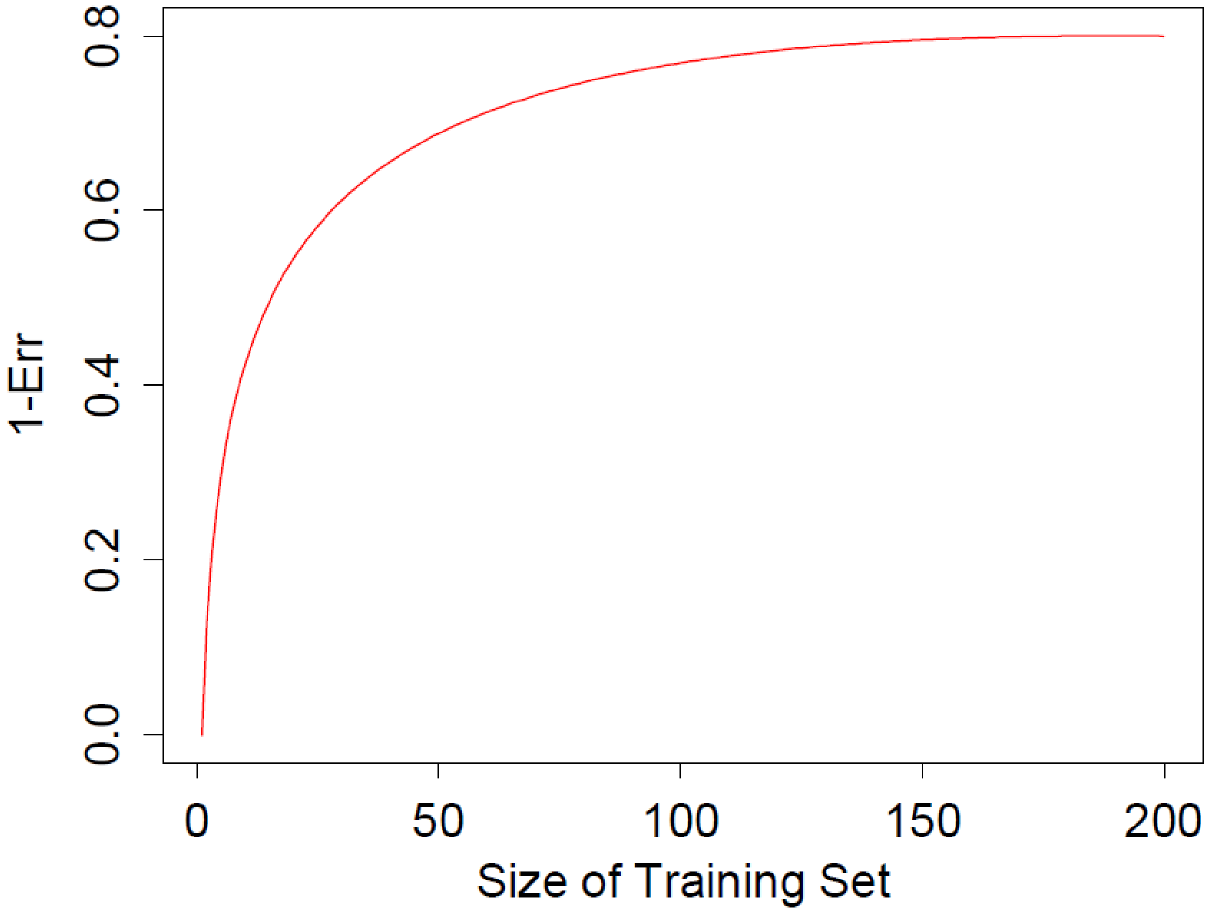
Correct.

Scenario - 2

- 1. Divide the samples into K cross-validation folds (groups) at random.
- 2. For each fold k = 1, 2, . . . ,K
  - Find a subset of “good” predictors that show fairly strong (uni-variate) correlation with the class labels, using all of the samples except those in fold k.
  - Using just this subset of predictors, build a multivariate classifier, using all of the samples except those in fold k.
  - Use the classifier to predict the class labels for the samples in fold k.

Hastie-Tibsherani_Friedman, “The Elements of Statistical Learning”

Effect of Sample Size



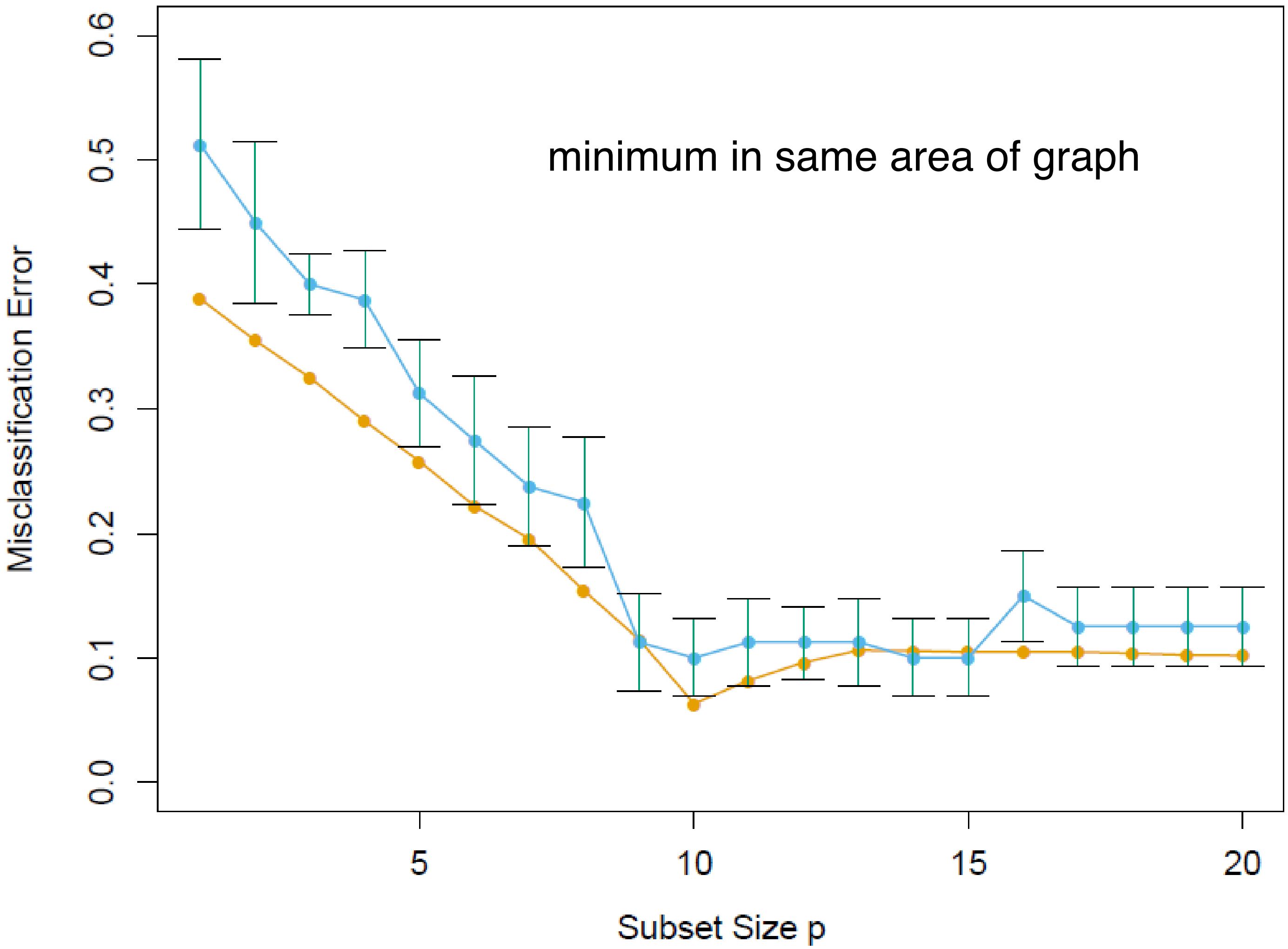
5-fold cross validation:

- n=200 => 160 samples
- n=50 => 40 samples

small dataset, get big diff in error, This is ok because under-promised, but perhaps using fewer folds -> set aside less for test and you can use more in training

Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction"

Cross Validation Over Estimates Error



minimum in same area of graph

Hastie et al.,"The Elements of Statistical Learning: Data Mining, Inference, and Prediction"

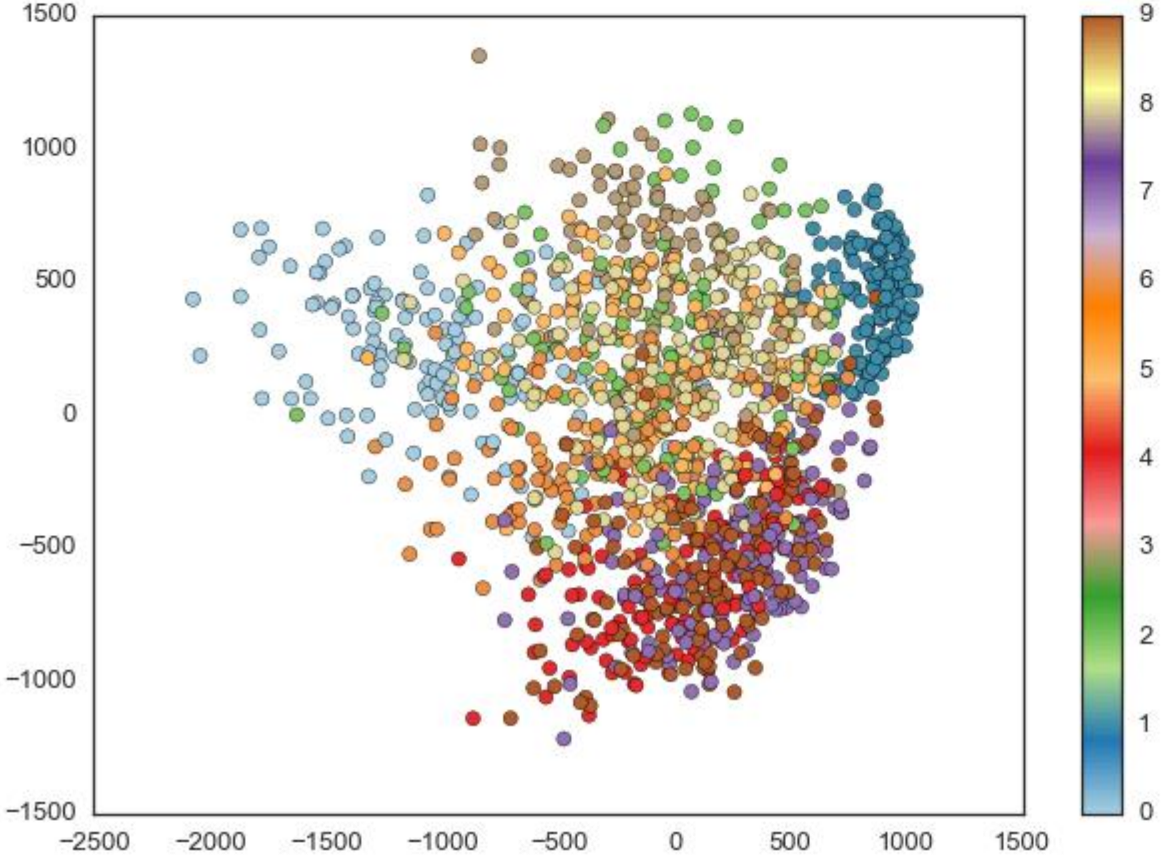
Normalization

- Be very careful.
- Do not leak into the test data.
- Think about what is useful.

  Units, standard deviation, range of data?

Already normalized: features go (0,1)

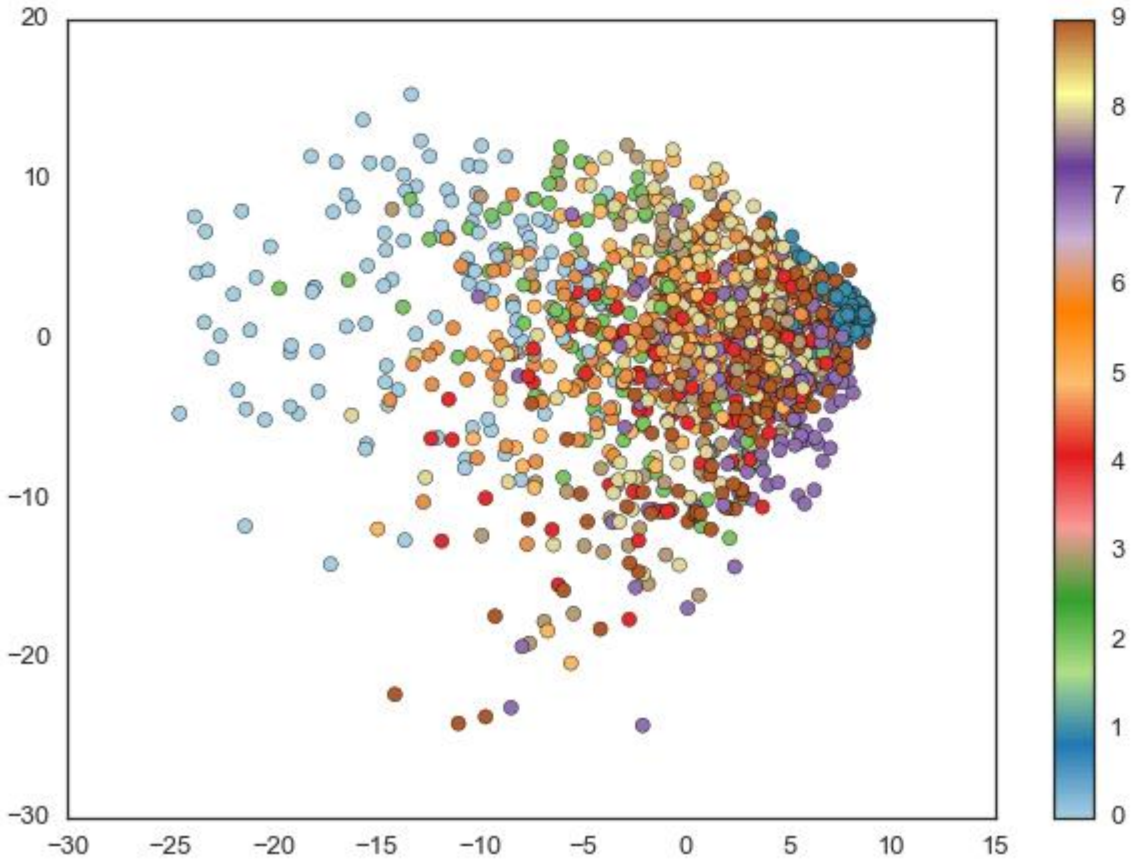
Example PCA on MNIST



standard PCA

Worse than above: visualization ruined.

Example PCA on MNIST

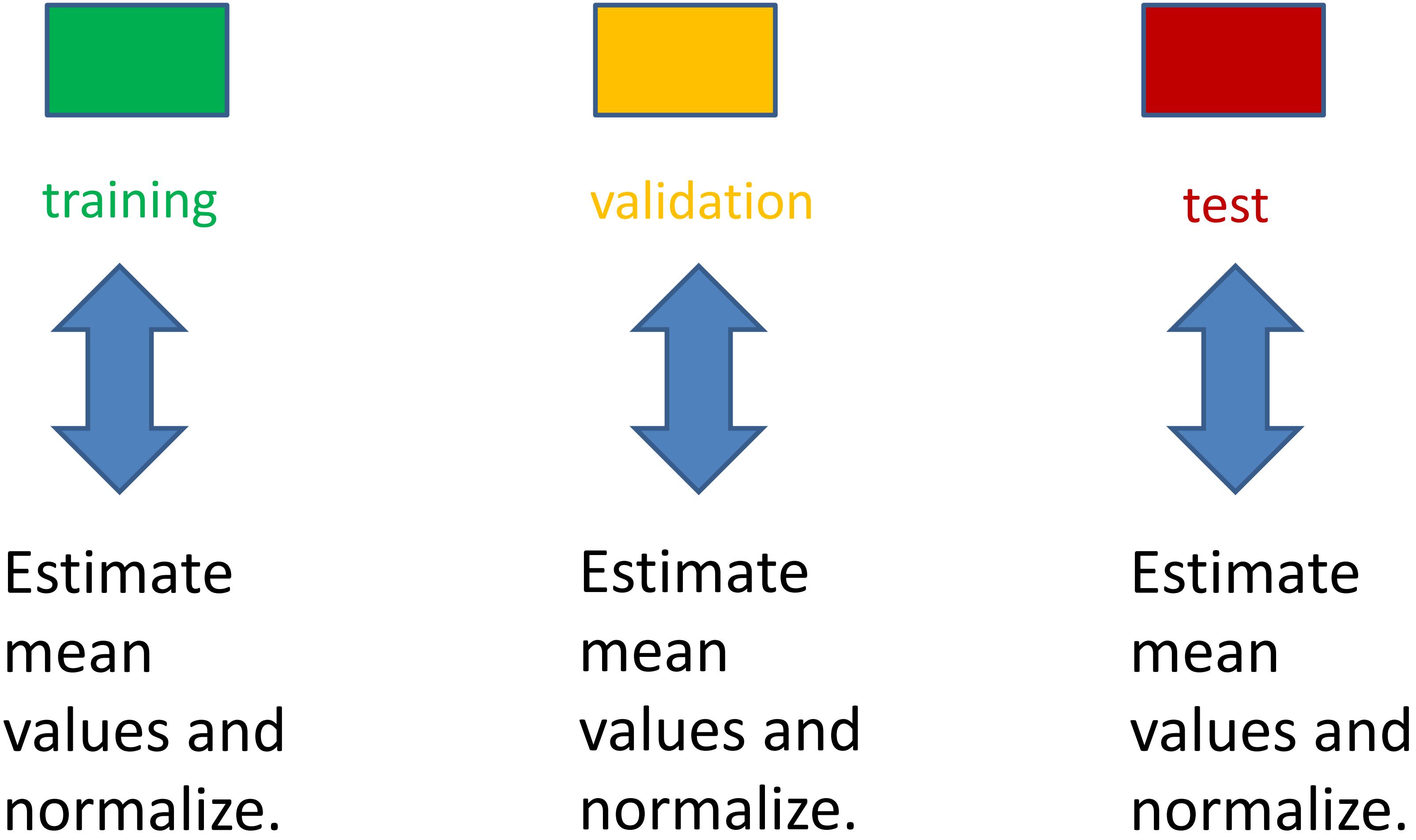


PCA with normalized std dev

normalize all parts of data seperately. —> this is fine

assumes all data from same distribution: and it's ok to have small validation and test sets

Normalization - 1



training

validation

test

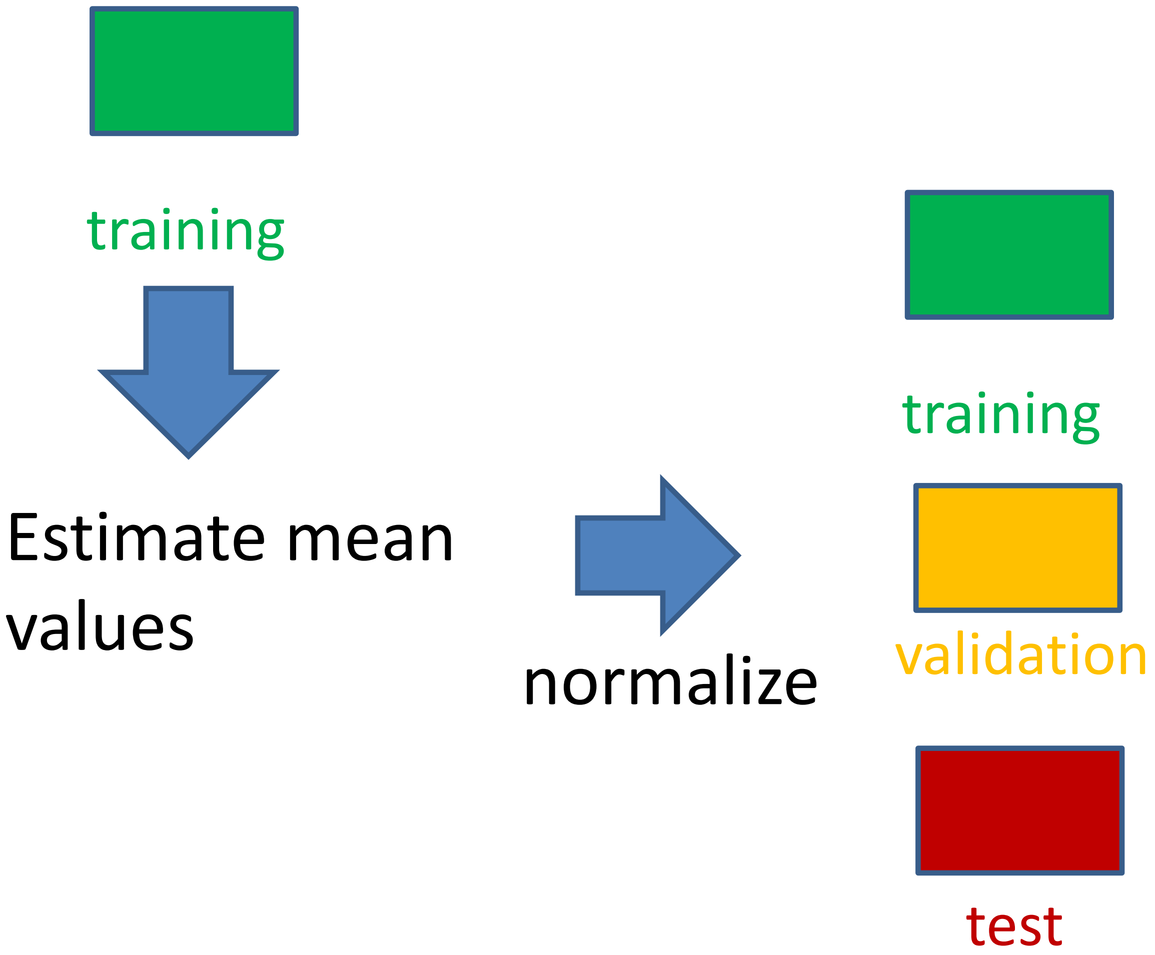
Estimate mean values and normalize.

Estimate mean values and normalize.

Estimate mean values and normalize.

Use largest chunk of data: training to get out mean, then normalize seperately
Now we don’t assume same distribution.

Normalization - 2



training

Estimate mean values

normalize

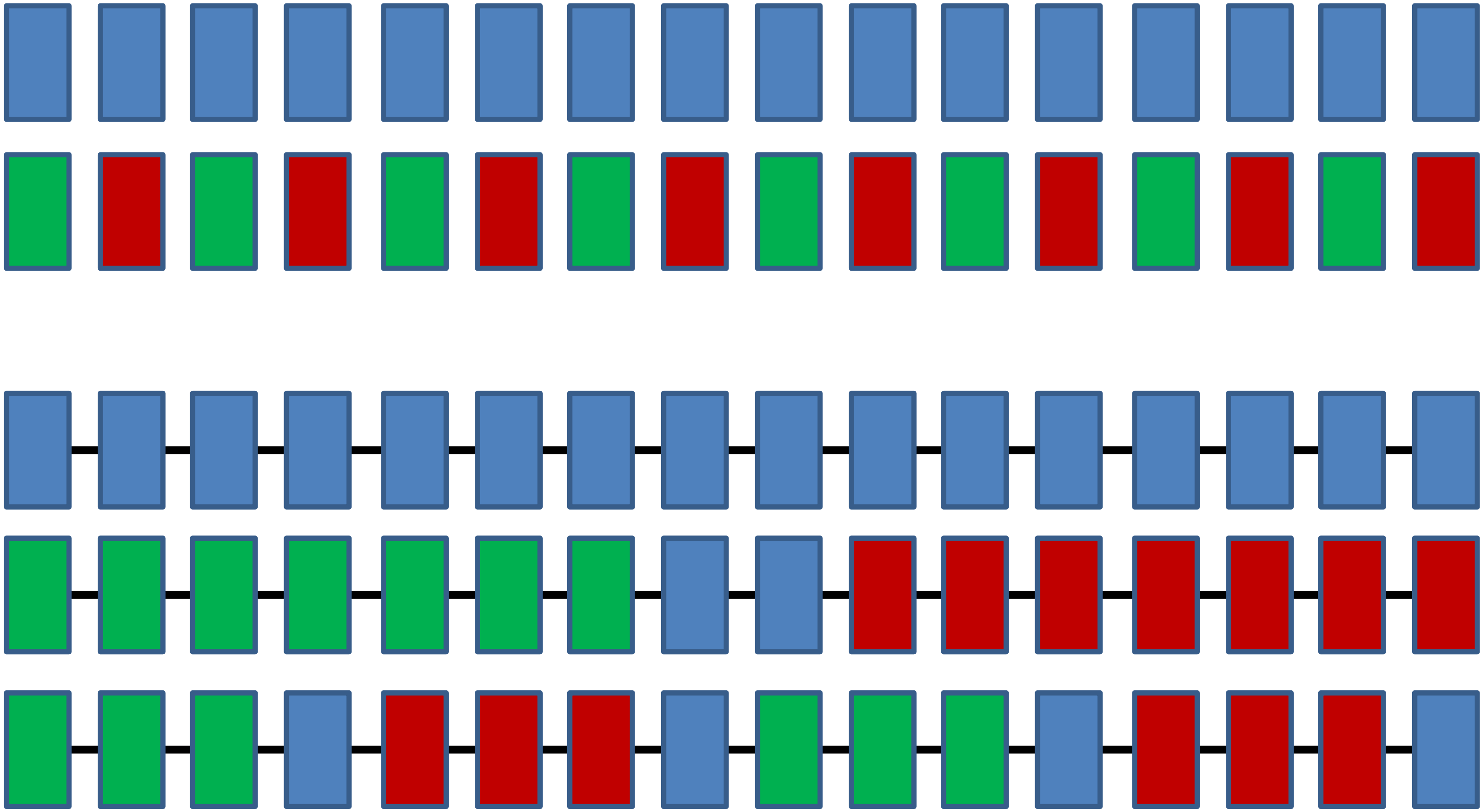
training

validation

test

This cross validation scheme meant for independent samples.
What if the data are correlated to each other, even if randomly selected out?

Know Your Data



Correlated test and train: well your test data isn't helpful now?

Imbalanced Data

- subsample
- oversample
- re-weight sample points
- use clustering to reduce majority class

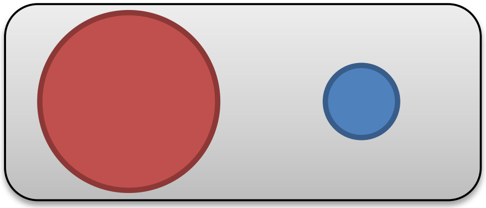
- re-calibrate classifier output

  ie, change threshold.

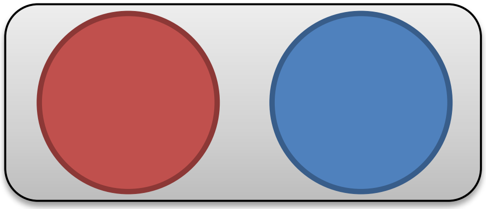
- Beware the easy true negatives

Imbalanced Classes

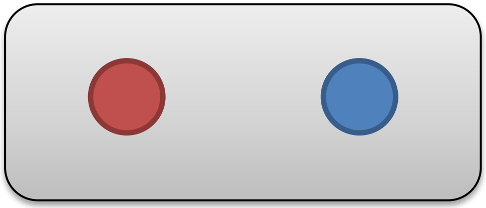
- The Problem:



- Oversample:

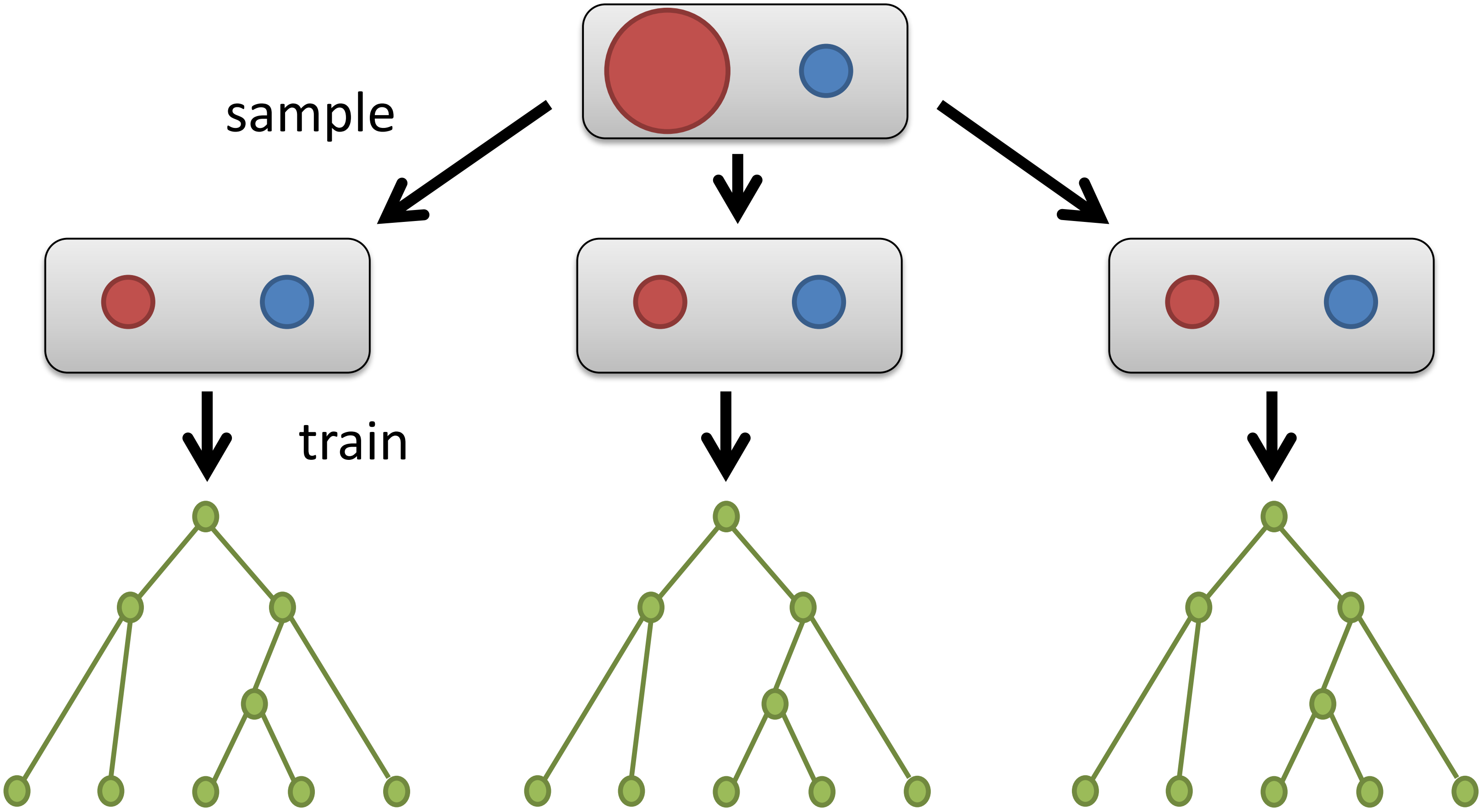


- Subsample:



- Subsample for each tree in a random forest

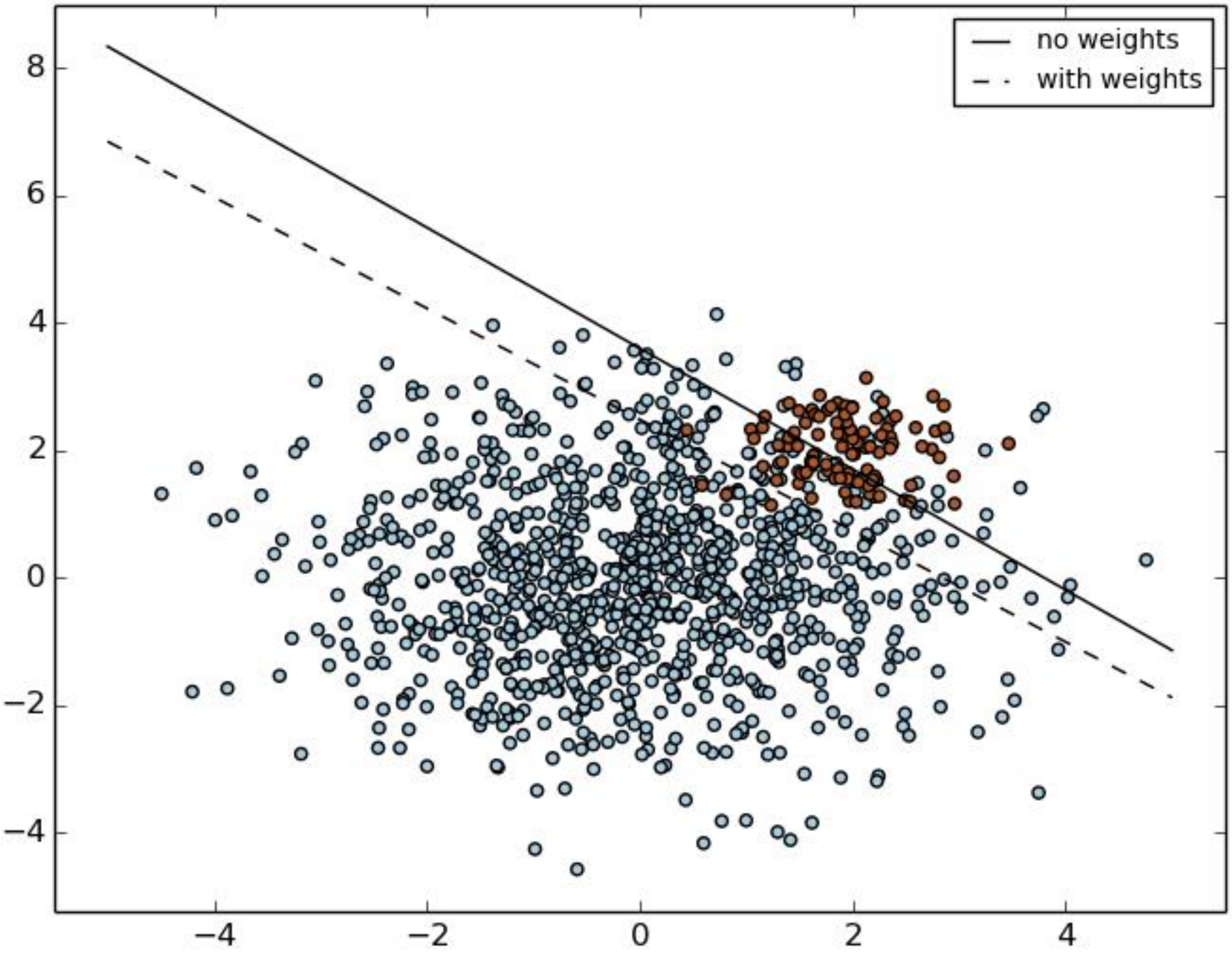
Example: Random Forest Subsampling



bootstrap: slightly different subsamples each time. Each tree sees balanced sample.

Reweight points. Ie, boosting.
Similar to counting parts of bootstrap sample more.

Class Weights



http://scikit-learn.org/stable/_images/plot_separating_hyperplane_unbalanced_0011.png

Cross Validation with Imbalanced Classes

- Think about using stratified sampling to generate the folds
- The goal is to have the same class ratio in training, validation and test set.

Missing data

- Delete data points
  - Can cause sample size to be way too small

- Use the mean of the feature
  - Does not change the sample mean, but is independent of the other features.

    destroyed correlation within one observation and other features.

- Use regression to estimate the value
  - Values will be deterministic

  interpolating

I think you'd like to buy X given you've bought Y before.

Recommender Systems

- We are already surrounded by them

NETFLIX
amazon
PANDORA

Good Resources (also for this lecture)

Survey on recommender systems by Michael D. Ekstrand et al.

- http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf

Good slides from Stanford lecture by Lester Mackey

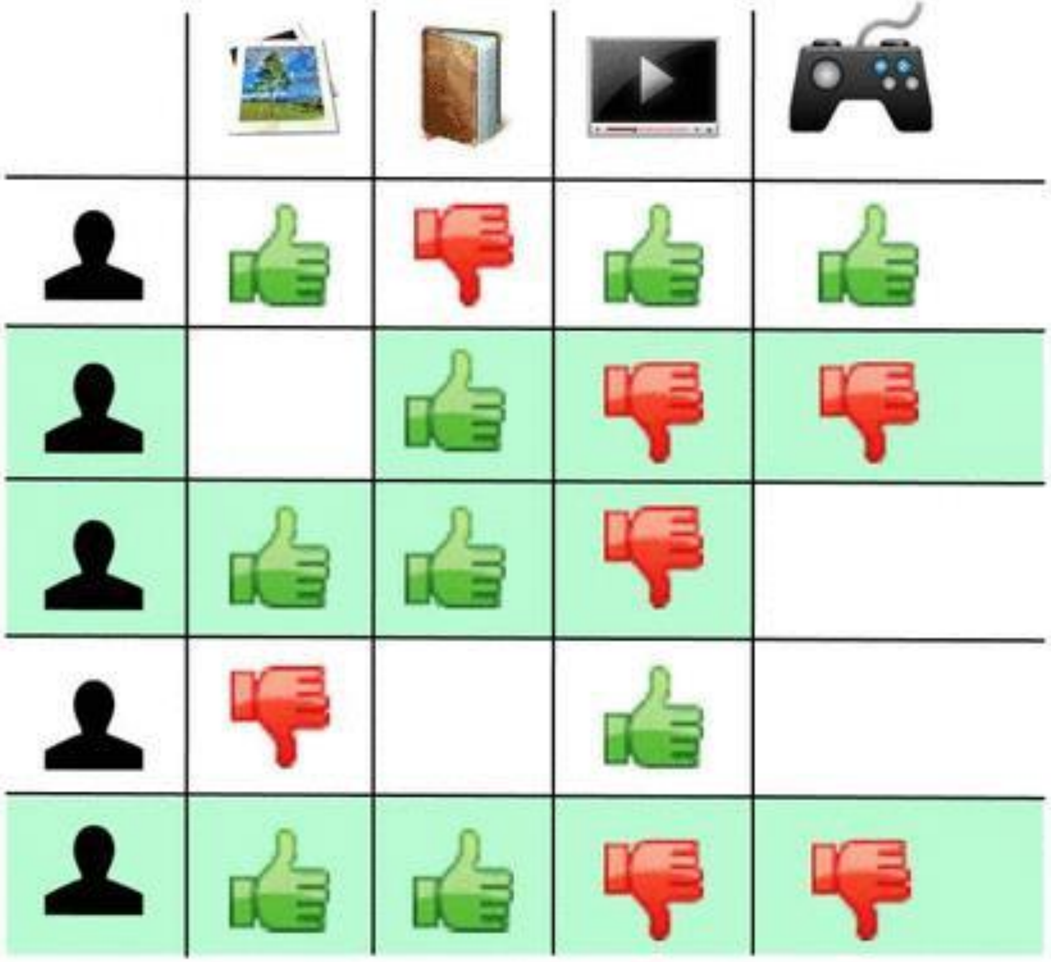
- http://web.stanford.edu/~lmackey/papers/cf_slides-pml09.pdf

recommendation system: want to fill in blanks based on other:
want to use users similar to the one predicting on based on past preferences.

Rating Matrix Completion Problem

features

observations (users)



👤	👍	👎	👍	👍
👤		👍	👎	👎
👤	👍	👍	👎	
👤	👎		👍	
👤	👍	👍	👎	👎

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

Collaborative Filtering

Insight: Personal preferences are correlated

- If Jack loves A and B, and Jill loves A, B, and C, then Jack is more likely to love C

- Does not rely on item or user attributes (e.g. demographic info, author, genre)

May recommend diff category (ie movie genre) based on preferences of other
—> move outside comfort zone.

Netflix: genre, actors, screen writer, etc. Recommend movies that are similar along these features if user has liked something similar before.

Content-based Filtering

- Each item is described by a set of features
- Measure similarity between items
- Recommend items that are similar to the items the User liked

Keeps user in comfort zone: similar items.

Comparison

- Collaborative filtering:
  - Items entirely described by user ratings
  - Good for new discoveries
  - People who like SciFi maybe also like Fantasy
- Content-based filtering:
  - Predictions are in users comfort zone
  - Can start with a single item
- Can do a hybrid approach

User Based Collaborative Filtering

Intuition:

- I like what people similar to me like
- Users give ratings
- People with similar ratings in the past assumed to have similar ratings in the future

cold start problem: new user, no information.

Item-based Collaborative Filtering

- Similar to user-based, but looks at the items instead of the users
- Useful if the user base is way larger than the number of items.
- More useful: Items are relatively stable in their rating, users vary more.

We Could Use Missing Data Strategies

All that we talked about earlier:

- Omitting samples
- Using the mean rating of an item
- Doing regression

CF as Regression

- Choose favorite regression algorithm
- Train a predictor for each item
- Each user who rated that item provides one sample
- To predict rating of an item A, apply predictor for A to the user’s incomplete ratings vector.

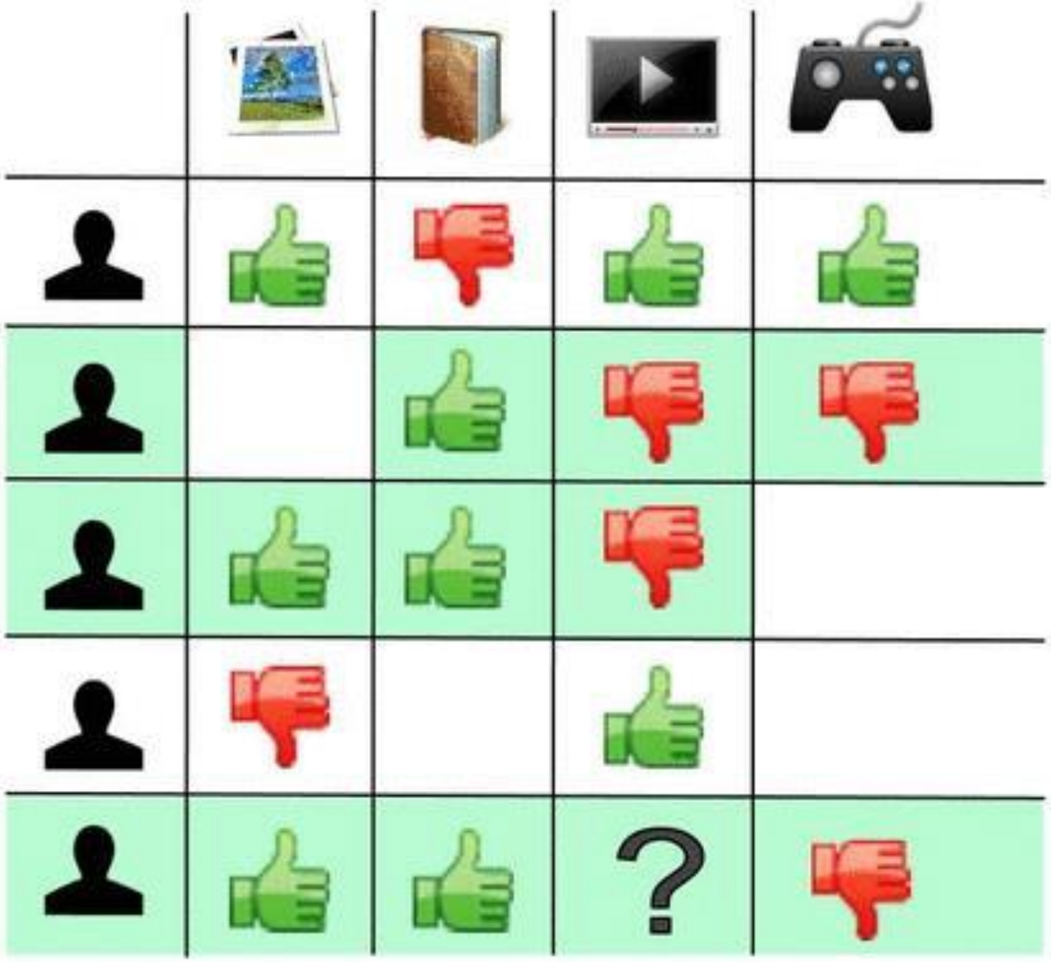
Recommendation by Regression

- **Pros:**
  - Reduces recommendations to a well-studied problem
  - Many good prediction algorithms available

- **Cons:**
  - Have to handle tons of missing data
  - Training M predictors is expensive

Very fast to train —> closets neighbors: ppl similar

KNN

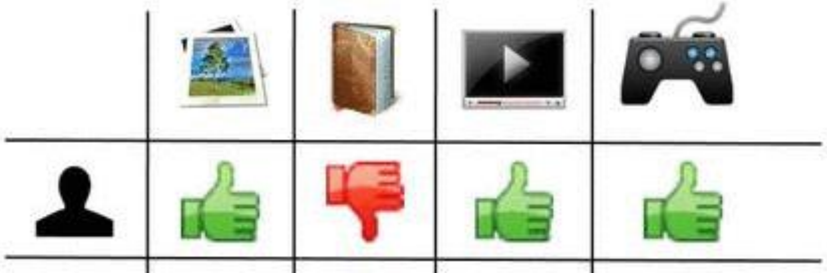


	🖼	📖	▶	🎮
👤	👍	👎	👍	👍
👤		👍	👎	👎
👤	👍	👍	👎	
👤	👎		👍	
👤	👍	👍	?	👎

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

KNN for Collaborative Filtering

- Widely used
- Item-based and User-based focus
- Represent each user as incomplete vector of item ratings
- Compute similarity between query user and all other users similarity = prozimity
- Find K most similar users who rated the query item
- Predict weighted average of ratings



Similarity Measures

- Pearson Correlation Coefficient
  - bound between 1 and -1
  - suffers from computing high similarity between users with few ratings in common

If too many missing values

  - set threshold for minimum number of co-rated itemssuffers from computing high similarity between users with few ratings in common

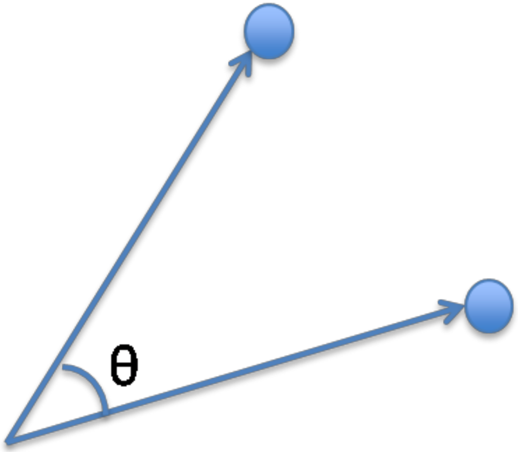
threshold: at least n common ratings.

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}}$$

Similarity Measures

- Cosine similarity
  - vector-space approach based on linear algebra
  - Unknown ratings are considered to be 0
  - this causes them to effectively drop out of the numerator

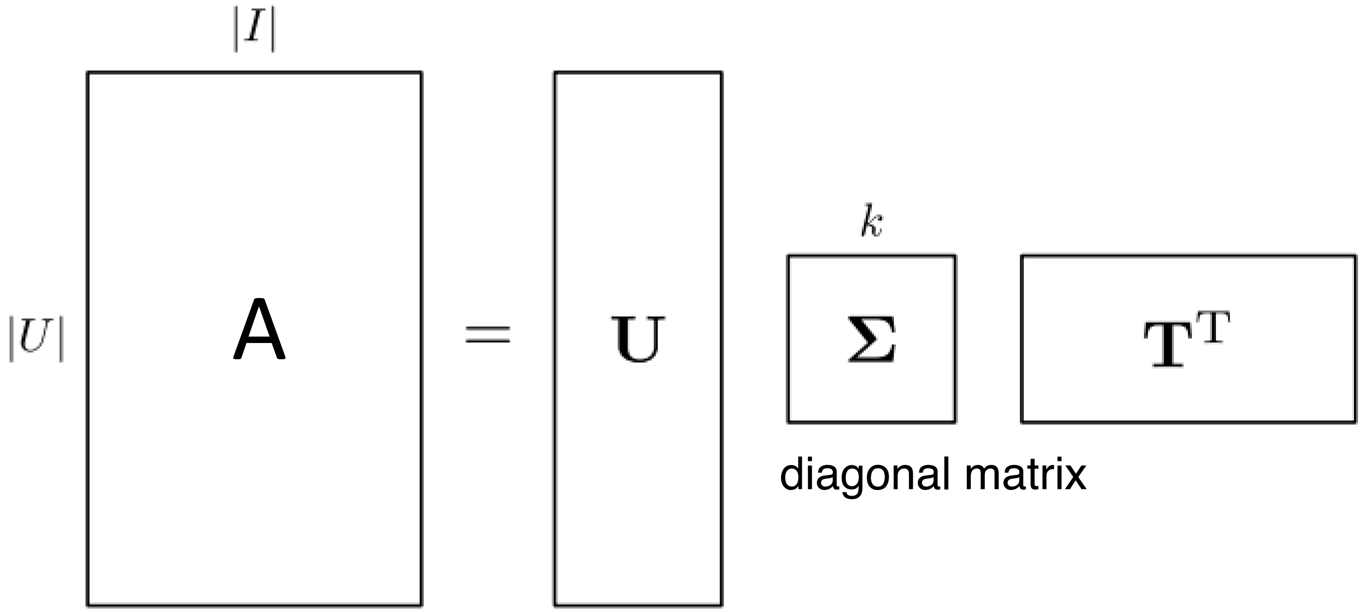
$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$



Netflix Prize

- Remember when we saw the Netflix prize video they mentioned SVD
- SimonFunk did this publicly on his blog with the title “Try this at home”
- http://sifter.org/~simon/journal/20061027.2.html

Singular Value Decomposition

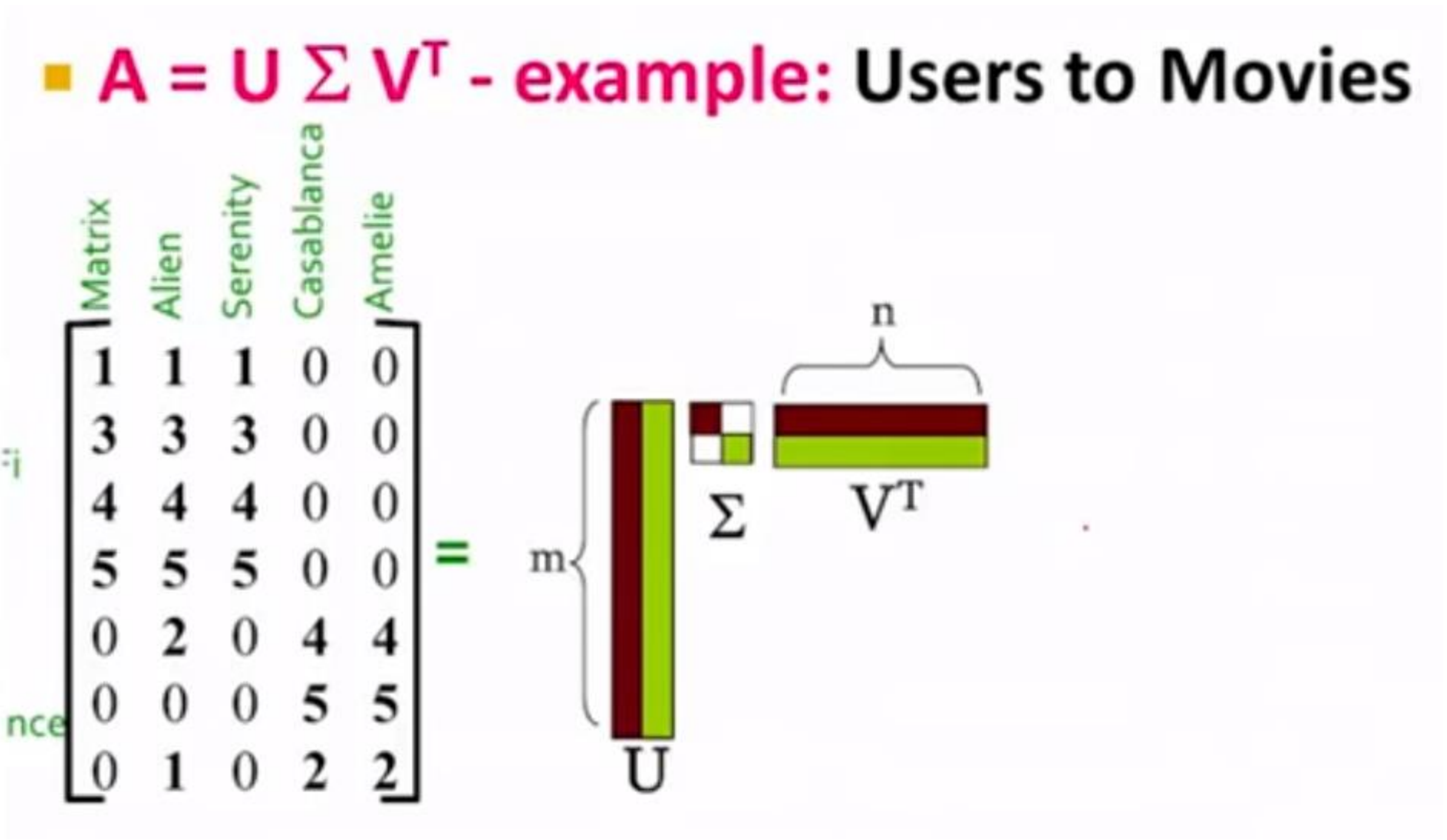


- If we know the SVD, we could compute the missing values in R.
- Try to infer SVD from matrix with missing data, and reconstruct full matrix R

Best SVD Explanation I have seen!

- Leskovec, Rajaraman, Ullman

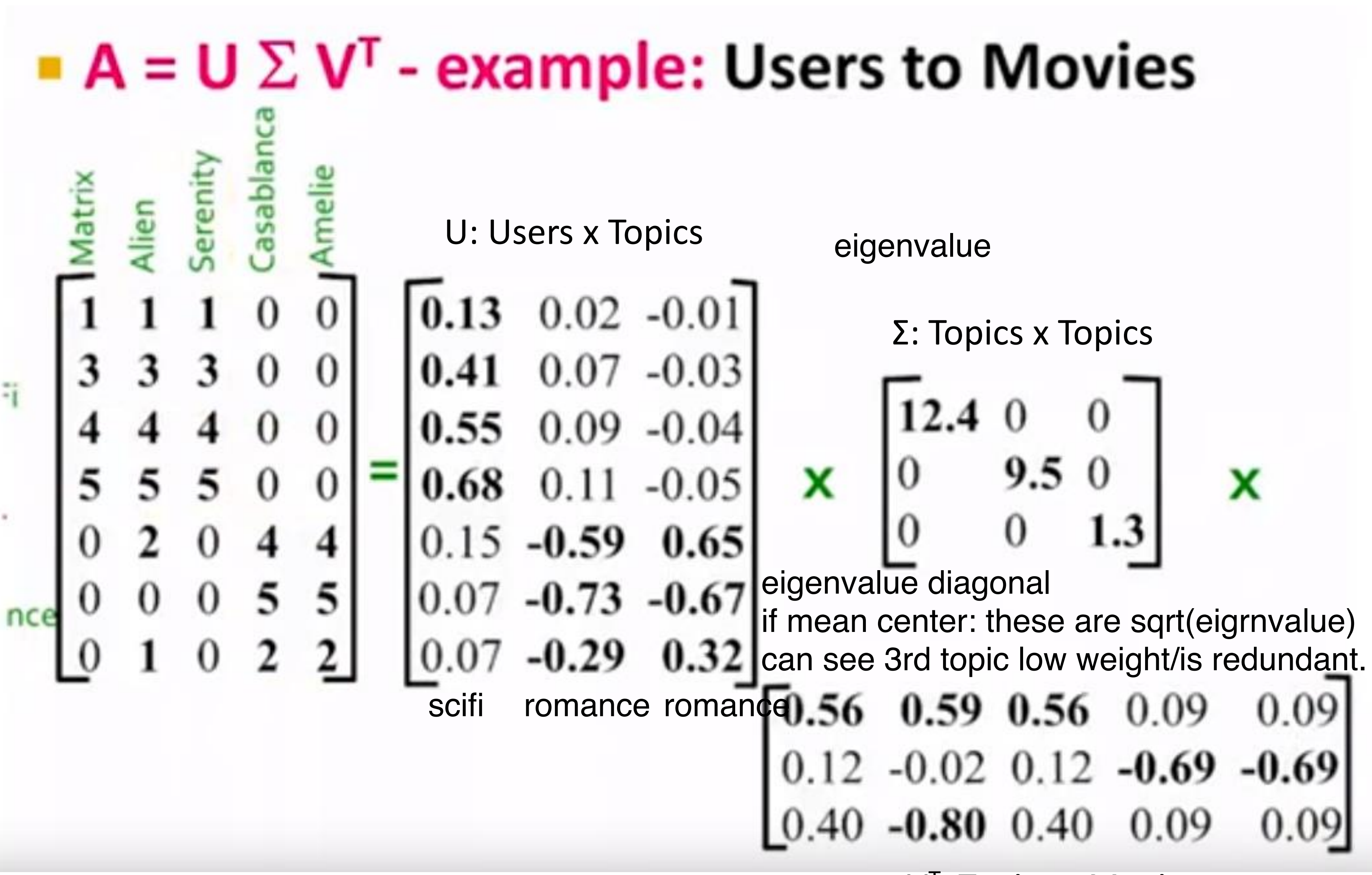
- https://www.youtube.com/watch?v=YKmkAoIUxkU



A = U Σ V^T - example: Users to Movies

$$\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ \left[\begin{matrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{matrix}\right. & & & & \left.\right] \end{array} = U \Sigma V^T$$

https://www.youtube.com/watch?v=YKmkAoIUxkU



■ A = U Σ V^T - example: Users to Movies

U: Users x Topics

eigenvalue

Σ: Topics x Topics

$$
\begin{array}{cccccc}
 & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\
 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\text{-i} & \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
 & \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
 & \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
 & 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
\text{nce} & 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
 & 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{array}
=
\begin{array}{ccc}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32} \\
\text{scifi} & \text{romance} & \text{romance}
\end{array}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

eigenvalue diagonal
if mean center: these are sqrt(eigrnvalue)
can see 3rd topic low weight/is redundant.

V^T: Topics x Movies

https://www.youtube.com/watch?v=YKmkAoIUxkU

SVD for Recommender Systems

- Not only good for estimating missing data
- We might actually care about the topics more

Find decomposition for incomplete A, then reconstruct A.

HW 4 Map Reduce
HW 5 Spark.

What is Map Reduce

- programming model
- addressing large data sets very large: pentabytes of data.
- parallel and distributed algorithms
- cluster framework

- It also is a way of thinking!

Map Reduce Background

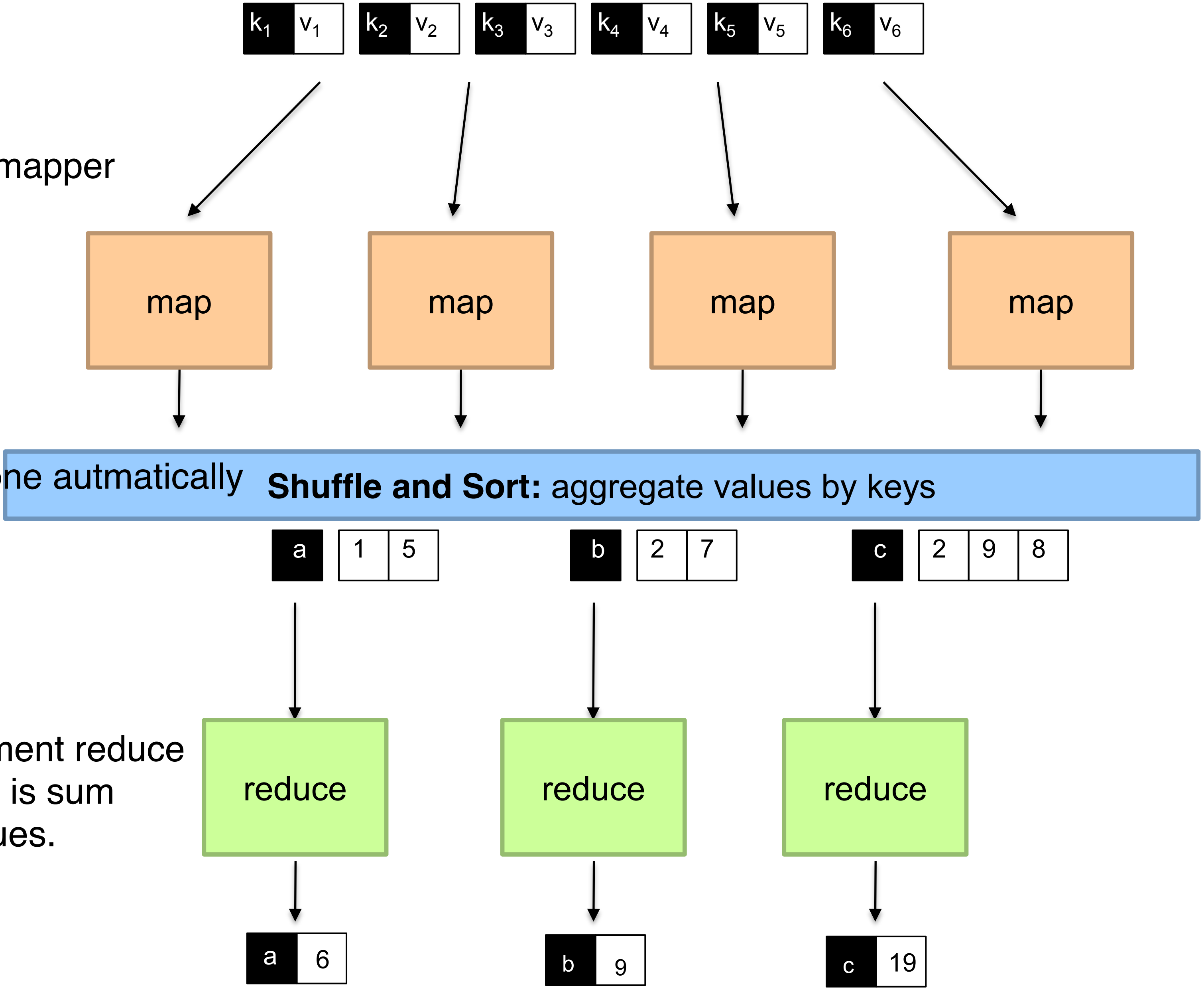
- Originally developed by Google
- Apache Hadoop is open source implementation in Java
- MrJob is a Python interface to Hadoop

  Does not run from an iPython notebook.

The Map and the Reduce

- Map:
  - performs filtering and sorting
- Reduce:
  - summary operation

key, value pair.

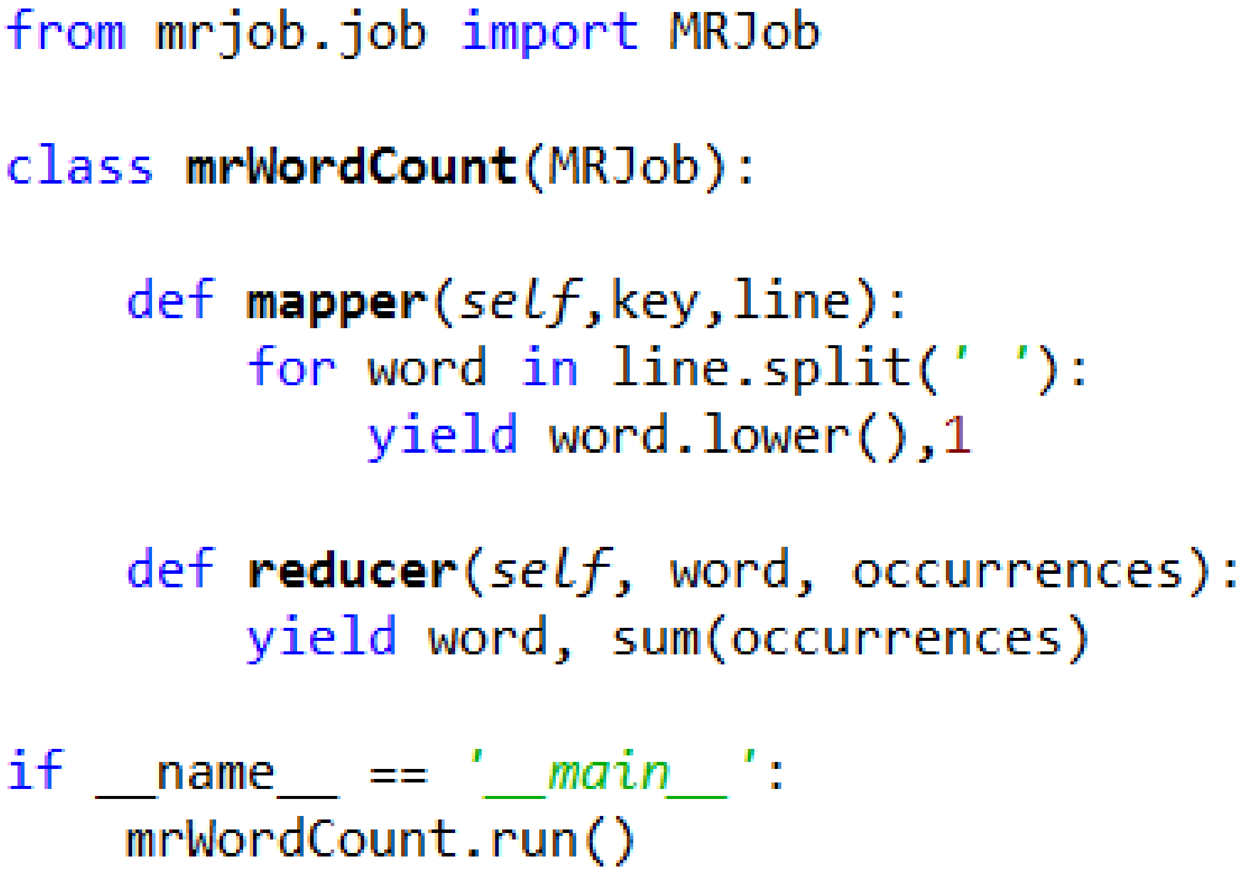


program mapper

shuffle done autmatically

we implement reduce
ie, reduce is sum
of the values.

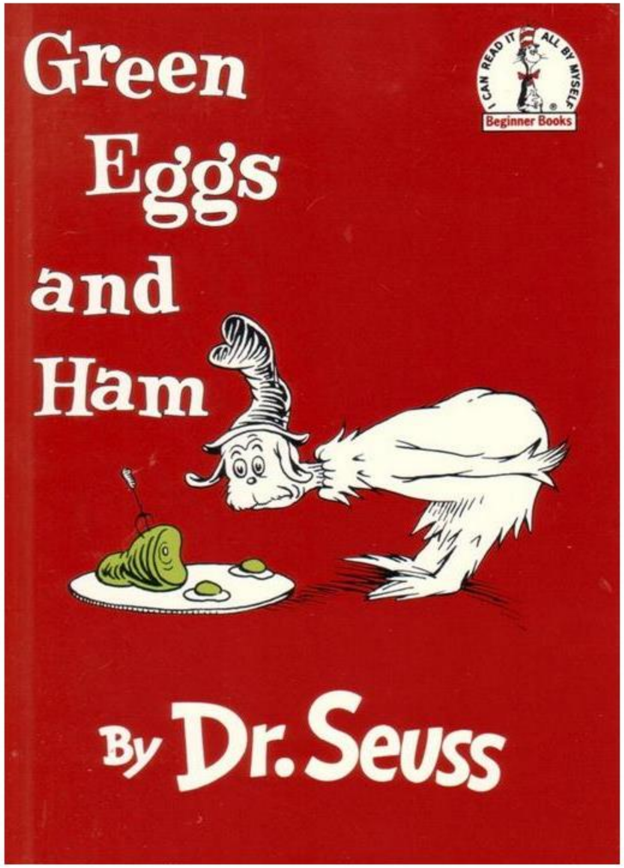
The Famous Word Count Example



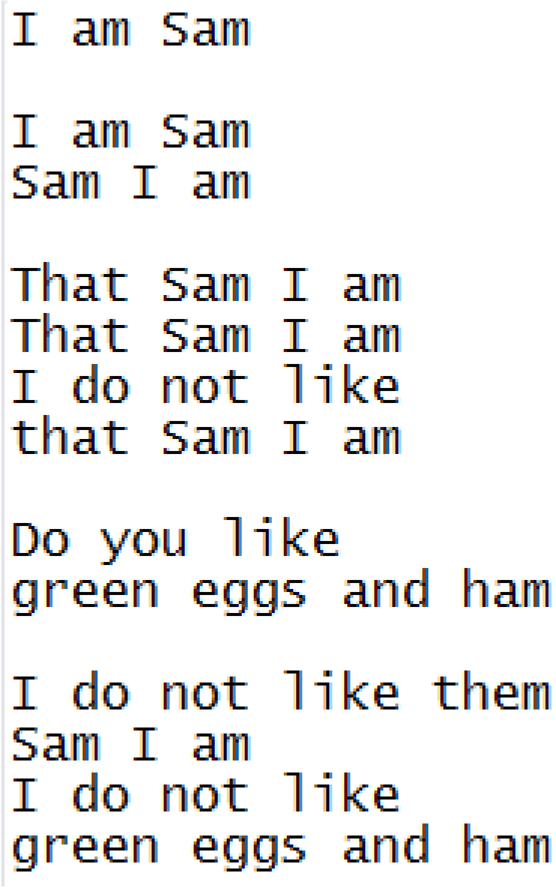
```python
from mrjob.job import MRJob

class mrWordCount(MRJob):

    def mapper(self,key,line):
        for word in line.split(' '):
            yield word.lower(),1

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

Green Eggs and Ham

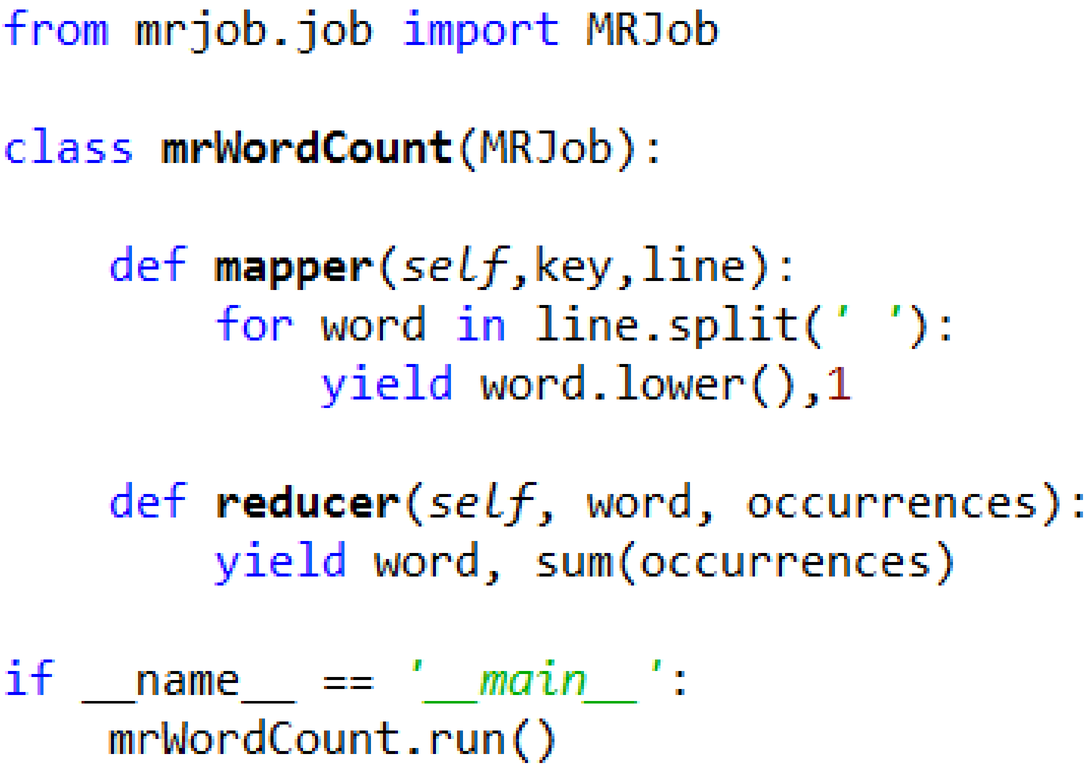
- Result of a bet:
- Can Dr. Seuss write a book using only 50 words?
- Bennett Cerf (Dr. Seuss's publisher) lost.
- It is the fourth best selling English-language children's hardcover book of all time.



Example Input File



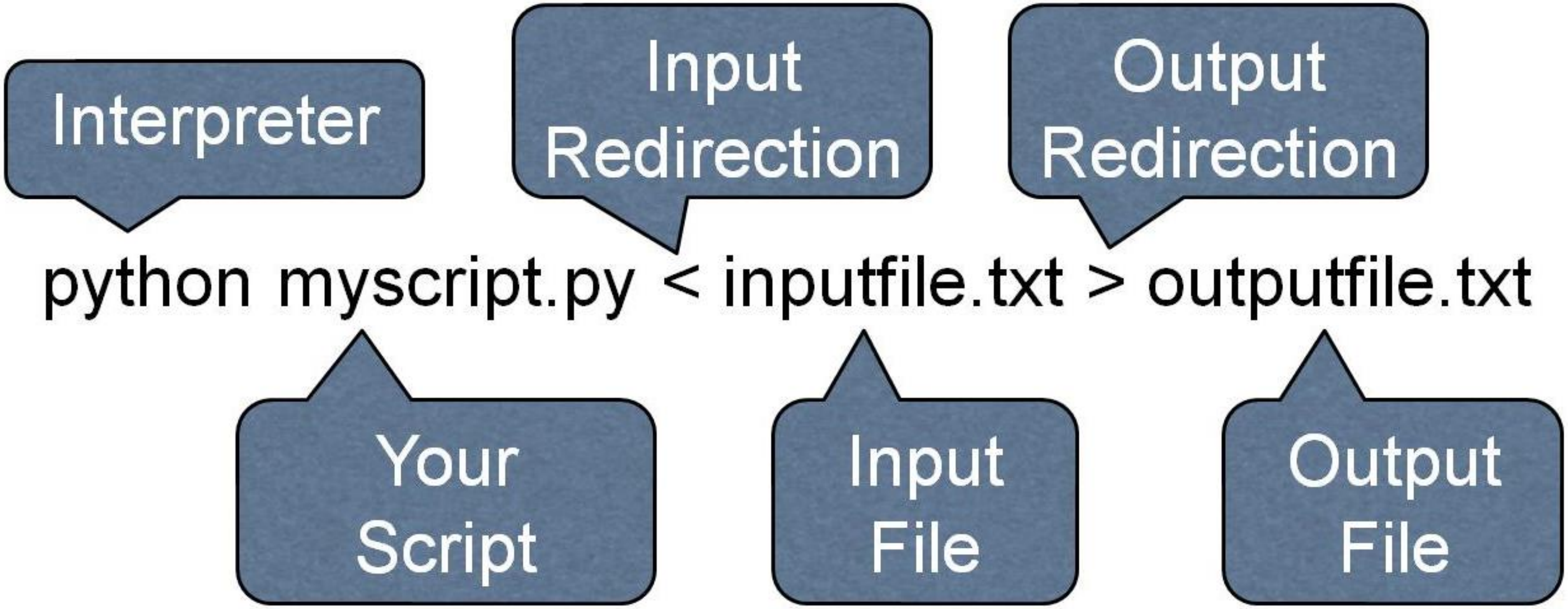
```
I am Sam

I am Sam
Sam I am

That Sam I am
That Sam I am
I do not like
that Sam I am

Do you like
green eggs and ham

I do not like them
Sam I am
I do not like
green eggs and ham
```



```
from mrjob.job import MRJob

class mrWordCount(MRJob):

    def mapper(self,key,line):
        for word in line.split(' '):
            yield word.lower(),1

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

Test files go in: mapper sees row by row.
Shuffle sort handles sorting and aggregating: reduced only sums.

Launching the Job



Interpreter

Input Redirection

Output Redirection

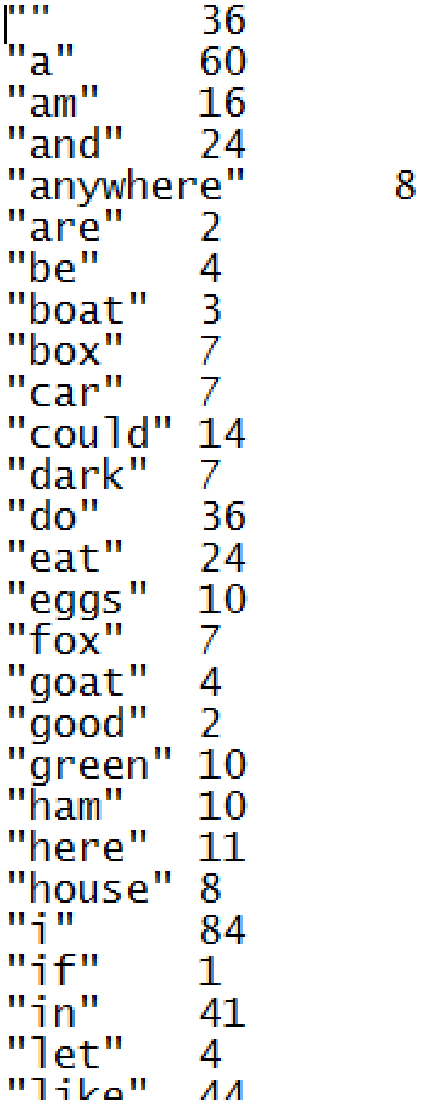
```
python myscript.py < inputfile.txt > outputfile.txt
```

Your Script

Input File

Output File

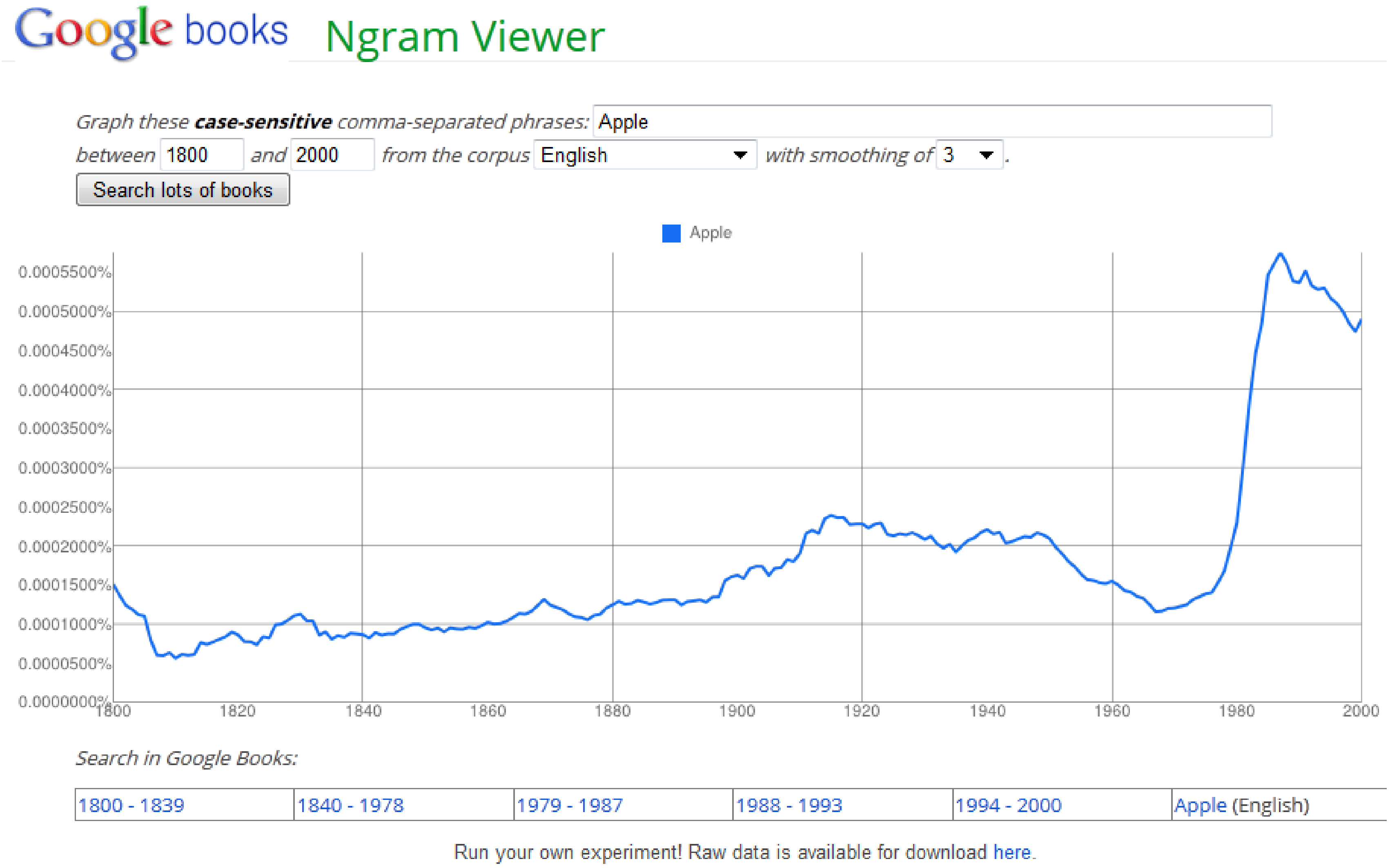
Output File



```
""      36
"a"     60
"am"    16
"and"   24
"anywhere"      8
"are"   2
"be"    4
"boat"  3
"box"   7
"car"   7
"could" 14
"dark"  7
"do"    36
"eat"   24
"eggs"  10
"fox"   7
"goat"  4
"good"  2
"green" 10
"ham"   10
"here"  11
"house" 8
"i"     84
"if"    1
"in"    41
"let"   4
"like"  44
```

This word count on all of google books.

Culturomics



Google books Ngram Viewer

*Graph these **case-sensitive** comma-separated phrases:* Apple

between 1800 *and* 2000 *from the corpus* English *with smoothing of* 3 .

Search in Google Books:

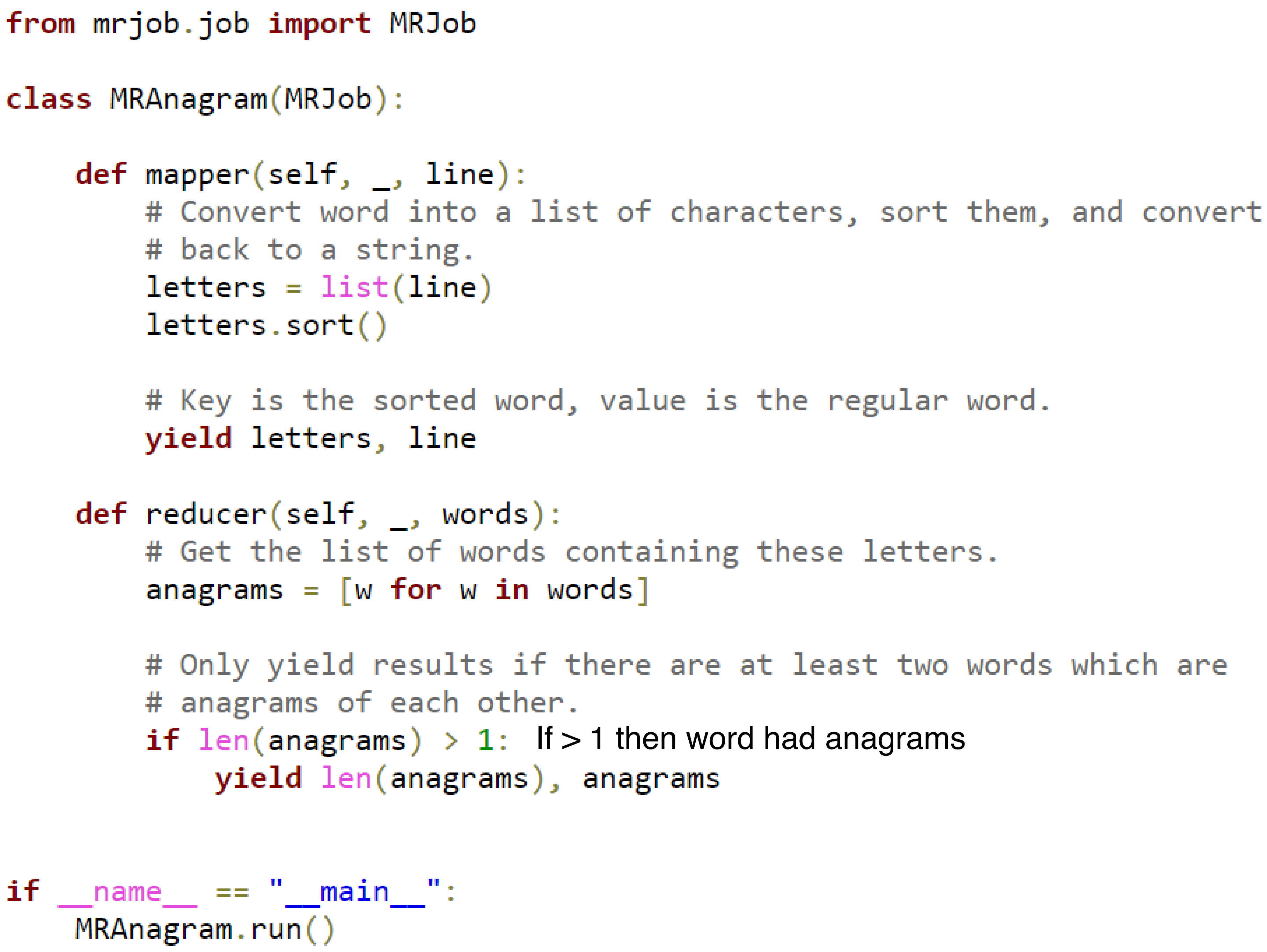
1800 - 1839	1840 - 1978	1979 - 1987	1988 - 1993	1994 - 2000	Apple (English)

Run your own experiment! Raw data is available for download here.

Anagram Finder

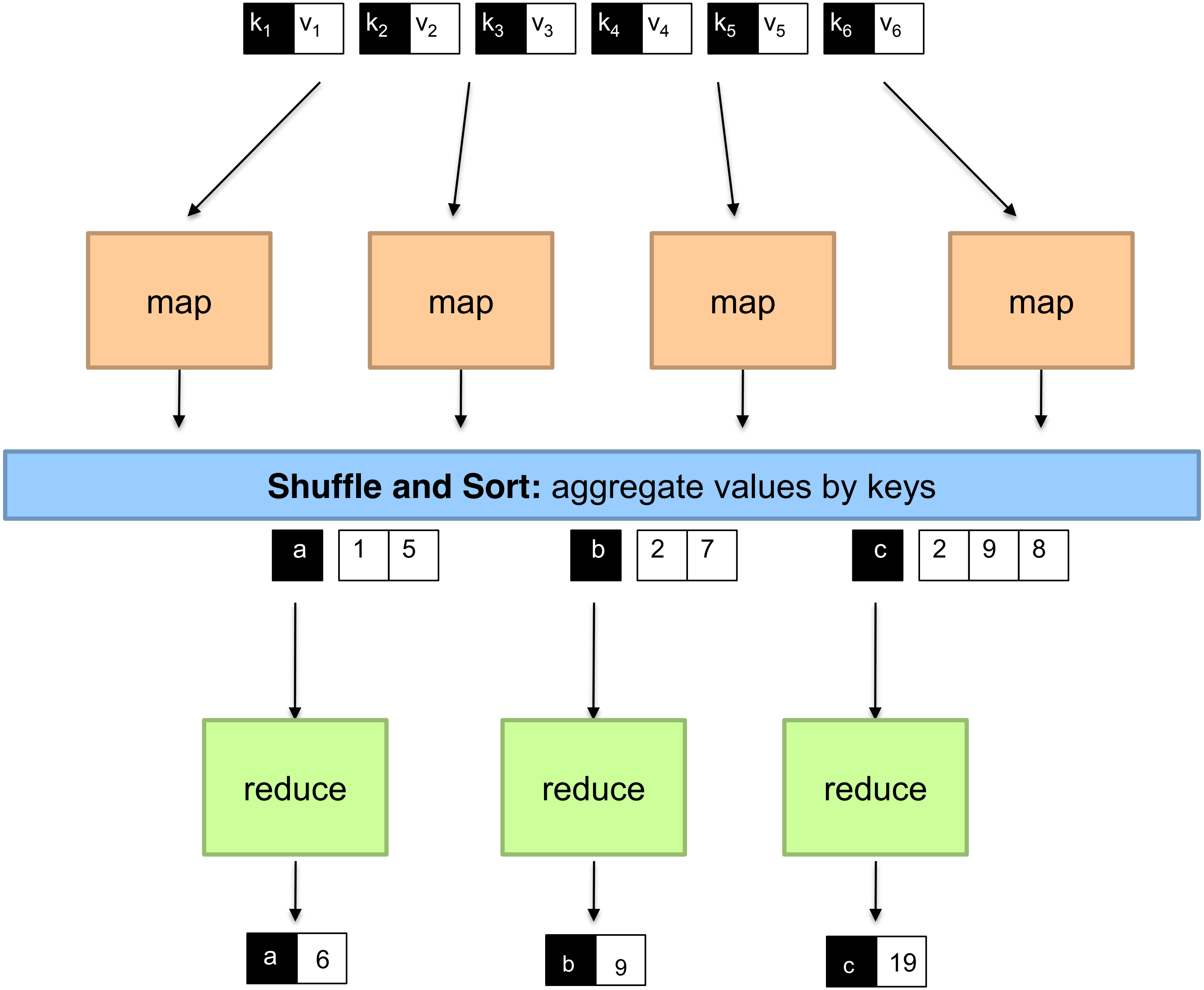
- Anagram: Words or phrases consisting of the same letters
- Examples:
  - Dormitory – Dirty room
  - Astronomer – Moon starer
  - Election results – Lies let’s recount

- Verifying anagrams with map reduce
- Input: file with one word per line



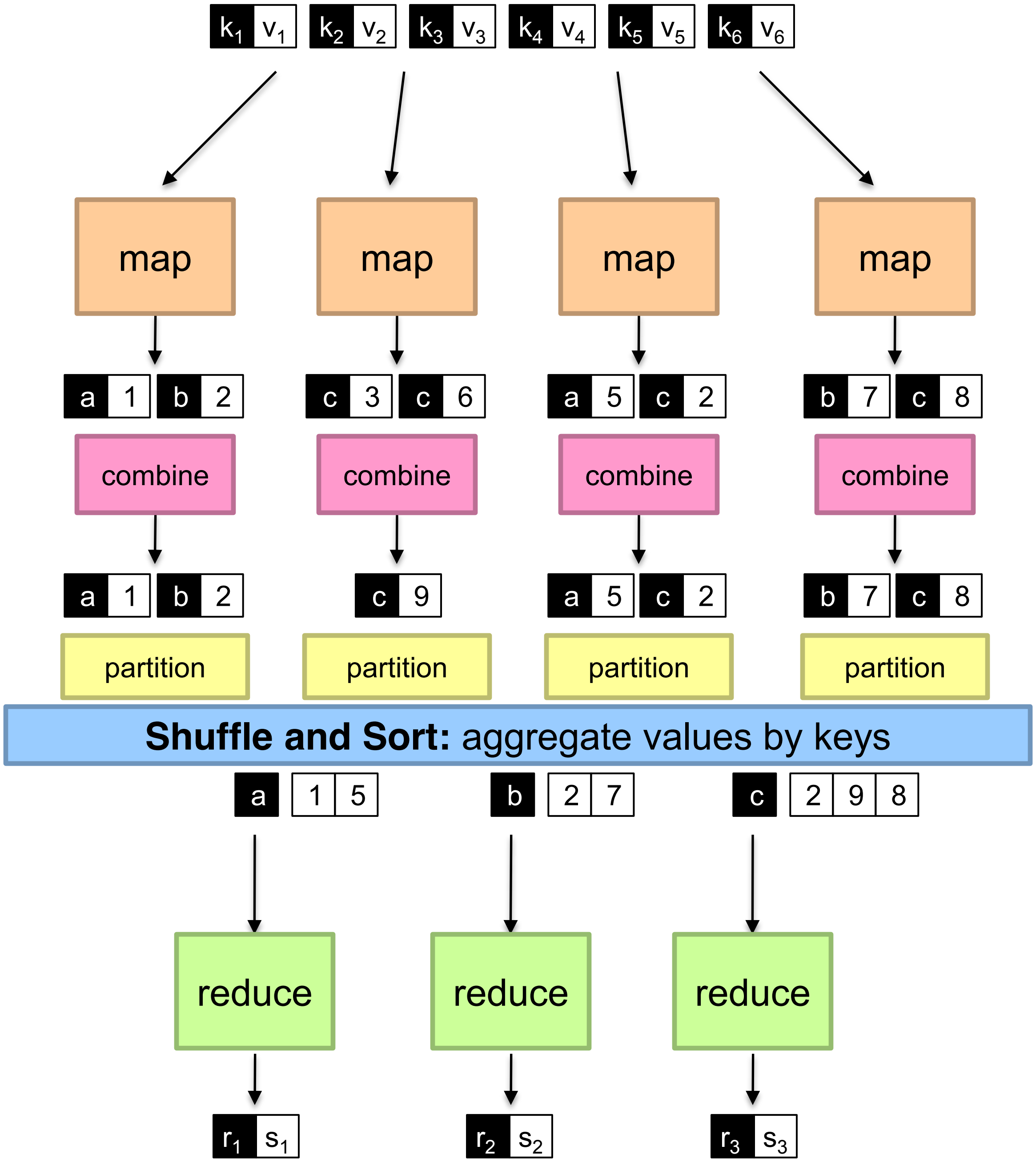
```python
from mrjob.job import MRJob

class MRAnagram(MRJob):

    def mapper(self, _, line):
        # Convert word into a list of characters, sort them, and convert
        # back to a string.
        letters = list(line)
        letters.sort()

        # Key is the sorted word, value is the regular word.
        yield letters, line

    def reducer(self, _, words):
        # Get the list of words containing these letters.
        anagrams = [w for w in words]

        # Only yield results if there are at least two words which are
        # anagrams of each other.
        if len(anagrams) > 1:
            yield len(anagrams), anagrams


if __name__ == "__main__":
    MRAnagram.run()
```

If > 1 then word had anagrams

k_1 v_1
k_2 v_2
k_3 v_3
k_4 v_4
k_5 v_5
k_6 v_6
map
map
map
map
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 9 8
reduce
reduce
reduce
a 6
b 9
c 19

Importance of Local Aggregation

- Ideal scaling characteristics:
  - Twice the data, twice the running time
  - Twice the resources, half the running time
- Why can’t we achieve this?
  - Synchronization requires communication
  - Communication kills performance
- Thus… avoid communication!
  - Reduce intermediate data via local aggregation
  - Two possibilities:
    - Combiners
    - In-mapper combining



k_1 v_1 k_2 v_2 k_3 v_3 k_4 v_4 k_5 v_5 k_6 v_6

map map map map

a 1 b 2 c 3 c 6 a 5 c 2 b 7 c 8

combine combine combine combine

a 1 b 2 c 9 a 5 c 2 b 7 c 8

partition partition partition partition

Shuffle and Sort: aggregate values by keys

a 1 5 b 2 7 c 2 9 8

reduce reduce reduce

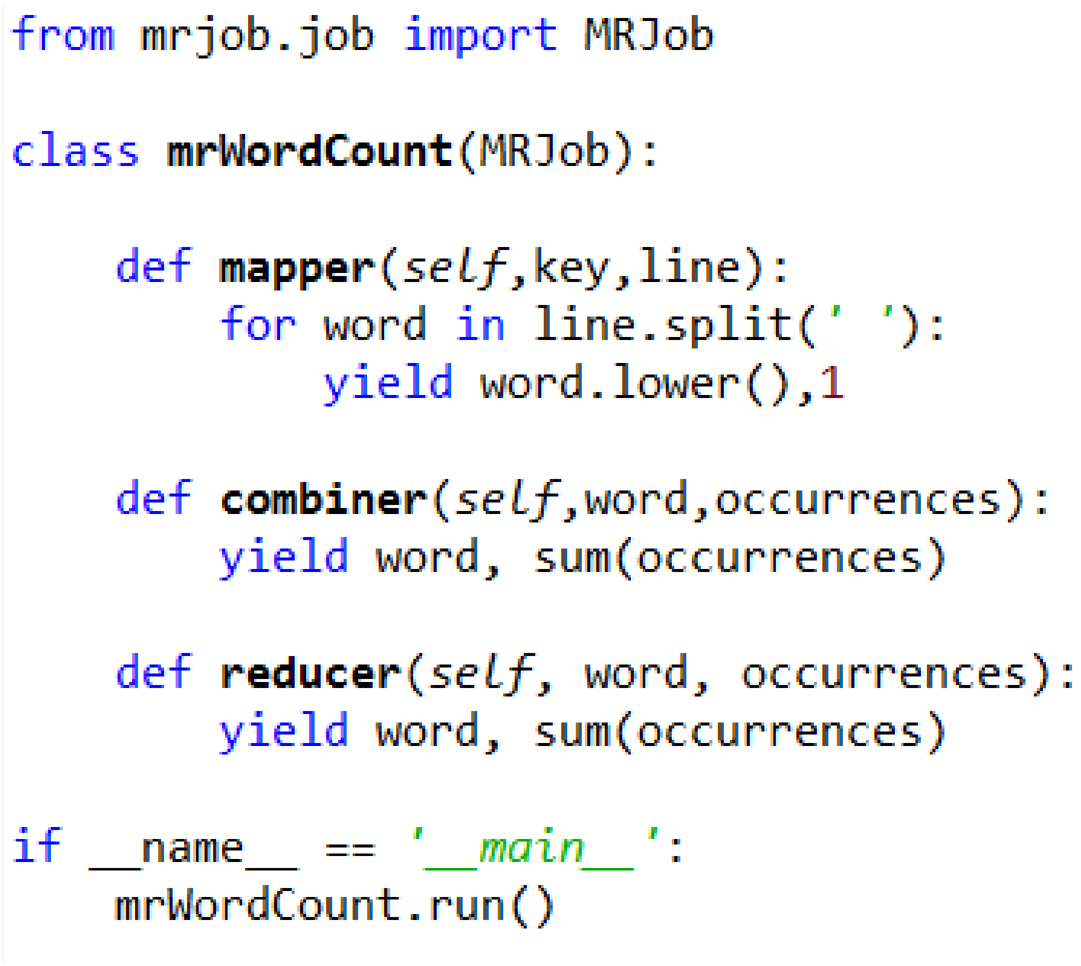
r_1 s_1 r_2 s_2 r_3 s_3

Combiner

- “mini-reducers”
- Takes mapper output before shuffle and sort
- Can significantly reduce network traffic
- No access to other mappers
- Not guaranteed to get all values for a key
- Not guaranteed to run at all!
- Key and value output must match mapper

Why does the key and value output have to match the mapper output?

Word Count with Combiner

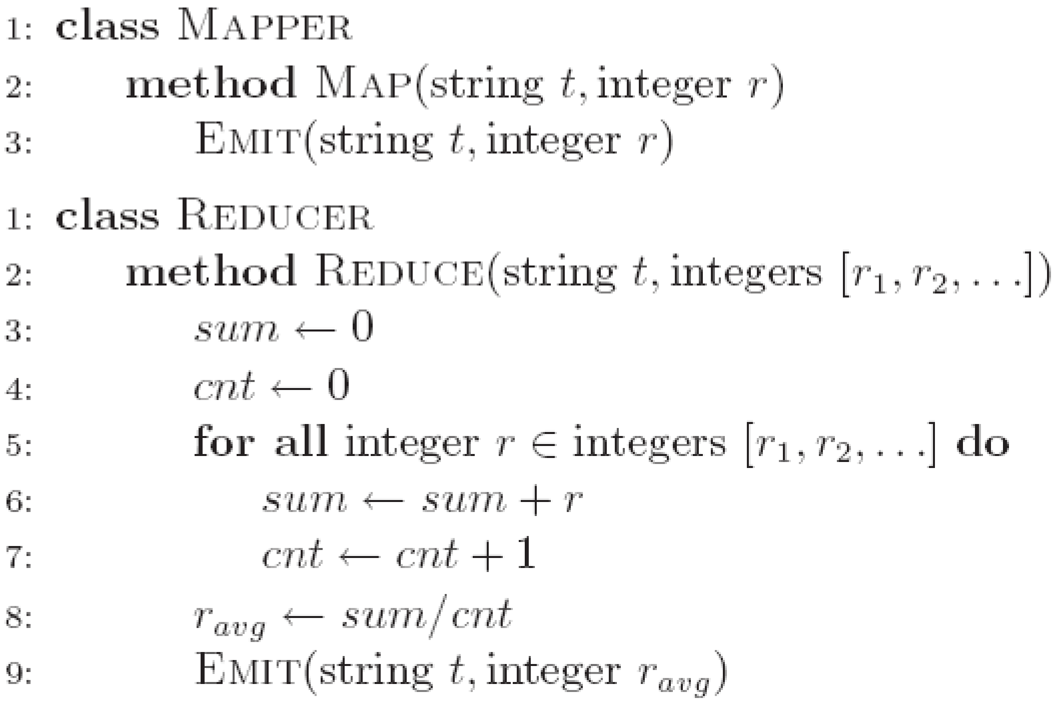


```
from mrjob.job import MRJob

class mrWordCount(MRJob):

    def mapper(self,key,line):
        for word in line.split(' '):
            yield word.lower(),1

    def combiner(self,word,occurrences):
        yield word, sum(occurrences)

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

Combiner Design

- Combiners and reducers share same method signature
  - Sometimes, reducers can serve as combiners
  - Often, not…
- Remember: combiners are optional optimizations
  - Should not affect algorithm correctness
  - May be run 0, 1, or multiple times
- Example: find average of all integers associated with the same key

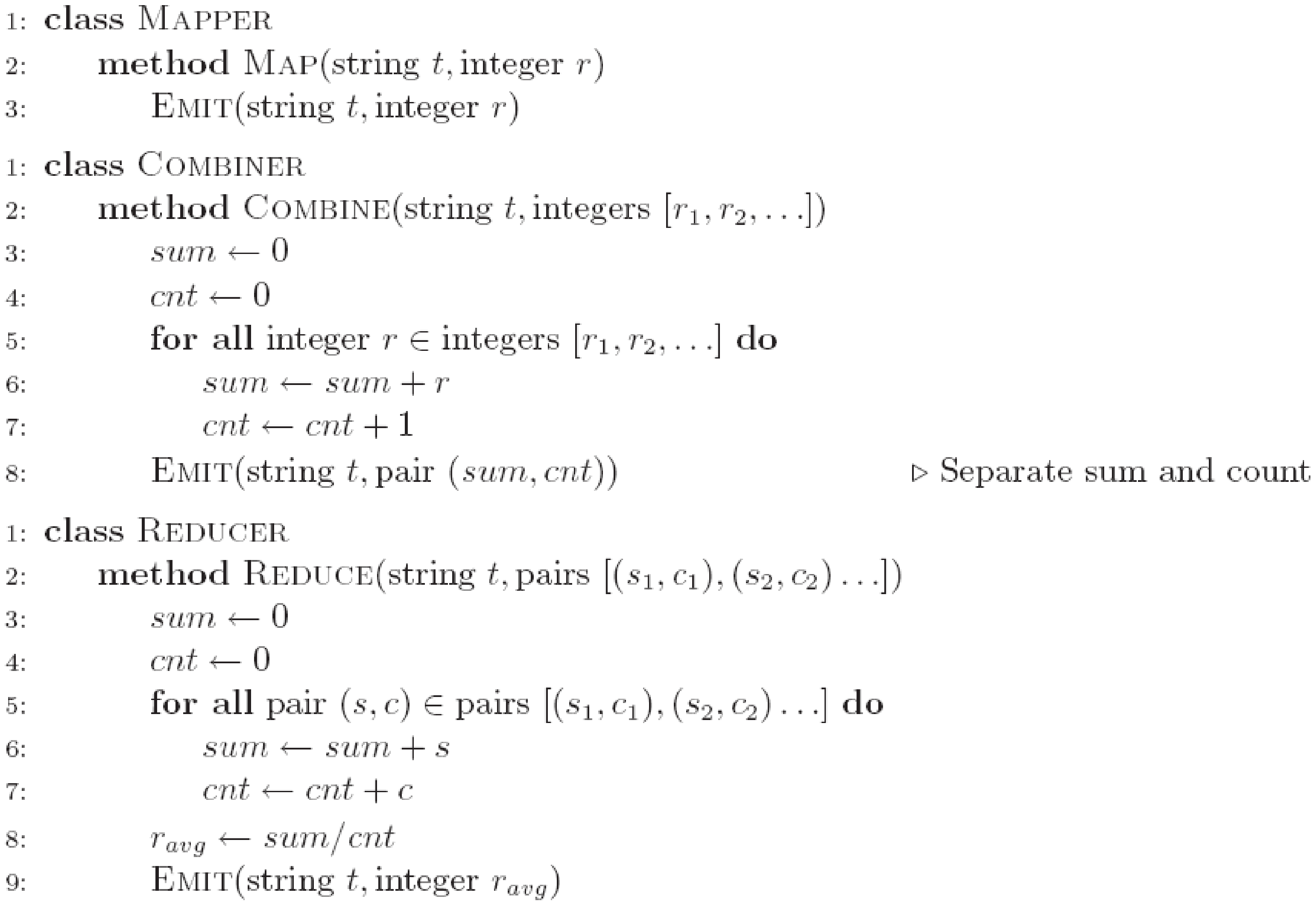
Computing the Mean: Version 1



```
class Mapper
    method Map(string $t$, integer $r$)
        Emit(string $t$, integer $r$)

class Reducer
    method Reduce(string $t$, integers $[r_1, r_2, \ldots]$)
        $sum \leftarrow 0$
        $cnt \leftarrow 0$
        for all integer $r \in$ integers $[r_1, r_2, \ldots]$ do
            $sum \leftarrow sum + r$
            $cnt \leftarrow cnt + 1$
        $r_{avg} \leftarrow sum/cnt$
        Emit(string $t$, integer $r_{avg}$)
```

Why can’t we use reducer as combiner?

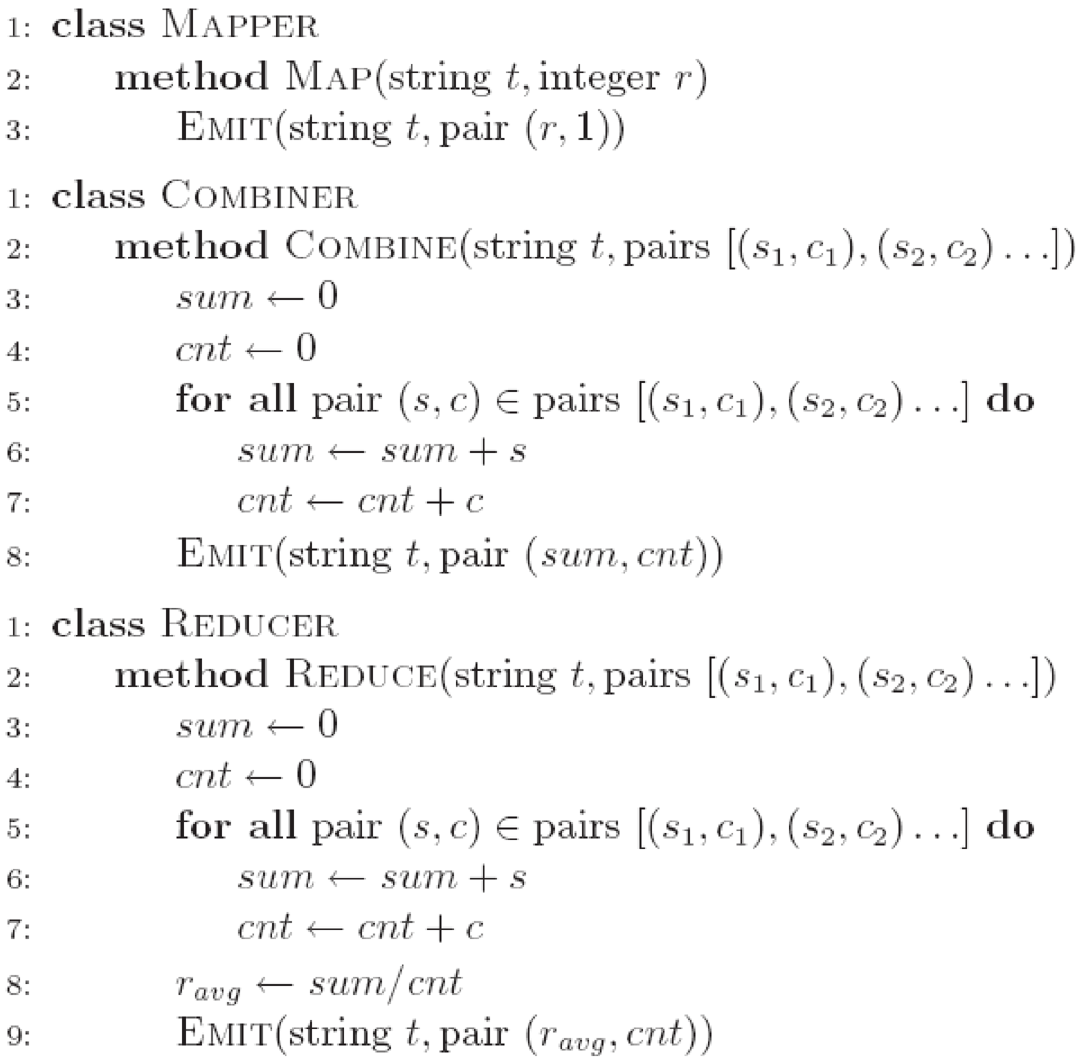
Computing the Mean: Version 2



```
class Mapper
    method Map(string $t$, integer $r$)
        Emit(string $t$, integer $r$)

class Combiner
    method Combine(string $t$, integers $[r_1, r_2, \ldots]$)
        $sum \leftarrow 0$
        $cnt \leftarrow 0$
        for all integer $r \in$ integers $[r_1, r_2, \ldots]$ do
            $sum \leftarrow sum + r$
            $cnt \leftarrow cnt + 1$
        Emit(string $t$, pair $(sum, cnt)$)                    ▷ Separate sum and count

class Reducer
    method Reduce(string $t$, pairs $[(s_1, c_1), (s_2, c_2) \ldots]$)
        $sum \leftarrow 0$
        $cnt \leftarrow 0$
        for all pair $(s, c) \in$ pairs $[(s_1, c_1), (s_2, c_2) \ldots]$ do
            $sum \leftarrow sum + s$
            $cnt \leftarrow cnt + c$
        $r_{avg} \leftarrow sum/cnt$
        Emit(string $t$, integer $r_{avg}$)
```

Why doesn’t this work?

Computing the Mean: Version 3



```
class Mapper
    method Map(string $t$, integer $r$)
        Emit(string $t$, pair $(r, 1)$)
```

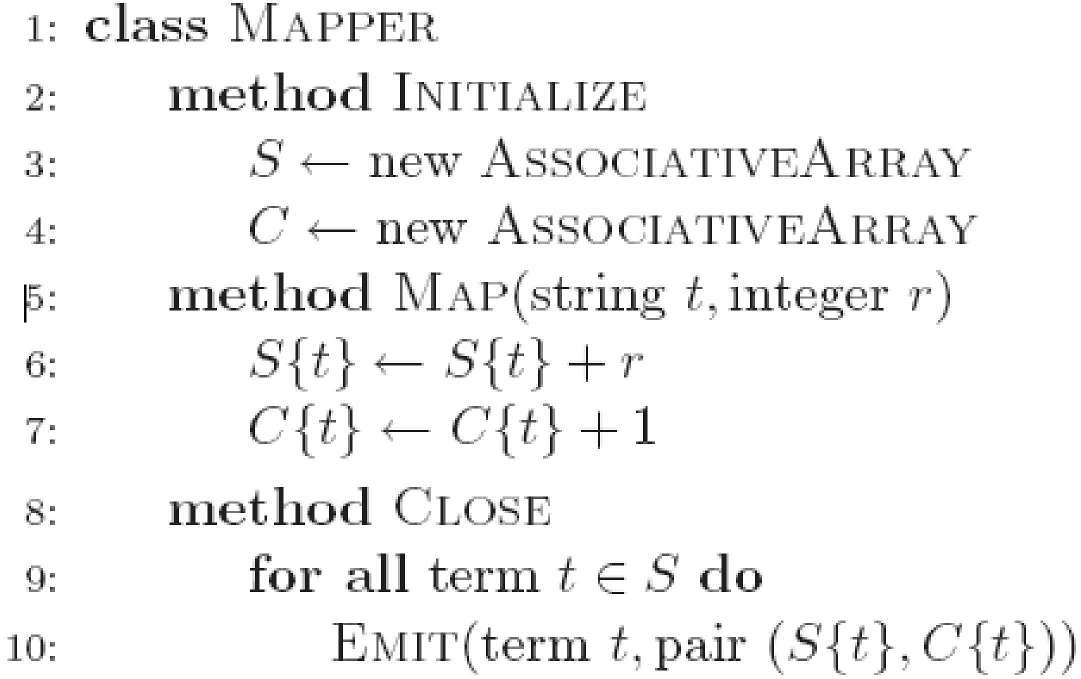
```
class Combiner
    method Combine(string $t$, pairs $[(s_1, c_1), (s_2, c_2) \ldots]$)
        $sum \leftarrow 0$
        $cnt \leftarrow 0$
        for all pair $(s, c) \in$ pairs $[(s_1, c_1), (s_2, c_2) \ldots]$ do
            $sum \leftarrow sum + s$
            $cnt \leftarrow cnt + c$
        Emit(string $t$, pair $(sum, cnt)$)
```

```
class Reducer
    method Reduce(string $t$, pairs $[(s_1, c_1), (s_2, c_2) \ldots]$)
        $sum \leftarrow 0$
        $cnt \leftarrow 0$
        for all pair $(s, c) \in$ pairs $[(s_1, c_1), (s_2, c_2) \ldots]$ do
            $sum \leftarrow sum + s$
            $cnt \leftarrow cnt + c$
        $r_{avg} \leftarrow sum/cnt$
        Emit(string $t$, pair $(r_{avg}, cnt)$)
```

Fixed? What if combiner does not run?

In-Mapper Combining

- “Fold the functionality of the combiner into the mapper by preserving state across multiple map calls

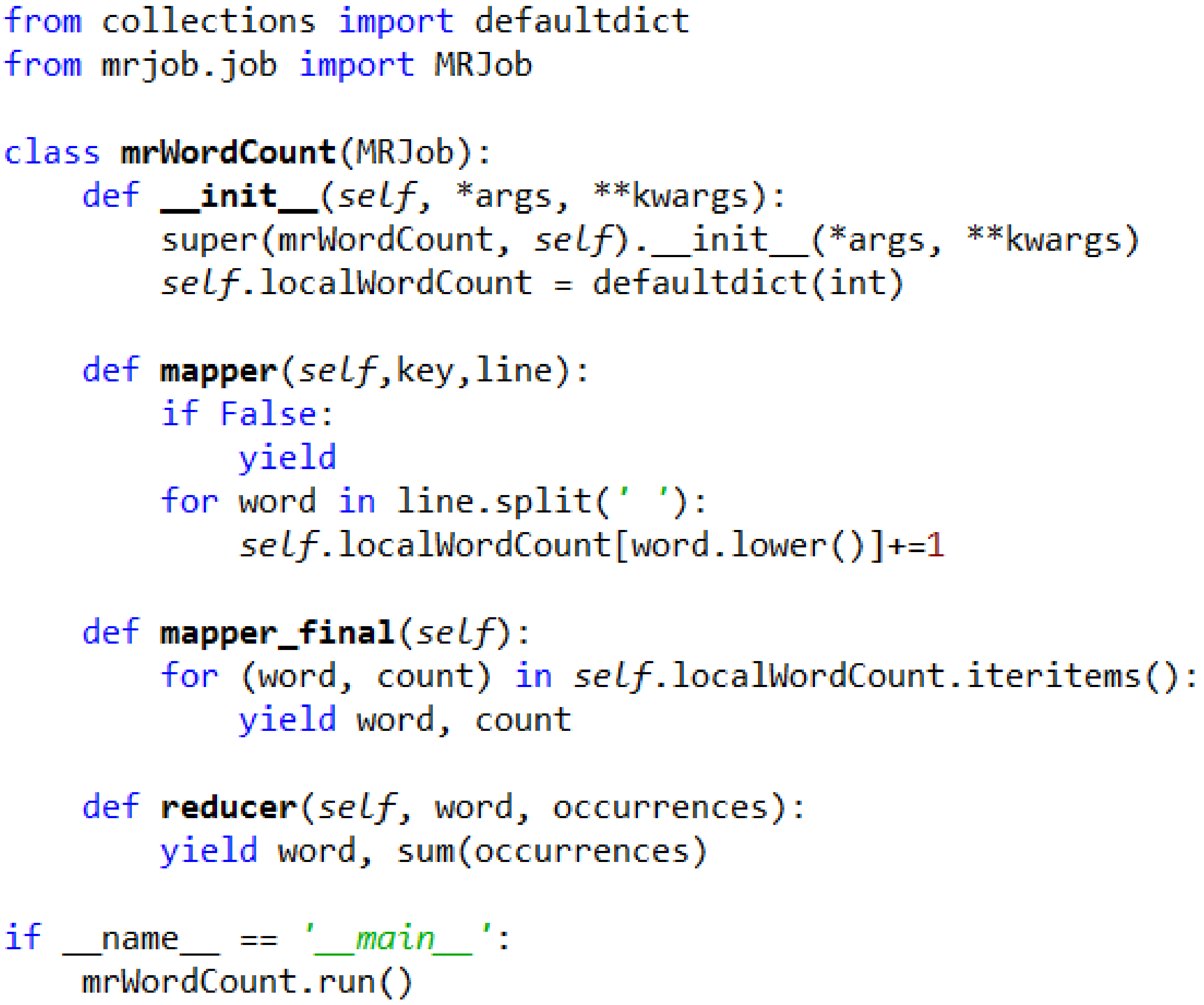


```
class Mapper
    method Initialize
        $S \leftarrow$ new AssociativeArray
        $C \leftarrow$ new AssociativeArray
    method Map(string $t$, integer $r$)
        $S\{t\} \leftarrow S\{t\} + r$
        $C\{t\} \leftarrow C\{t\} + 1$
    method Close
        for all term $t \in S$ do
            Emit(term $t$, pair $(S\{t\}, C\{t\})$)
```

In-Mapper Combining

- Advantages
  - Speed
  - Why is this faster than actual combiners?
- Disadvantages
  - Explicit memory management required
  - Potential for order-dependent bugs

Word Count with In-Mapper-Comb.



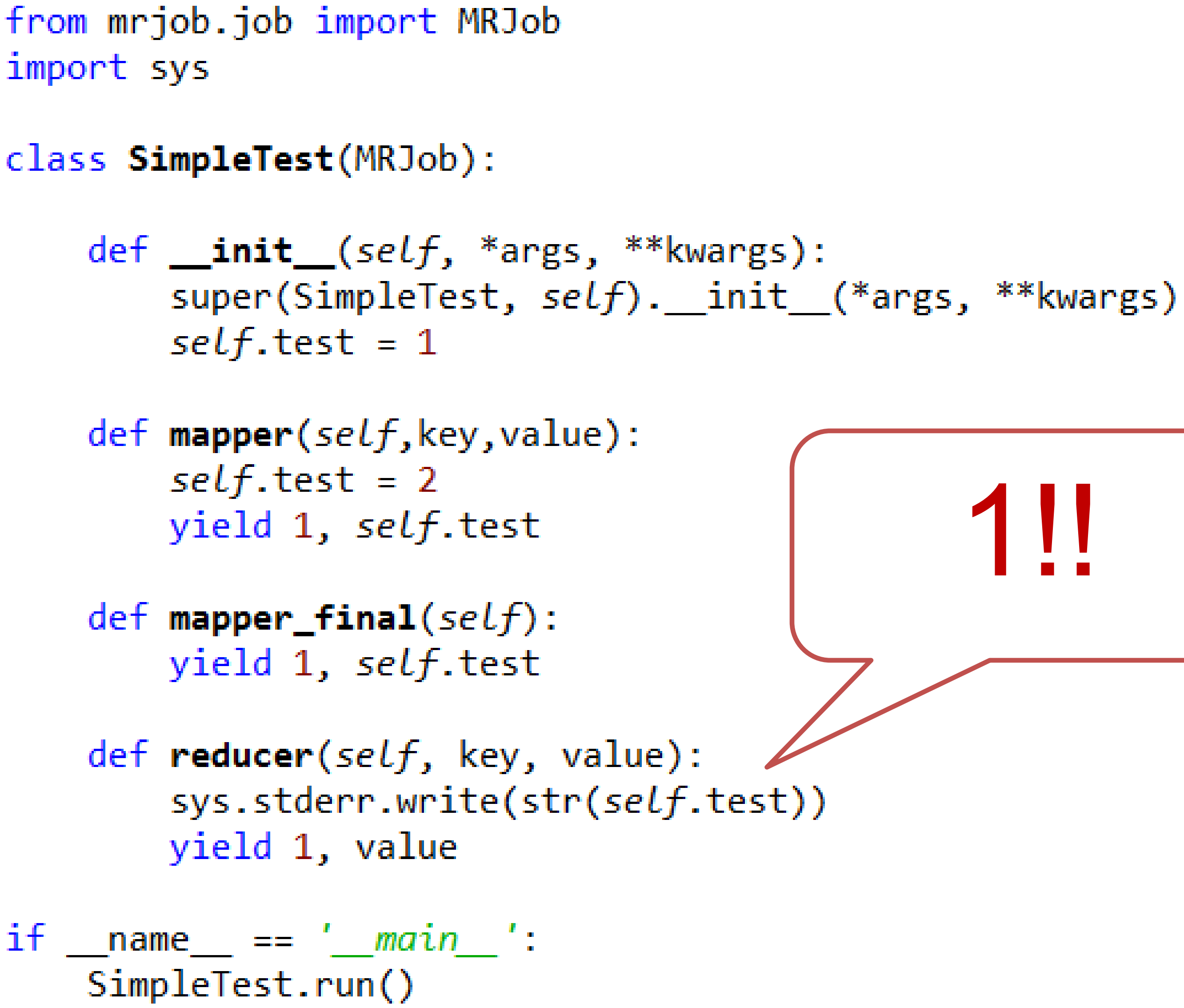
```
from collections import defaultdict
from mrjob.job import MRJob

class mrWordCount(MRJob):
    def __init__(self, *args, **kwargs):
        super(mrWordCount, self).__init__(*args, **kwargs)
        self.localWordCount = defaultdict(int)

    def mapper(self,key,line):
        if False:
            yield
        for word in line.split(' '):
            self.localWordCount[word.lower()]+=1

    def mapper_final(self):
        for (word, count) in self.localWordCount.iteritems():
            yield word, count

    def reducer(self, word, occurrences):
        yield word, sum(occurrences)

if __name__ == '__main__':
    mrWordCount.run()
```

Which is better?

- For large dictionaries?
  - Combiner has no memory problems

- For skewed word distributions (“the”)?
  - In-mapper reduces load on reducer

Word of Caution



```
from mrjob.job import MRJob
import sys

class SimpleTest(MRJob):

    def __init__(self, *args, **kwargs):
        super(SimpleTest, self).__init__(*args, **kwargs)
        self.test = 1

    def mapper(self,key,value):
        self.test = 2
        yield 1, self.test

    def mapper_final(self):
        yield 1, self.test

    def reducer(self, key, value):
        sys.stderr.write(str(self.test))
        yield 1, value

if __name__ == '__main__':
    SimpleTest.run()
```

1!!

Pairs and Stripes:

- Term co-occurrence matrix for a text collection
  - M = N x N matrix (N = vocabulary size)
  - M_{ij}: number of times *i* and *j* co-occur in some context
  - Context can be a sentence, sequence of m words, etc.
  - In this case co-occurrence matrix is symmetric

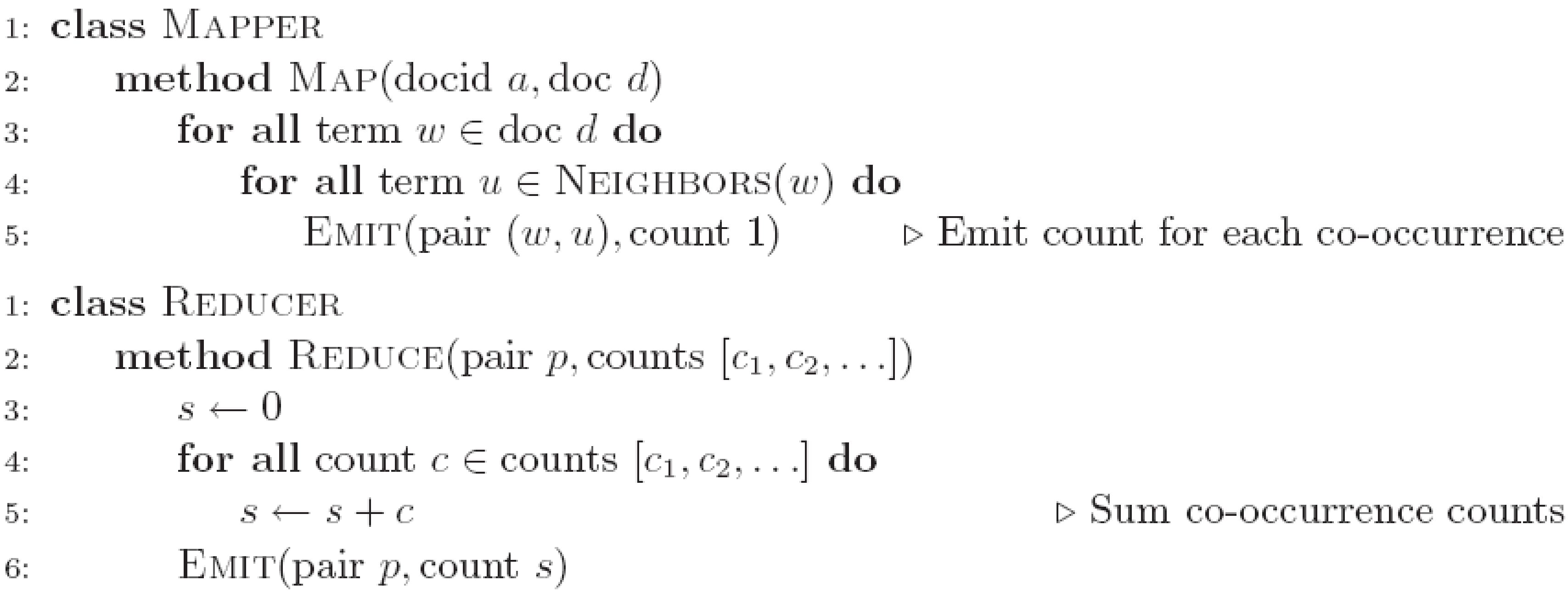
MapReduce: Large Counting Problems

- Term co-occurrence matrix for a text collection = specific instance of a large counting problem
  - A large event space (number of terms)
  - A large number of observations (the collection itself)
  - Goal: keep track of interesting statistics about the events
- Basic approach
  - Mappers generate partial counts
  - Reducers aggregate partial counts

First Try: “Pairs”

- Each mapper takes a sentence:
  - Generate all co-occurring term pairs
  - For all pairs, emit (a, b) → count
- Reducers sum up counts associated with these pairs
- Use combiners!

Pairs: Pseudo-Code



```
class Mapper
    method Map(docid $a$, doc $d$)
        for all term $w \in$ doc $d$ do
            for all term $u \in$ Neighbors($w$) do
                Emit(pair $(w, u)$, count 1)        ▷ Emit count for each co-occurrence
class Reducer
    method Reduce(pair $p$, counts $[c_1, c_2, \ldots]$)
        $s \leftarrow 0$
        for all count $c \in$ counts $[c_1, c_2, \ldots]$ do
            $s \leftarrow s + c$                    ▷ Sum co-occurrence counts
        Emit(pair $p$, count $s$)
```

“Pairs” Analysis

- Advantages
  - Easy to implement, easy to understand
- Disadvantages
  - Lots of pairs to sort and shuffle around
  - Not many opportunities for combiners to work

Another Try: “Stripes”

- Idea: group together pairs into an associative array

(a, b) → 1
(a, c) → 2
(a, d) → 5
(a, e) → 3
(a, f) → 2

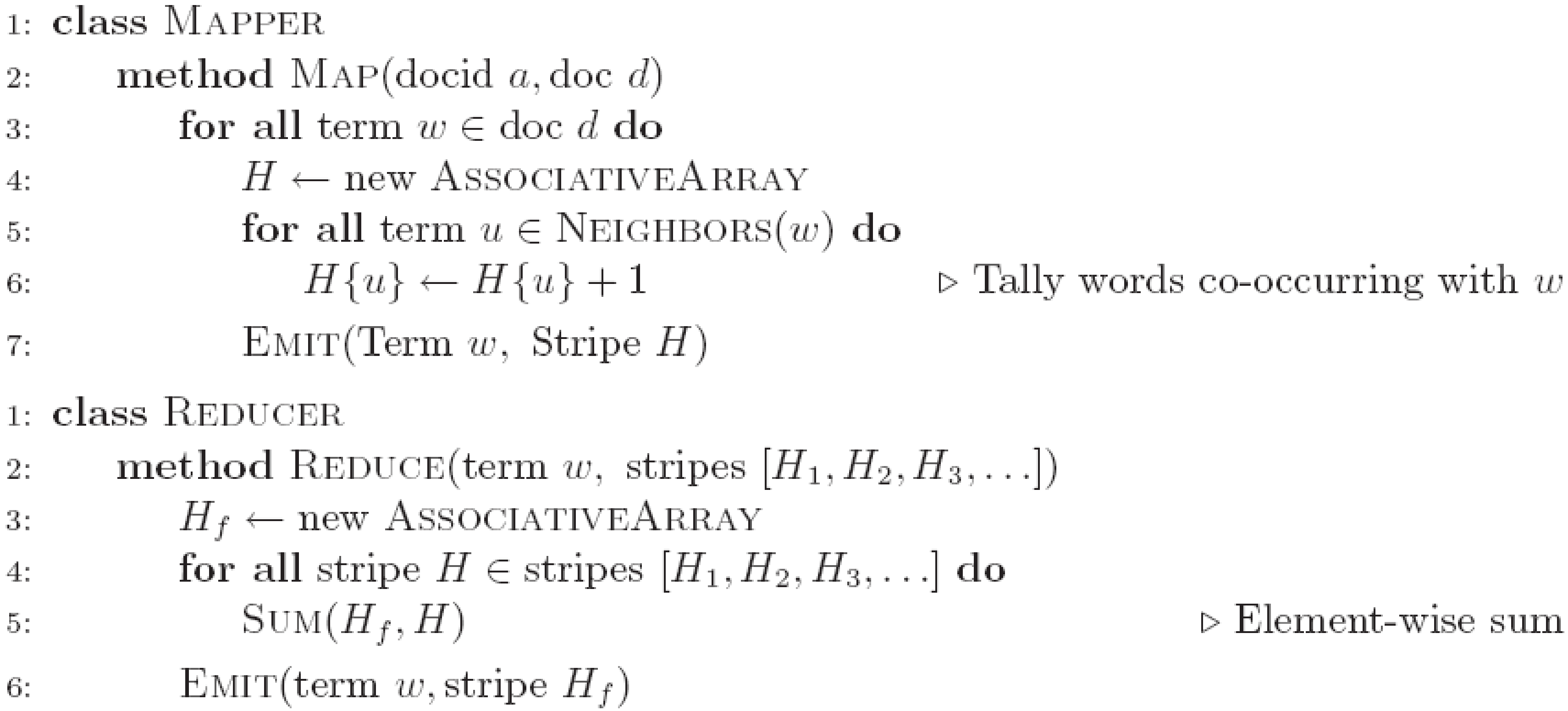
a → { b: 1, c: 2, d: 5, e: 3, f: 2 }

- Each mapper takes a sentence:
  - Generate all co-occurring term pairs
  - For each term, emit a → { b: count_b, c: count_c, d: count_d … }
- Reducers perform element-wise sum of associative arrays

$$\begin{array}{llllll} & a \to \{\ b{:}\ 1, & & d{:}\ 5, & e{:}\ 3\ \} & \\ + & a \to \{\ b{:}\ 1, & c{:}\ 2, & d{:}\ 2, & & f{:}\ 2\ \} \\ \hline & a \to \{\ b{:}\ 2, & c{:}\ 2, & d{:}\ 7, & e{:}\ 3, & f{:}\ 2\ \} \end{array}$$

Key: cleverly-constructed data structure brings together partial results

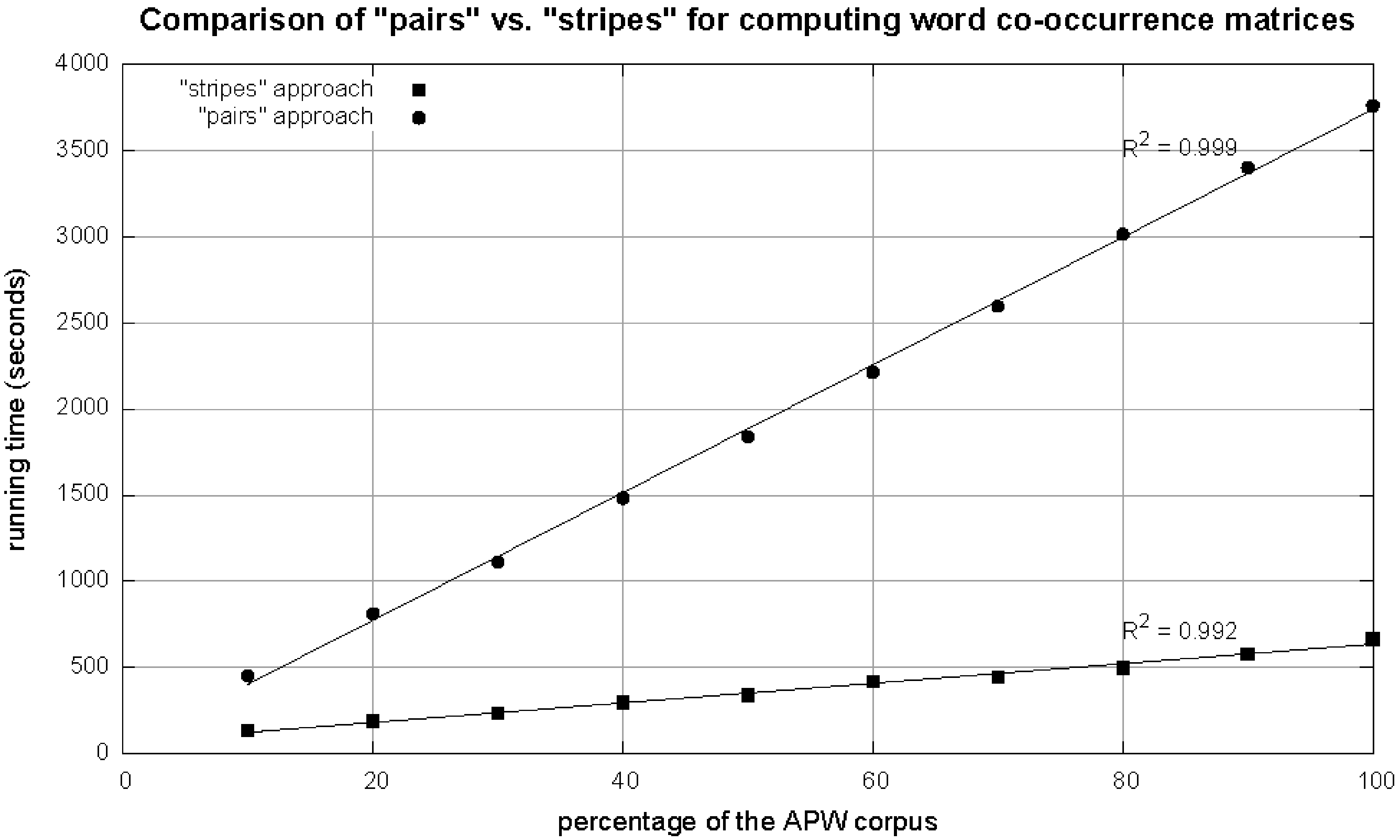
Stripes: Pseudo-Code



```
class Mapper
    method Map(docid $a$, doc $d$)
        for all term $w \in$ doc $d$ do
            $H \leftarrow$ new AssociativeArray
            for all term $u \in$ Neighbors($w$) do
                $H\{u\} \leftarrow H\{u\} + 1$                ▷ Tally words co-occurring with $w$
            Emit(Term $w$, Stripe $H$)

class Reducer
    method Reduce(term $w$, stripes $[H_1, H_2, H_3, \ldots]$)
        $H_f \leftarrow$ new AssociativeArray
        for all stripe $H \in$ stripes $[H_1, H_2, H_3, \ldots]$ do
            Sum($H_f, H$)                                    ▷ Element-wise sum
        Emit(term $w$, stripe $H_f$)
```

“Stripes” Analysis

- Advantages
  - Far less sorting and shuffling of key-value pairs
  - Keys are less unique than in pairs approach
  - Can make better use of combiners
- Disadvantages
  - More difficult to implement
  - Underlying object more heavyweight
  - Fundamental limitation in terms of size of event space



Cluster size: 38 cores
Data Source: Associated Press Worldstream (APW) of the English Gigaword Corpus (v3), which
contains 2.27 million documents (1.8 GB compressed, 5.7 GB uncompressed)

Map Reduce for Machine Learning

- Random Forest?
- SVM?