Hands-on lab: Security analytics

ENEE 657

Octavian Suciu
osuciu@umiacs.umd.edu
University of Maryland, College Park

http://ter.ps/enee657

Today's Lecture

- Where we've been
  - OS protection mechanisms
- Where we're going today
  - Intro to supervised learning
  - Intro to Apache Spark
  - Document Similarity
  - Hands-on: Spark
- Where we're going next
  - **Homework 2 out today, due next Wednesday!**
  - **First paper critiques due next Monday!**
  - Network security fundamentals

2

Homework & Paper Critique Submissions

- Use the submit command on GRACE
  - SSH into grace.umd.edu
    `submit <year> <semester> <college> <course> <section> <assignment> <filename>`
    - Example: `submit 2017 fall enee 657 0101 1 exploit_1.c`
  - Wrapper that performs some checks on the submission
    /afs/glue.umd.edu/class/fall2017/enee/657/0101/bin/submit
  - For more information on GRACE: http://www.grace.umd.edu/

- For critiques, submit BibTeX files in plain text
  - No Word DOC, no RTF, no HTML!
  - Do not remove BibTeX syntax (e.g. the @ sign before entries)
    - This confuses my parser and **I may think that you did not submit the homework** if I don't catch the error!
  - Submission deadline: at noon one week before class
    - Example: critiques for Mon 09/25 papers **due Mon 09/18**

3

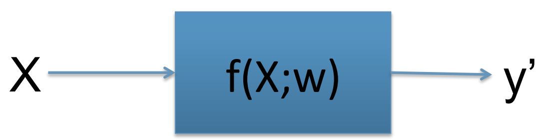
Predicting which papayas are tasty

- You arrive in a small Pacific island. Papayas are an important ingredient here.
- You don't know how papayas taste like, but you want to be able to pick tasty papayas from the market.
- You taste a lot of papayas and record a part of their features: softness and color.
- Based on these features, you want to predict which (new) papayas from the market are tasty.
- Supervised learning aims to solve this!

4

Introduction to Supervised Learning

- Components of a classification system:
  - Training set **(X, y)**
  - Prediction: **y' = f(X;w)**
  - Cost function: **c(y', y)**
  - Goal: find **w** that minimizes **c** w.r.t. to **(X,y)** on **f**



5

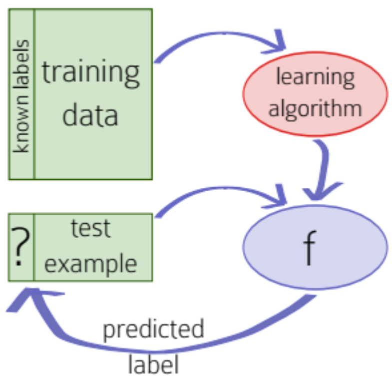
Supervised Learning in Context

- Training set (X, y)
  - If y is unknown → unsupervised learning
  - If y is categorical → classification
    - If y is binary → detection
  - If y is continuous → regression

6

Supervised Learning

- Training: a learning algorithm reads in training data and computes **f**
- Testing: **f** can then automatically label future text examples.



7

Example of Available Tools

- Libraries
  - Scikit-learn
  - Spark MLlib
  - R
  - Weka

- Specific
  - OpenCV
  - LibSVM
  - TensorFlow, Theano, Keras, …

8

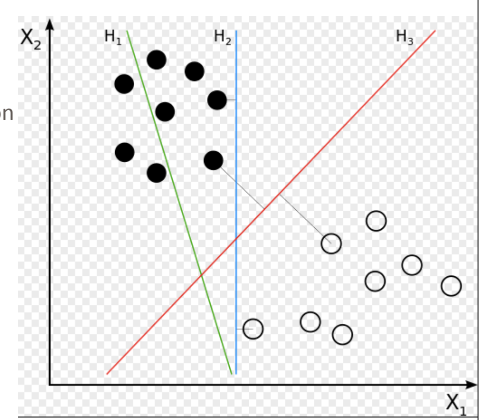
Popular Classification Techniques

- Logistic regression
- Naïve Bayes
- SVM
- Decision trees

9

SVM

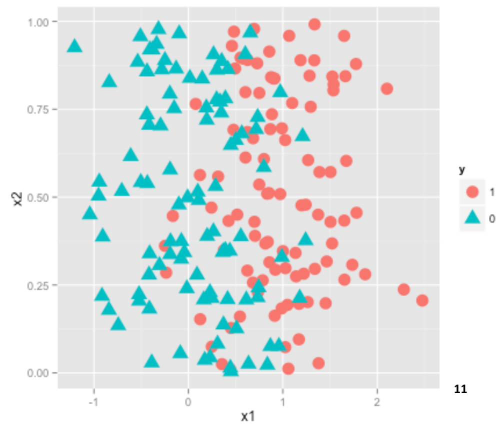
- Support Vector Machine
- Intuition
  - Training instances
    - Points in feature space
  - Classifier
    - Hyperplane that maximizes separation



10

Artificial Example

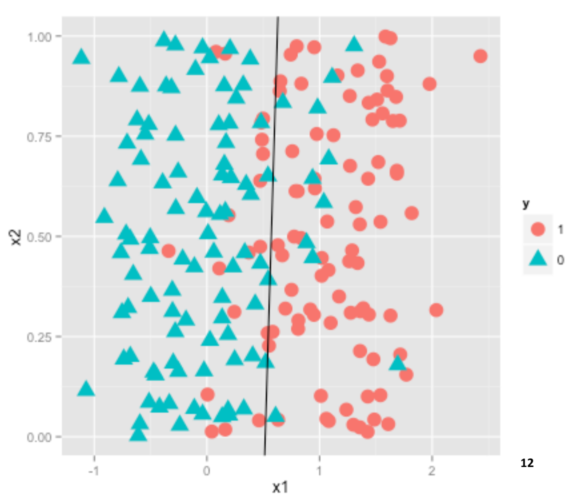
- Two groups (randomly generated)
  - X1
    - y=0: N(0, 0.5)
    - y=1: N(1, 0.5)
  - X2
    - U(0, 1)



11

Artificial Example (2)

- Separation hyperplane:
  - Ideal:
    - X1 = 0.5
  - Estimated:
    - Slope = 10
- Finding the ideal classifier is hard!



12

Implementation

- Steps
  1. Extract features
  2. Select model and classifier
  3. Select features
  4. Train the model
  5. Evaluate the performance
  6. Test on unlabeled examples

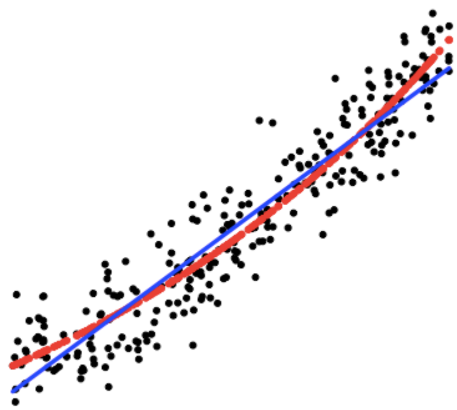
Feature Extraction

- Detecting malicious Android apps

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="org.sergez.splayer"
      android:versionCode="37"
      android:versionName="2.1">
  <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="19"/>
  <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
  <application
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:debuggable="false" >
    <activity
        android:name=".activity.SimplePlayerActivity"
        android:label="@string/app_name"
        android:theme="@style/Theme.Sherlock">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
    </activity>
    <service android:name=".service.SimplePlayerService" android:enabled="true"/>
  </application>
</manifest>
```

Model Selection

- Fitting a 2D set of points
  - Linear hypothesis
  - Higher order polynomial



15

Classifier Selection

- SVM

```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
```

- Naïve Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> clf = GaussianNB()
>>> clf.fit(X, Y)
```

- Decision Tree

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

16

Feature Selection

- Remove features with low variance

```
>>> from sklearn.feature_selection import VarianceThreshold
>>> X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
>>> sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
>>> sel.fit_transform(X)
```

- http://scikit-learn.org/stable/modules/feature_selection.html

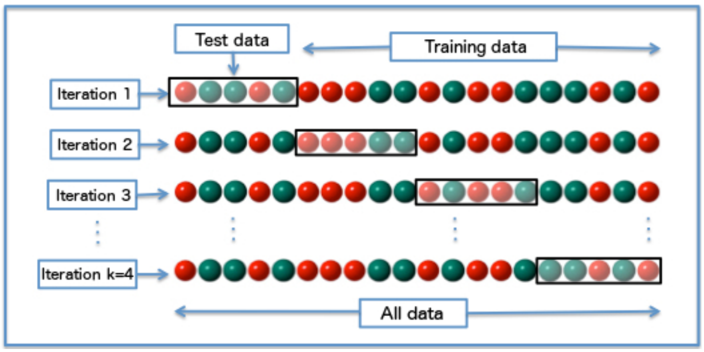
Performance Evaluation

- Popular metrics
  - Precision TP/(TP+FP)
    - Fraction of detected samples that are malicious
  - Recall (True positive rate) TP/(TP+FN)
    - Fraction of malicious samples that are detected
  - False positive rate FP/(FP+TN)

	TRUE (Truth)	FALSE (Truth)
TRUE (Detector)	True Positive (TP)	False Positive (FP)
FALSE (Detector)	False Negative (FN)	True Negative (TN)

Performance Evaluation (2)

- K-fold Cross validation
  - Partition data randomly to k subsamples
  - Training data: k-1 subsamples
  - Testing data: 1 subsample



Performance Evaluation (3)

```
>>> from sklearn import metrics
>>> scores = cross_validation.cross_val_score(clf, iris.data, iris.target,
...     cv=5, scoring='f1_weighted')
```

Scoring	Function	Comment
Classification		
'accuracy'	metrics.accuracy_score	
'average_precision'	metrics.average_precision_score	
'f1'	metrics.f1_score	for binary targets
'f1_micro'	metrics.f1_score	micro-averaged
'f1_macro'	metrics.f1_score	macro-averaged
'f1_weighted'	metrics.f1_score	weighted average
'f1_samples'	metrics.f1_score	by multilabel sample
'log_loss'	metrics.log_loss	requires predict_proba support
'precision' etc.	metrics.precision_score	suffixes apply as with 'f1'
'recall' etc.	metrics.recall_score	suffixes apply as with 'f1'
'roc_auc'	metrics.roc_auc_score	
Clustering		
'adjusted_rand_score'	metrics.adjusted_rand_score	
Regression		
'mean_absolute_error'	metrics.mean_absolute_error	
'mean_squared_error'	metrics.mean_squared_error	
'median_absolute_error'	metrics.median_absolute_error	
'r2'	metrics.r2_score	

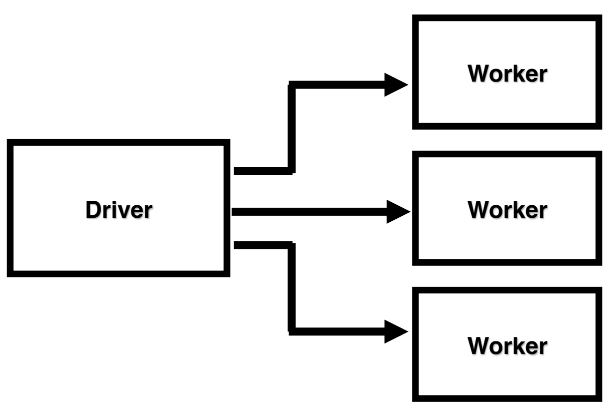
Apache Spark

- Framework for processing large volumes of data
- Based on the Map/Reduce paradigm
- Architecture based on Driver & Workers
  - Driver sends computation to the workers
  - Workers compute & report to the Driver for synchronization
- Primitive: Resilient Distributed Datasets - RDDs
- All workers execute the same task

21

Spark architecture

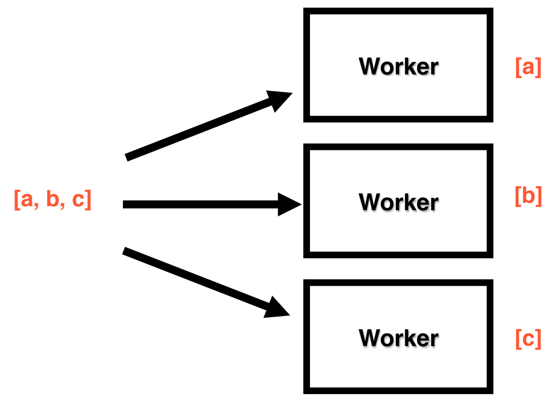
- Driver spawns & assigns tasks to workers



22

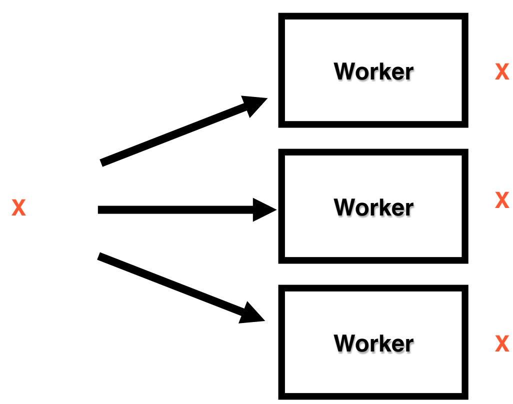
RDDs

- Distributed array, evenly split across workers
- Enable operation pipelining & fault tolerance

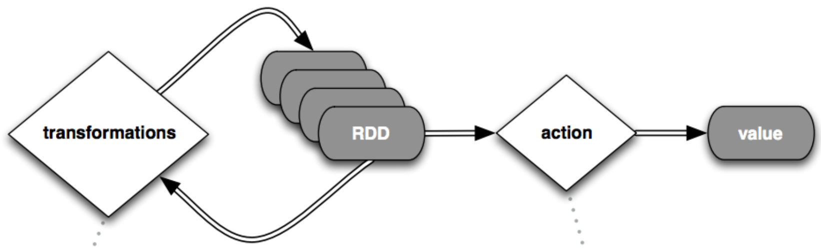


Broadcast Variables

- Immutable values cached by all workers



Operations on RDDs



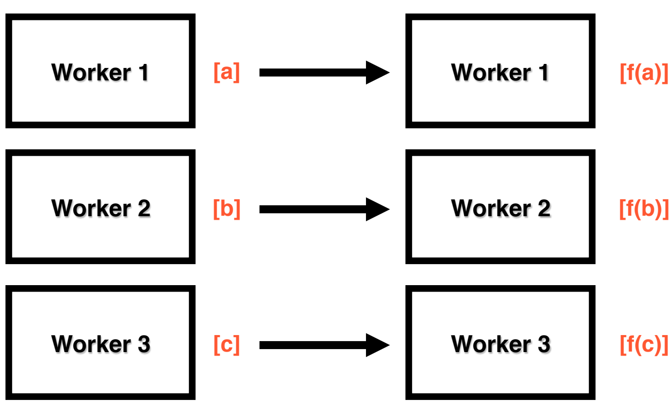

```
// transformed RDDs
val errors = lines.filter(_.startsWith("ERROR"))
val messages = errors.map(_.split("\t")).map(r => r(1))
messages.cache()
```

```
// action 1
messages.filter(_.contains("mysql")).count()
```

25

Example Operation: Map()

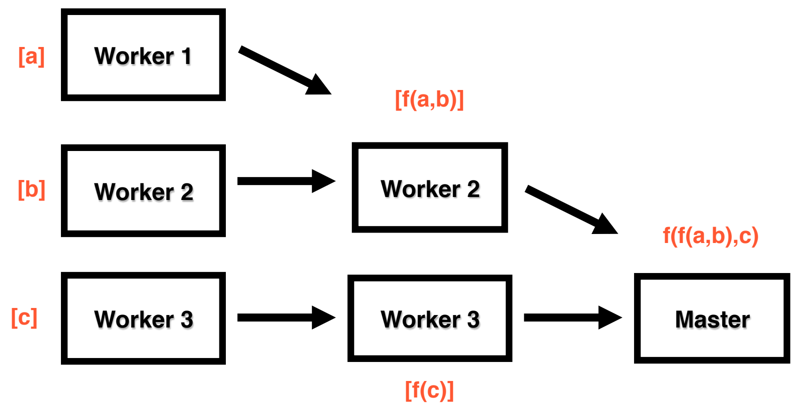
- Each element of the RDD is transformed using **f()**



26

Example Operation: Reduce()

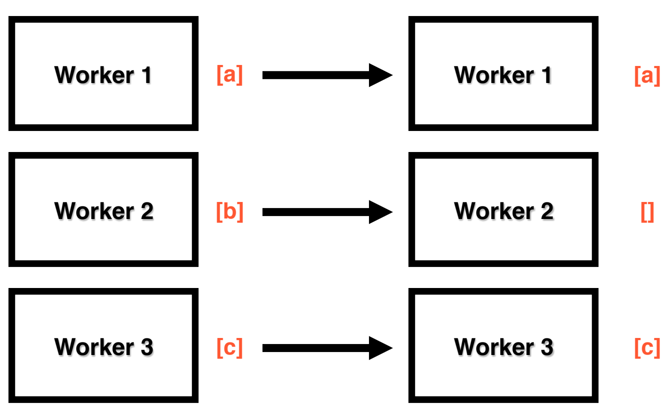
- Merges elements using a commutative & associative function **f()**



27

Example Operation: Filter()

- Retains elements matching a condition



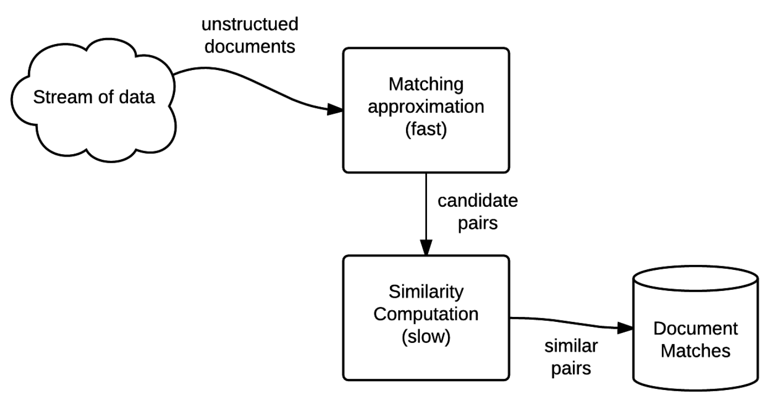
28

Document Similarity

- Textual similarity between millions of documents
- All-pairs similarity is not feasible
- Example applications:
  - Plagiarism detection
  - Exploit code reuse

29

Document Similarity Approach

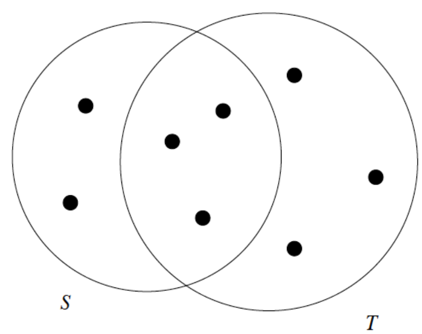


30

Jaccard Similarity

- Popular metric where documents are represented as sets

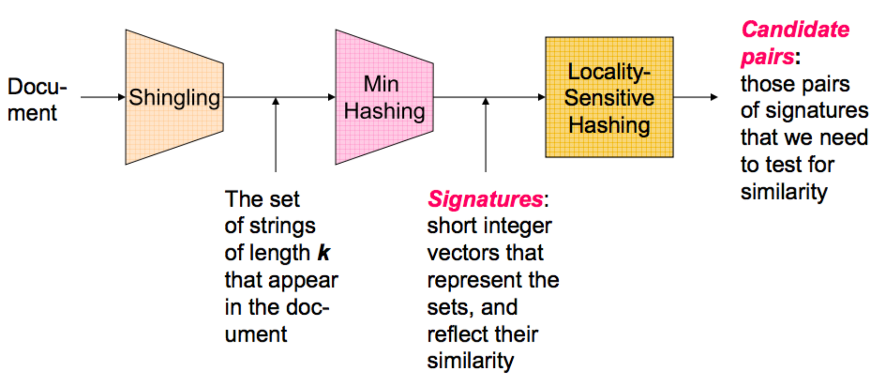
$$SIM(S,T) = \frac{|S \cap T|}{|S \cup T|}$$



SIM(S,T) = 3/8

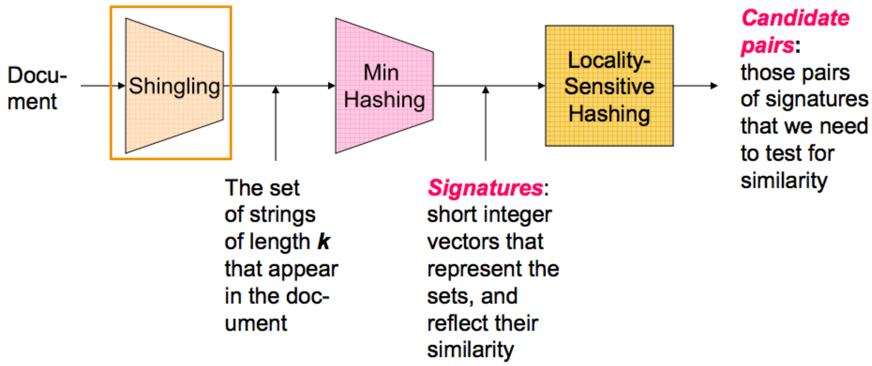
31

Computing the Jaccard Similarity Efficiently



32

Computing the Jaccard Similarity Efficiently



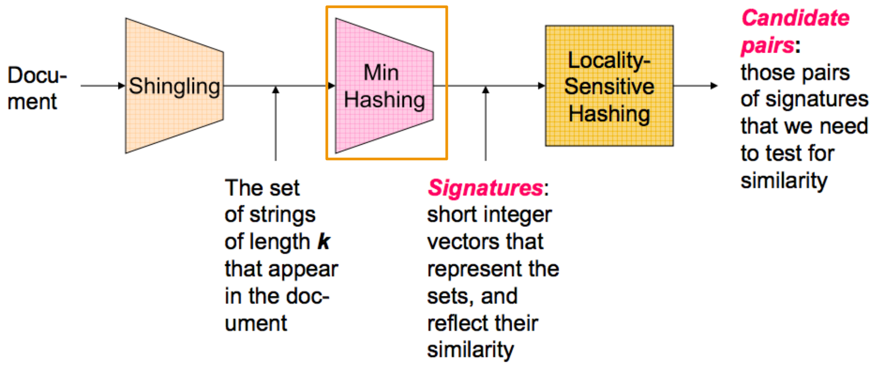
33

Document Shingling

- Split document in sequences of tokens
- Tokens are words/characters etc
- Sequence of k tokens = k-shingle (k-gram)
- Example:
  - **D = abcab**
  - **k = 2 chars**
  - **S(D) = { ab, bc, ca}**

34

Computing the Jaccard Similarity Efficiently



35

Minhashing

- Represent large sets of tokens through smaller signatures
- Preserves the original Jaccard similarity when compared

36

Computing the minhash (1)

- Characteristic matrix:

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

37

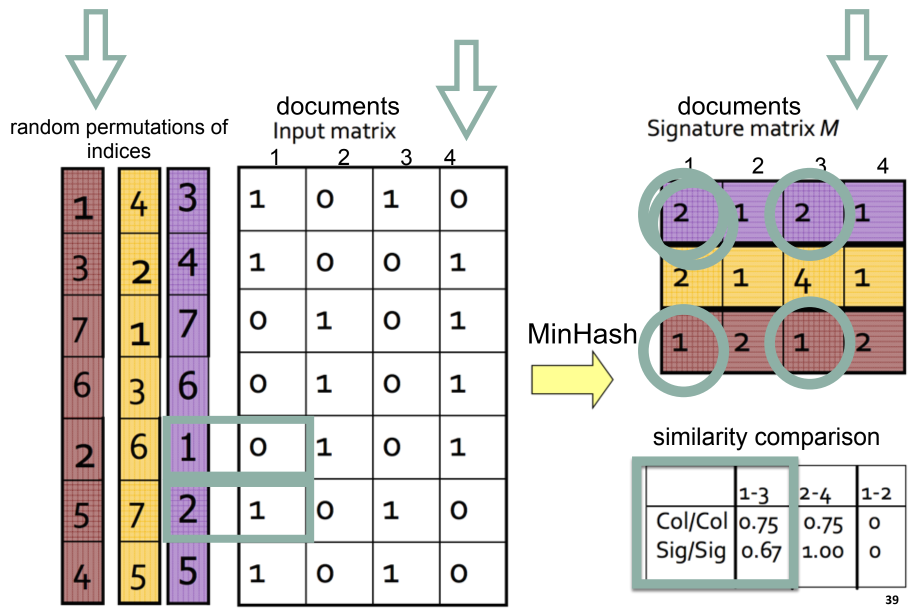
Computing the minhash (2)

- Random permutation of rows
- MinHash = first row in which a document has '1'

	Element	S_1	S_2	S_3	S_4
	b	0	0	1	0
	e	0	0	1	0
h(S1)=a	a	1	0	0	1
	d	1	0	1	1
	c	0	1	0	1

38

Minhash Example



	1-3	2-4	1-2
Col/Col	0.75	0.75	0
Sig/Sig	0.67	1.00	0

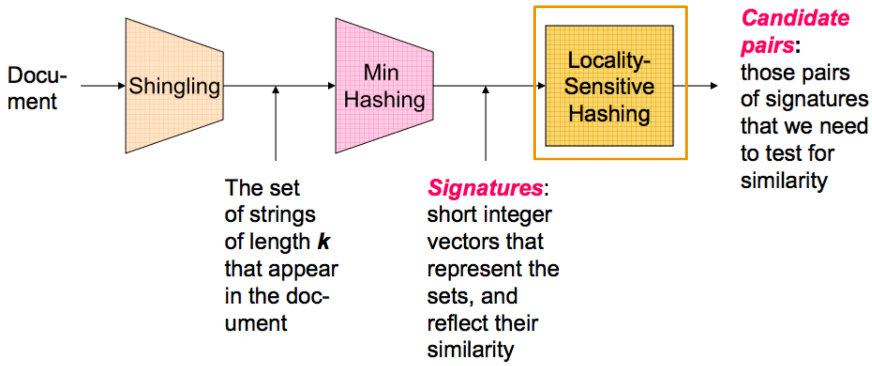
39

Minhash Property

- Probability that h(D1) = h(D2) ~ SIM(D1, D2)

40

Computing the Jaccard Similarity Efficiently



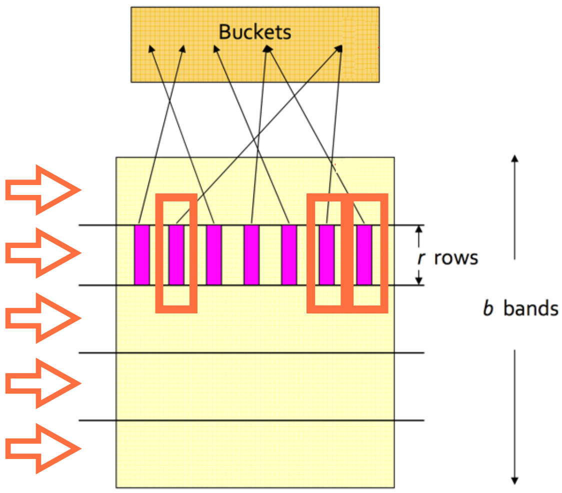
41

Locality Sensitive Hashing (LSH)

- Generate small list of candidate pairs from collection of signatures
- Idea:
  - Hash signatures to many buckets
  - Elements in the same bucket are candidate pairs
- Documents are split in bands (chunks) then hashed independently

42

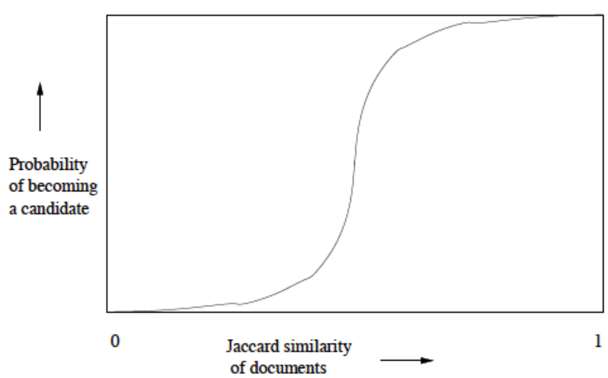
LSH Example



43

Property of LSH

- Probability that two similar signatures will agree on at least one of the bands is high
  - And that is exactly when they become candidates!



44

Sources

- J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org
- DataBricks: Spark Tutorial, http://lintool.github.io/SparkTutorial/
- A Course in Machine Learning by Hal Daumé III, http://ciml.info/
- Understanding Machine Learning: From Theory to Algorithms: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/
- Ziyun Zhu
- Radu Marginean

45