Stream Analyses Technical Challenges
Example IP Packet Stream Instantiation

Inputs

Dataflow Graph

By IBM Dense Information Gliding Team
Semantic MM Filtering

Inputs

Packet content analysis

Dataflow Graph

per PE rates

<table>
<thead>
<tr>
<th></th>
<th>200-500MB/s</th>
<th>~100MB/s</th>
<th>10 MB/s</th>
</tr>
</thead>
</table>

Advanced content analysis

Interested MM streams

Interest Routing

Interest Filtering

Keywords

Session video

Session audio

Interest Filtering

Keywords

Packet content analysis

ftp

http

udp

rtsp

rtp

ntp

Session audio

Session video

Keywords

Packet content analysis

Interest Filtering

Keywords

Packet content analysis

Interest Filtering

Keywords

Packet content analysis

Interest Filtering

Keywords

Packet content analysis

Interest Filtering

Keywords

Packet content analysis

Interest Filtering

Keywords

Packet content analysis

Interest Filtering

Keywords

Packet content analysis

Interest Filtering

Keywords
Resource-Accuracy Trade-Offs

Configurable Parameters of Processing Elements to maximize relevant information:

\[ Y''(X \mid q, R) > Y'(X \mid q, R), \]

with resource constraint.

Required resource-efficient algorithms for:
- Classification, routing and filtering of signal-oriented data: (audio, video and, possibly, sensor data)

- Input data X – Queries q – Resource R
  - \( Y(X \mid q) \): Relevant information
  - \( Y'(X \mid q, R) \cdot Y(X \mid q) \): Achievable subset given R
Example: Distributed Video Signal Understanding (Lin et al.)

(Distributed Smart Sensors) Block diagram of the smart sensors

- Encoding
- GOP Extraction
- Feature Extraction
- Event Extraction

(Control) Concept Detection Processing Elements

- PE1: 9.2.63.66:1220
- PE2: 9.2.63.67
- PE3: 9.2.63.66:1240
- PE4: 9.2.63.66:1235
- PE5: 9.2.63.66:1240
- PE6: 9.2.63.66
- PE7: 9.2.63.66
- PE100: 9.2.63.66

Meta-data

- Face
- Female
- Outdoors
- Male
- Indoors
- Airplane
- Chair
- Clock

User Interests

- Resource Constraints

Display and Information Aggregation Modules

TV broadcast, VCR, DVD discs, Video File Database, Webcam
Semantic Concept Filters

E.g.:
Complexity Reduction Introduction

- Objective: Real-time classification of instances using Support Vector Machines (SVMs)
- Computationally efficient and reasonably accurate solutions
- Techniques capable of adjusting tradeoff between accuracy and speed based on available computational resources
SVM formulation

- **Given:**
  - Training instances \( \{ x_i \} \) with labels \( y_i \)

- **Objective:**
  - Find maximum margin hyperplane separating positive and negative training instances
Decision

- Score of unseen instance: \( u_j : w \cdot \phi(u_j) \)
- In terms of Lagrangian multipliers:
  \[ \sum_i \alpha_i y_i k(x_i, u_j) \]
- Computational Cost: \( O(n_{sv} d) \)
  - \( n_{sv} \): Number of support vectors
  - \( d \): Dimensionality of each data instance
Problems

- Number of support vectors grows quasi-linearly with size of training set [Tipping 2000]

- Inner product with each support vector of dimensionality $d$ expensive
  - Example TREC2003
    - Human : 19745 support vectors
    - Face : 18090

- High data rates (10Gbits/sec) means large number of abandoned data
Example

- Processing Power 1 Ghz
- 10000 support vectors
- 1000 / 2 features per instance
- Order of at least $10^7$ operations required per stream per sec
- Translates to less than 100 instances evaluated per sec with only one classifier
Naïve Approach I – Feature Dimension Reduction

- Experimental Results for Weather_News Detector
- Model Selection based on the Model Validation Set
- E.g., for Feature Dimension Ratio 0.22, (the best selection of features are: 3 slices, 1 color, 2 texture selections), the accuracy is decreased by 17%.

### Table: Slice, Color, Texture, Feature Dimension Ratio, AP

<table>
<thead>
<tr>
<th>Slice</th>
<th>Color</th>
<th>Texture</th>
<th>Feature Dimension Ratio</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0.7861</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0.666666667</td>
<td>0.7861</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>0.666666667</td>
<td>0.7757</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3</td>
<td>0.666666667</td>
<td>0.5822</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0.444444444</td>
<td>0.7757</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>0.444444444</td>
<td>0.5822</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0.444444444</td>
<td>0.5235</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>0.333333333</td>
<td>0.4685</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0.333333333</td>
<td>0.6581</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0.333333333</td>
<td>0.1684</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.296296296</td>
<td>0.5235</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0.222222222</td>
<td>0.427</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0.222222222</td>
<td>0.6581</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
<td>0.222222222</td>
<td>0.1241</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0.222222222</td>
<td>0.3451</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0.222222222</td>
<td>0.1684</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0.222222222</td>
<td>0.1065</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0.148148148</td>
<td>0.0699</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0.148148148</td>
<td>0.3457</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0.148148148</td>
<td>0.1065</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0.111111111</td>
<td>0.3219</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0.111111111</td>
<td>0.0314</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0.111111111</td>
<td>0.07</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.074074074</td>
<td>0.0318</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>0.074074074</td>
<td>0.0175</td>
</tr>
</tbody>
</table>
Naïve Approach II – Reduction on the Number of Support

- Proposed Novel Reduction Methods:
  - Ranked Weighting
  - P/N Cost Reduction
  - Random Selection
  - Support Vector Clustering and Centralization

- Experimental Results on Weather_News Detectors show that complexity can be at 50% for the cost of 14% decrease on accuracy
Weighted Clustering Approach

- **Basic steps**
  - Cluster support vectors
  - Use cluster center as representative for all support vectors in cluster
  - Determine scalar weight associated with each cluster center
  - Use only cluster centers to score new instances
Cluster center weight (contd.)

- Choose $\gamma_i$ minimizing square of difference in scores over all $\pm_i$ and $d$
- Sub-cases:

\[
d \geq \Delta_i \\
d < \Delta_i
\]
Using the weights

- For every support vector in cluster
  - Distance $\Delta_i$ known
  - Two weights computed

- Cumulative effect of all support vectors in clusters additive
  - $\Delta_i$ because of various support vectors added up at center to simulate effect of all support vectors

- $\Delta_i$ sorted, weight arrays rearranged
Experiments

• Datasets
  • TREC video datasets (2003 and 2005)
    • 576 features per instance
    • > 20000 test instances overall
  • MNist handwritten digit dataset (RBF kernel)
    • 576 features
    • 60000 training instances, 10000 test instances

• Performance metrics
  • Speedup achieved over evaluation with all support vectors
  • Average precision achieved
Results (Mnist 0-4)

![Graph showing Speedup Ratio vs. Average Precision for different values of n (0-4)].

- **Average Precision**: Varies from 0.95 to 1.0.
- **Speedup Ratio**: Ranges from 0 to 300.

Each line represents a different value of n (0, 1, 2, 3, 4), with distinct markers for clarity.
Results (Mnist 5-9)

![Graph showing average precision vs. speedup ratio for different classes (5-9)]
Results (TREC 2003)

Average Precision

<table>
<thead>
<tr>
<th>Concept</th>
<th>AP_fast</th>
<th>AP_original</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>0.75</td>
<td>0.675</td>
</tr>
<tr>
<td>Outdoors</td>
<td>0.675</td>
<td>0.5</td>
</tr>
<tr>
<td>Sport-Event</td>
<td>0.5</td>
<td>0.35</td>
</tr>
<tr>
<td>Crowd</td>
<td>0.25</td>
<td>0.225</td>
</tr>
<tr>
<td>People-Event</td>
<td>0.2</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Speedup

<table>
<thead>
<tr>
<th>Concept</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>10000</td>
</tr>
<tr>
<td>Studio-Setting</td>
<td>1000</td>
</tr>
<tr>
<td>Crowd</td>
<td>100</td>
</tr>
</tbody>
</table>
Summary of Complexity Reduction

- Techniques presented demonstrate reasonable performance in terms of both speedup and average precision over multiple concepts in datasets

- Speedups
  - MNist: All concepts at least 50 times faster with AP within 0.04 of original
  - TREC 2003: Eight out of nine concepts speedup greater than 80 times with AP within 0.05 of original
  - TREC 2005: APs in some cases along with speedup respectable

- APs of most concepts close to original APs
Acceleration of Neural Network for Streams
Methods for Running CNNs on Mobile Devices

Sending CNN jobs to cloud

Acceleration CNN on Local Device

Apple A9X SoC, 12-core GPU

- How to trade off between algorithmic complexity and performance?
- How to utilize hardware effectively
Summary of Acceleration of Neural Network on Mobile Devices

- Porting Deep Convolution Neural Network on iOS Device with near real-time computation
  - Reduce algorithmic complexity with ignorable performance degradation.
  - Utilize computational hardware to achieve better performance.

<table>
<thead>
<tr>
<th></th>
<th>iPhone 7</th>
<th>iPhone 7+</th>
<th>iPhone 6s</th>
<th>iPad Pro 12.9&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex Net</td>
<td>70 ms</td>
<td>70 ms</td>
<td>130 ms</td>
<td>69 ms</td>
</tr>
<tr>
<td>p-Alex Net</td>
<td>N/A</td>
<td>35 ms</td>
<td>N/A</td>
<td>28 ms</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>130 ms</td>
<td>128 ms</td>
<td>195 ms</td>
<td>110 ms</td>
</tr>
<tr>
<td>p-GoogLeNet</td>
<td>N/A</td>
<td>80 ms</td>
<td>N/A</td>
<td>70 ms</td>
</tr>
<tr>
<td>VGG16 Net</td>
<td>880 ms</td>
<td>883 ms</td>
<td>1450 ms</td>
<td>725 ms</td>
</tr>
</tbody>
</table>
Challenges for Running CNN on Mobile Devices

<table>
<thead>
<tr>
<th>Model Size</th>
<th>Weights</th>
<th>Mult.s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AlexNet</strong></td>
<td>243MB</td>
<td>61M</td>
</tr>
<tr>
<td><strong>VGG-S</strong></td>
<td>393MB</td>
<td>103M</td>
</tr>
<tr>
<td><strong>VGG-16</strong></td>
<td>552MB</td>
<td>138M</td>
</tr>
<tr>
<td><strong>GoogLeNet</strong></td>
<td>51MB</td>
<td>6.9M</td>
</tr>
</tbody>
</table>

Statistics of some popular CNNS

Reference:
Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications
### Computational Resource on iPhone and iPad

<table>
<thead>
<tr>
<th></th>
<th>iPhone 6S (Plus)</th>
<th>iPad Air 2</th>
<th>iPad Pro (12.9/9.7)</th>
<th>iPhone 7 (Plus)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoC</td>
<td>A9</td>
<td>A8X</td>
<td>A9X</td>
<td>A10 Fusion</td>
</tr>
<tr>
<td>CPU</td>
<td>2x Twister @ 1.85 GHz</td>
<td>3x Typhone @ 1.5 GHz</td>
<td>2x Twister @ 2.26 GHz</td>
<td>4-core</td>
</tr>
<tr>
<td>GPU</td>
<td>PVR GT7600 (6 cluster)</td>
<td>PVR GXA6850 (8 cluster)</td>
<td>PVR 12 Cluster Series 7</td>
<td>6 cluster GPU?</td>
</tr>
<tr>
<td>RAM (shared memory)</td>
<td>2GB LDDR4</td>
<td>2GB LDDR3</td>
<td>4GB LDDR4</td>
<td>3GB on Plus?</td>
</tr>
<tr>
<td>Memory bus width</td>
<td>64-bit</td>
<td>128-bit</td>
<td>128-bit</td>
<td>?</td>
</tr>
<tr>
<td>Max # of threads per group</td>
<td>512</td>
<td>512</td>
<td>512</td>
<td>?</td>
</tr>
</tbody>
</table>
Neuron Importance Score Propagation (NISP, Yu et al 2018)
Methods for Running CNNs on Mobile Devices

- Compression (pruning) of CNN
- Speeding up CNN
- Sending CNN jobs to cloud
Thinking Differently

• All existing methods can be viewed as approximations of an overly-redundant CNN, but do we really need such a CNN as the starting
Slim CNN

- Slim CNN leads to:
  - less storage space
  - less memory usage
  - less computation
  - less power consumption
Feature Selection on CNN

- CNNs can be viewed as a set of "overly-redundant" feature extractors.
A method for Pruning Redundant Neurons and Kernels of

Apply thermal

A pre-trained CNN

Extract CNN Responses

Measure the Importance of Feature Extractors

Prune Model

Fine-tuning
A method for Pruning Redundant Neurons and Kernels of Deep Convolutional Neural Networks (NISP)

- Intractable ➔ tractable
- Inconsistent ➔ consistent

A pre-trained CNN ➔ Extract Responses of a High-level Layer ➔ Measure the Importance of Feature Extractors ➔ Back-propagate the Importance & Prune Model ➔ Fine-tuning

Forward Propagation

Important Score Back Propagation and Pruning
Fine-tuning the Pruned Model

- Our method outperforms the baselines in three aspects
  - Very small accuracy loss at the beginning $\Rightarrow$ retains the most important neurons
  - Converges much faster than baselines
  - For LeNet on MIST, our method only decreases 0.02% top-1 accuracy with a running ratio of 50% as compared to the pre-pruned network.
Fine-tuning the Pruned Model

- The pruned model consists of important feature extractors, but will suffer loss of accuracy due to loss of redundant features
  - Good starting point on the learning curve due to feature selection
  - Fine-tuning the pruned model with a lower learning rate to recover the performance
Spark Streaming

- Basic Concepts
  - Linking
  - Initializing StreamingContext
  - Discretized Streams (DStreams)
  - Input DStreams and Receivers
  - Transformations on DStreams
  - Output Operations on DStreams
  - DataFrame and SQL Operations
  - MLlib Operations
  - Caching / Persistence
  - Checkpointing
  - Accumulators, Broadcast Variables, and Checkpoints
  - Deploying Applications
  - Monitoring Applications
- Performance Tuning
  - Reducing the Batch Processing Times
  - Setting the Right Batch Interval
  - Memory Tuning
- Fault-tolerance Semantics
Spark Streaming
Spark Streaming
Spark Streaming Example

First, we import `StreamingContext`, which is the main entry point for all streaming functionality. We create a local `StreamingContext` with two execution threads, and batch interval of 1 second.

```python
from pyspark import SparkContext
from pyspark.streaming import StreamingContext

# Create a local StreamingContext with two working thread and batch interval of 1 second
sc = SparkContext("local[2]", "NetworkWordCount")
ssc = StreamingContext(sc, 1)
```

Using this context, we can create a DStream that represents streaming data from a TCP source, specified as hostname (e.g. `localhost`) and port (e.g. 9999).

```python
# Create a DStream that will connect to hostname:port, like localhost:9999
lines = ssc.socketTextStream("localhost", 9999)
```

This `lines` DStream represents the stream of data that will be received from the data server. Each record in this DStream is a line of text. Next, we want to split the lines by space into words.

```python
# Split each line into words
words = lines.flatMap(lambda line: line.split(" "))
```

```python
# Count each word in each batch
pairs = words.map(lambda word: (word, 1))
wordCounts = pairs.reduceByKey(lambda x, y: x + y)
```

```python
# Print the first ten elements of each RDD generated in this DStream to the console
wordCounts.pprint()
```
Spark Streaming Example

$ ./bin/spark-submit examples/src/main/python/streaming/network_wordcount.py localhost 9999

Then, any lines typed in the terminal running the netcat server will be counted and printed on screen every second. It will look something like the following.

# TERMINAL 1:
# Running netcat
$ nc -lk 9999
hello world

# TERMINAL 2: RUNNING network_wordcount.py
$ ./bin/spark-submit examples/src/main/python/streaming/network_wordcount.py localhost 9999
...
-----------------------------------------------
-----------------------------------------------
(hello,1)
(world,1)
...
Discretized Streams

DStream → RDD @ time 1: data from time 0 to 1 → RDD @ time 2: data from time 1 to 2 → RDD @ time 3: data from time 2 to 3 → RDD @ time 4: data from time 3 to 4
Discretized Streams

[Diagram showing the flow of data through a series of operations, including `flatMap` operations.]
Discretized Streams

https://www.edureka.co/blog/spark-streaming/
DStream Transforms

https://www.edureka.co/blog/spark-streaming/
Output DStreams

https://www.edureka.co/blog/spark-streaming/
DStreams Caching

https://www.edureka.co/blog/spark-streaming/
import org.apache.spark.streaming.{Seconds, StreamingContext}
import org.apache.spark.SparkContext._

object twitterSentiment {
  def main(args: Array[String]) {
    if (args.length < 4) {
      System.err.println("Usage: TwitterPopularTags <consumer key> <consumer secret> " + "+"<access token> <access token secret>
      System.exit(1)
    }

    StreamingExamples.setStreamingLogLevels()
    //Passing our Twitter keys and tokens as arguments for authorization
    val Array(consumerKey, consumerSecret, accessToken, accessTokenSecret) = args.take(4)
    val filters = args.takeRight(args.length - 4)

    // Set the system properties so that Twitter4j library used by twitter stream
    // Use them to generate OAuth credentials
    System.setProperty("twitter4j.oauth.consumerKey", consumerKey)
    System.setProperty("twitter4j.oauth.consumerSecret", consumerSecret)

    val sparkConf = new SparkConf().setAppName("twitterSentiment").setMaster("local[2]")
    val ssc = new StreamingContext
    val stream = TwitterUtils.createStream(ssc, None, filters)

    //...
// Input DStream transformation using flatMap
val tags = stream.flatMap { status => Get Text From The Hashtags }

// RDD transformation using sortBy and then map function
tags.countByValue().foreachRDD { rdd =>
  val now = Get current time of each Tweet
  rdd .sortBy(_._2)
  .map(x => (x, now))
  // Saving our output at ~/twitter/ directory
  .saveAsTextFile(s"~/twitter/$now")
}

// DStream transformation using filter and map functions
val tweets = stream.filter { t =>
  val tags = t. Split On Spaces .filter(_.startsWith("#")). Convert To Lower Case
tags.exists { x => true }
}

val data = tweets.map { status =>
  val sentiment = SentimentAnalysisUtils.detectSentiment(status.getText)
  val tagss = status.getHashtagEntities.map(_.getText.toLowerCase) (status.getText, sentiment.toString, tagss.toString())
}

data.print()
// Saving our output at ~/ with filenames starting like twitter
data.saveAsTextFiles("~/twitter", "20000")

ssc.start()
ssc.awaitTermination()
DStreams Example — Twitter Sentiment Analysis

Results:

The following are the results that are displayed in the Eclipse IDE while running the Twitter Sentiment Streaming program.

All the tweets are categorized into Positive, Neutral and Negative according to the sentiment of the contents of the tweets.

https://www.edureka.co/blog/spark-streaming/
Spark Clustering

- **K-means**
  - Input Columns
  - Output Columns
- Latent Dirichlet allocation (LDA)
- Bisecting k-means
- Gaussian Mixture Model (GMM)
  - Input Columns
  - Output Columns
Clustering — Gaussian Mixture Models

(Left) Fit with one Gaussian distribution (Right) Fit with Gaussian mixture model with two components
One-dimensional Model

\[ p(x) = \sum_{i=1}^{K} \phi_i \mathcal{N}(x \mid \mu_i, \sigma_i) \]

\[ \mathcal{N}(x \mid \mu_i, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp \left( -\frac{(x - \mu_i)^2}{2\sigma_i^2} \right) \]

\[ \sum_{i=1}^{K} \phi_i = 1 \]
Gaussian Mixture Model

Multi-dimensional Model

\[ p(\bar{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\bar{x} \mid \bar{\mu}_i, \Sigma_i) \]

\[ \mathcal{N}(\bar{x} \mid \bar{\mu}_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} \exp \left( -\frac{1}{2} (\bar{x} - \bar{\mu}_i)^T \Sigma_i^{-1} (\bar{x} - \bar{\mu}_i) \right) \]

\[ \sum_{i=1}^{K} \phi_i = 1 \]
Expectation maximization (EM) is a numerical technique for maximum likelihood estimation, and is usually used when closed form expressions for updating the model parameters can be calculated (which will be shown below). Expectation maximization is an iterative algorithm and has the convenient property that the maximum likelihood of the data strictly increases with each subsequent iteration, meaning it is guaranteed to approach a local maximum or saddle point.

https://brilliant.org/wiki/gaussian-mixture-model/
### Gaussian Mixture Model spark code

**Input Columns**

<table>
<thead>
<tr>
<th>Param name</th>
<th>Type(s)</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>featuresCol</td>
<td>Vector</td>
<td>&quot;features&quot;</td>
<td>Feature vector</td>
</tr>
</tbody>
</table>

**Output Columns**

<table>
<thead>
<tr>
<th>Param name</th>
<th>Type(s)</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>predictionCol</td>
<td>Int</td>
<td>&quot;prediction&quot;</td>
<td>Predicted cluster center</td>
</tr>
<tr>
<td>probabilityCol</td>
<td>Vector</td>
<td>&quot;probability&quot;</td>
<td>Probability of each cluster</td>
</tr>
</tbody>
</table>

```python
from pyspark.ml.clustering import GaussianMixture

# loads data
dataset = spark.read.format("libsvm").load("data/mllib/sample_kmeans_data.txt")

gmm = GaussianMixture().setK(2).setSeed(538009335)
model = gmm.fit(dataset)

print("Gaussians shown as a DataFrame: ")
model.gaussiansDF.show(truncate=False)
```
**Content Analysis - Latent Dirichlet Allocation (LDA) [Blei et al. 2003]**

Goal – categorize the documents into topics

- Each document is a probability distribution over topics
- Each topic is a probability distribution over words

The probability of the word $w_i$ under the $j$th topic $\theta_j^{(d)}$ is

$$P(w_i) = \sum_{j=1}^{T} P(w_i | z_i = j)P(z_i = j)$$

The probability of choosing a word from the $j$th topic in the current document $\phi_{w}^{(j)}$ is

DOCUMENT 1: money$^1$ bank$^1$ bank$^1$ loan$^1$ river$^2$ stream$^2$ bank$^1$
money$^1$ river$^2$ bank$^1$ money$^1$ bank$^1$ loan$^1$ money$^1$ stream$^2$
bank$^1$ money$^1$ loan$^1$ river$^2$ stream$^2$ bank$^1$ money$^1$

DOCUMENT 2: loan$^1$ river$^2$ stream$^2$ loan$^1$ bank$^2$ river$^2$ bank$^2$
bank$^1$ stream$^2$ river$^2$ loan$^1$ bank$^2$ stream$^2$ bank$^2$ money$^1$
loan$^1$ river$^2$ stream$^2$ bank$^2$ stream$^2$ bank$^2$ money$^1$ river$^2$
**INPUT:**
- document-word counts
  - \( D \) documents, \( W \) words

**OUTPUT:**
- likely topics for a document

\[
P(z \mid w) \propto P(w \mid z)P(z)
\]

- Parameters can be estimated by Gibbs Sampling
- Outperform Latent Semantic Analysis (LSA) and Probabilistic LSA in various experiments [Blei et al. 2003]

Bayesian approach: use priors
- Mixture weights \( \sim \text{Dirichlet}(\alpha) \)
- Mixture components \( \sim \text{Dirichlet}(\beta) \)
Traditional Content Clustering

**Clustering:**
Partition the feature space into segments based on training documents. Each segment represents a topic / category. (Topic Detection)

**Hard clustering:** e.g., K-mean clustering

\[ d = \{ f_{w_1}, f_{w_2}, ..., f_{w_N} \} \rightarrow z \]

- \( f_{w_j} \): the frequency of the word \( w_j \) in a document
Traditional Content Clustering

Clustering:
Partition the feature space into segments based on training documents. Each segment represents a topic / category. (Topic Detection)

Hard clustering: e.g., K-mean clustering

\[ d = \{f_{w_1}, f_{w_2}, \ldots, f_{w_N}\} \rightarrow z \]
Traditional Content Clustering

**Clustering:**
Partition the feature space into segments based on training documents. Each segment represents a topic / category. (_topic detection

**Hard clustering:** e.g., K-mean clustering

\[ d = \{ f_{w_1}, f_{w_2}, \ldots, f_{w_N} \} \rightarrow z \]
Clustering:
Partition the feature space into segments based on training documents. Each segment represents a topic / category. (_topic detection)

**Hard clustering**: e.g., K-mean clustering

\[
d = \{f_{w_1}, f_{w_2}, ..., f_{w_N}\} \rightarrow z
\]

*Traditional Content Clustering*

\[f_{w_j} \text{ : the frequency of the word } w_j \text{ in a document} \]

*Another representation of clustering* (w/o showing the deterministic part)

\[
\begin{align*}
\text{Words} & \quad w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \\
\text{Topics} & \quad z_1 \quad z_2 \quad z_3 \quad z_4 \quad z_5 
\end{align*}
\]
Clustering:
Partition the feature space into segments based on training documents. Each segment represents a topic / category. (_topic detection)

Hard clustering: e.g., K-mean clustering

\[ d = \{ f_{w_1}, f_{w_2}, \ldots, f_{w_N} \} \rightarrow z \]

Soft clustering: e.g., Fuzzy C-mean clustering

\[ P(Z \mid W = f_w) \]
Content Clustering based on Bayesian Network

**Bayesian Network:**
- Causality Network – models the causal relationship of attributes / nodes
- Allows hidden / latent nodes

**Hard clustering:**

\[
 h(D = d) = \arg \max_z P(W = f_w \mid Z) \leq \text{MLE}
\]

\[
 P(W \mid Z) = \frac{P(Z \mid W)P(W)}{P(Z)} \leq \text{Bayes Theorem}
\]
Content Clustering based on Bayesian Network – Hard Clustering

Topics

\(z_1\)  \(z_2\)  \(z_3\)  \(z_4\)  \(z_5\)

Words

\(w_1\)  \(w_2\)  \(w_3\)  \(w_4\)  \(w_5\)  \(w_6\)

\[P(W \mid Z) = \frac{P(Z \mid W)P(W)}{P(Z)} = \frac{P(Z \mid W)P(W)}{\int P(Z \mid W)dW}\]

**Major Solution 1 -- Dirichlet Process:**

- Models \(P(W \mid Z)\) as mixtures of Dirichlet probabilities
- Before training, the prior of \(P(W \mid Z)\) can be a easy Dirichlet (uniform distribution). After training, \(P(W \mid Z)\) will still be Dirichlet. (**The reason of using Dirichlet**)

**Major Solution 2 -- Gibbs Sampling:**

- A Markov chain Monte Carlo (MCMC) method for integration of large samples \(\Rightarrow\) calculate \(P(Z)\)

N: the number of words
(The number of topics (M) are pre-determined)
Content Clustering based on Bayesian Network – Soft Clustering

Documents

Topics

Words

$z_1$ $z_2$ $z_3$ $z_4$ $z_5$

$w_1$ $w_2$ $w_3$ $w_4$ $w_5$ $w_6$

$\alpha$ $\theta$

$\beta$ $\phi$

$\alpha$ $\theta$ $A$

$\beta$ $\phi$ $A$

$LDA$ (Blei 2003)

$N$: the number of words
$A$: the number of docs

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$

$d$

$z$

$w$

$A$

$N$

$\theta_A$

$\phi_M$
Comparison of Dirichlet Distribution with Gaussian Mixture Models (1)

Dirichlet Distribution:

\[ \rho(f_1, f_2, \ldots, f_{r-1}; a_1, a_2, \ldots, a_r) = \frac{\Gamma(N)}{\prod_{k=1}^{r} \Gamma(a_k)} f_1^{a_1-1} f_2^{a_2-1} \cdots f_r^{a_r-1} \]

\[ 0 \leq f_k \leq 1 \quad \sum_{k=1}^{r} f_k = 1 \]

- Multivariate Gaussian:

\[ \rho(f_1, f_2, \ldots, f_{r-1}; \mu_1, \sigma_1, \mu_2, \sigma_2, \ldots, \mu_{r-1}, \sigma_{r-1}) = \frac{1}{(2\pi)^{r-1} \prod_{k=1}^{r} \sigma_k} e^{-\frac{(f_1-\mu_1)^2}{\sigma_1^2}} e^{-\frac{(f_2-\mu_2)^2}{\sigma_2^2}} \cdots e^{-\frac{(f_{r-1}-\mu_{r-1})^2}{\sigma_{r-1}^2}} \]
Beyond Gaussian: Examples of Dirichlet Distribution

- Beta(0.5, 1, 1)
- Beta(0.5, 3, 3)
- Beta(0.5, 11, 5)
- Beta(0.5, 0.2, 0.2)
Use Dirichlet Distribution to model prior and posterior beliefs

Prior beliefs:
E.g.: *fair* coin?

Flipping a coin, what’s the probability of getting ‘head’.

\[
\rho(f) = \text{beta}(f; a, b)
\]

\(\text{beta}(1,1): \) No prior knowledge
\(\text{beta}(3,3): \) prior knowledge -- this coin may be fair
Use Dirichlet Distribution to model prior and posterior beliefs

Prior beliefs:
Posterior beliefs:

\[ \rho(f) = \text{beta}(f; a, b) \]

\[ \rho(f | d) = \text{beta}(f; a + s, b + t) \]

\[ \beta(3,3): \text{prior knowledge} \] -- this coin may be fair

\[ \beta(11,5): \text{posterior belief} \] -- the coin may not be fair after tossing 8 heads and 2 tails

\[ d: 8, 2 \]
Use Dirichlet Distribution to model prior and posterior beliefs

Prior beliefs:
Posterior beliefs:

\[ \rho(f) = \text{beta}(f; a, b) \]

\[ \rho(f | d) = \text{beta}(f; a + s, b + t) \]

\text{beta}(1/360, 19/360): prior knowledge that an 5% chance-event may be true

\text{beta}(1/360+3, 19/360): posterior belief that an 5% chance-event may be true
Importance of Dirichlet Distribution

In 1982, Sandy Zabell proved that, if we make certain assumptions about an individual’s beliefs, then that individual must use the Dirichlet density function to quantify any prior beliefs about a relative frequency.
Some Insight on BN-based Content Clustering

**Bayesian Network:**
- Models the *practical* causal relationships..

**Content Clustering:**
- Because documents and words are dependent,
  ➔ only close documents in the feature space can be clustered together as one topic.

⇒ Incorporating human factors can possibly *link* multiple clusters together.

$f_{w_1}, f_{w_2}, f_{w_3}$: the frequency of the word $w_j$ in a document
Gibbs Sampling

• Suppose that it is hard to sample $p(x)$ but that it is possible to “walk around” in $X$ using local state transitions
• Insight: we can use a “random walk” to help us draw random samples from $p(x)$
• At each transition change the state of just on $X_i$
• We can describe the transition probability as a stochastic procedure:
  Input: a state $x_1, \ldots, x_n$
  Choose $i$ at random (using uniform probability)
  Sample $x'_i$ from
    $p(X_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, e)$
  let $x'_j = x_j$ for all $j \neq i$
  return $x'_1, \ldots, x'_n$
Spark ML LDA code example

```scala
import org.apache.spark.ml.clustering.LDA

// Loads data.
val dataset = spark.read.format("libsvm")
  .load("data/mllib/sample_lda_libsvm_data.txt")

// Trains a LDA model.
val lda = new LDA().setK(10).setMaxIter(10)
val model = lda.fit(dataset)

val ll = model.logLikelihood(dataset)
val lp = model.logPerplexity(dataset)
println(s"The lower bound on the log likelihood of the entire corpus: $ll")
println(s"The upper bound on perplexity: $lp")

// Describe topics.
val topics = model.describeTopics(3)
println("The topics described by their top-weighted terms:")
topics.show(false)

// Shows the result.
val transformed = model.transform(dataset)
transformed.show(false)
```
Use Cases Example: Recommendation with Community Clustering or Information Flow

Exploiting Dynamic Patterns for Recommendation Systems:

- Community based Dynamic Recommendation [Song et al. SDM’06]
- Personalized Recommendation Driven by Information Flow [Song et al. SIGIR’06]
In E-Commerce: Rogers’ Diffusion of Innovations Theory

Users’ adoption patterns: Some users tend to adopt innovations earlier than others → Information virtually flows from early adopters to late adopters
Recommendation Driven by Information Flow

Influence is not symmetric!
Topic-Sensitive Early Adoption Based Information Flow (TEABIF) Network

Adoption is typically category specific

An early adopter of fashion may not be an early adopter of technology
Scheme Overview

Leverage the uneven influence

Dataset → EABIF → Information Propagation Model → Application: Personalized Recommendation

EABIF: Early Adoption based Information Flow Network
EABIF (1) -- Markov Chain

A Markov chain has two components
- A network structure where each node is called a state
- A transition probability of traversing a link given that the chain is in a state
  (P: transition probability matrix)

A stationary distribution is a probability distribution \( q \) such that \( q = qP \)
how likely you will stay at one node

Application -- PageRank [Brin and Page ‘98]
- Assumption: A link from page A to page B is a recommendation of page B
  by the author of A
  → Quality of a page is related to
    Number of pages linking to it
    The quality of pages linking to it
- Assume the web is a Markov chain
  PageRank = stationary distribution of this Markov chain
**EABIF (2)**

Early Adoption Matrix (EAB)
- Count how many items one user accesses earlier than the other – pairwise comparison

Markov Chain Model
- Normalize EAB to a transition matrix $F$ of a Markov chain
- Adjustment $F$ to guarantee the existence of stationary distribution of the Markov chain
  - Make the matrix stochastic
  - Make the Markov chain irreducible

$$
\sum_j F_{ij} = 1
$$
$$
\overline{F}_{ij} \neq 0, \quad 1 \leq i, j \leq N
$$
Information Propagation Models

1. Summation of various propagation steps

\[ F_{if(m)} = \left( \bar{F} + \bar{F}^{(2)} + \bar{F}^{(m)} \right)/m \]

2. Direct summation

\[ F_{if(d)} = \frac{\bar{F} \cdot \left( I - \bar{F}^{(N-1)} \right)}{I - \bar{F}} \]

3. Exponential weighted summation \( N \): number of the nodes

\[ F_{if(exp)} = \left( \beta \bar{F} \right) + \frac{1}{2!} \left( \beta \bar{F} \right)^2 + \frac{1}{(N-1)!} \left( \beta \bar{F} \right)^{(N-1)} + \text{exp} (-\beta) \]
Experimental Setup

ER dataset

- 2004 Apr. to 2005 Apr. as training data
- 2005 May to 2005 Jul. as test data
  1033 users, 586 documents

Process

- Construct information flow network based on the training data
- Trigger earliest users to start the process
- Predict who will be also interested in these documents

Evaluation

- Precision & Recall
Experimental Results -- Recommendation Quality

Comparing to Collaborative Filtering (CF),
precision: EABIF is 91.0% better, TEABIF is 108.5% better
Recall: EABIF is 87.1% better, TEABIF is 112.8% better
Experimental Results --

Propagation Performance

→ TEABIF with exponential weighted summation (\( \beta = 3 \)) achieves the best performance: improves 108.5% on precision and 116.9% on recall comparing to CF
Summary

Exploit dynamic patterns including

- Leverage dynamic patterns from both documents and users’ perspective
  - Analyzing documents accessing types
  - Predicting documents’ expiration date
  - Detecting users’ intentions
  - Identifying users’ interests evolving over time – CTC model
  - Ranking the documents adaptively – Time-sensitive Adaboost

- Utilize users’ adoption patterns
  - Information virtually flows from early adopters to late adopters

Experimental results demonstrate

- Dynamic factors are important for recommendations
Community based recommendation (1) -- AdaBoost Modeling

- Adaboost [Freund and Schapire 1996]
  - Constructing a “strong” learner as a linear combination of weak learners

  - Start with a uniform distribution (“weights”) over training examples
    (The weights tell the weak learning algorithm which examples are important)

  - Obtain a weak classifier from the weak learning algorithm, $h_j : X \rightarrow \{-1, 1\}$

  - Increase the weights on the training examples that were misclassified

  - (Repeat)

The final classifier is a linear combination of the weak classifiers obtained at all iterations

$$f_{final}(x) = sign \left( \sum_{s=1}^{S} \alpha_s h_s(x) \right)$$
Community based recommendation (2) -- Time-Sensitive AdaBoost Modeling

In AdaBoost, the goal is to minimize the energy function:

\[
\sum_{i=1}^{N} \exp \left( -c_i \sum_{s=1}^{S} \alpha_s h_s(x_i) \right)
\]

All samples are regarded equally important at the beginning of the learning process.

We proposed a time-adaptive AdaBoost algorithm that assigns larger weights to the latest documents to indicate their importance.

- **Weak learners**
  - linear classifiers corresponding to the content, community and dynamic patterns

\[
\sum_{i=1}^{N} \exp \left( -c_i \sum_{s=1}^{S} \alpha_s \exp \left( -\tau \cdot (t - t_i) \right) h_s(x_i, t) \right)
\]
Experiments - Methodology

Dataset

“EigyoRyoku” (Sales-Force) system (31,927 users, 26,631 documents)

Log files

- Apr. 2004 to Mar. 2005
  - Test data – Mar. 2005

Nine user actions

- "Search", "Register", "Update", "Delete"

Evaluation -- user satisfaction

How many people really downloaded the documents among these five recommendations during the testing period
Formal Community Recommendations

Performance based on formal communities

- Global Upper Bound
- Collaborative Filtering
- Static Community
- CBDR
- Community Upper Bound

90.4% of Community upper bound

No. of people

1 useful | 2 useful | 3 useful | 4 useful | 5 useful

90.4% of Community upper bound
Informal Community Recommendations

Performance based on informal communities

113% of Formal community’s

95.5% of Community upper bound
Questions?