CSC 390
Topics in Artificial Intelligence

"Unsupervised Machine Learning"

Fall 2016

Prof. Sara Mathieson
Smith College

Outline: 11/8

- Today:
  - **Jenny** 10:30am
  - **Alice** 10:45am
  - **Isha** 11:00am
  - **Yvaine** 11:15am
  - **Karen** 11:30am

- Office Hours today: 4-5pm (CSC 240), Ford 355
- Homework 6 (Project Proposal) due Nov 17 (Thurs)

- Presenters:
  - Speak loudly
  - I will give you a 2 min warning after 10 minutes

- Audience:
  - Give presenters your full attention and ask questions
  - I will ask questions too

Just One More: Modeling Binge Watching Behavior

William Trouleau, Azin Ashkan, Weicong Ding, Brian Ericcson

Jenny Huang

CSC 390


NETFLIX

Key Terms

- **Poisson distribution** $$f(v_i) = \Pr(v_i; \lambda) = \frac{\lambda^{v_i}}{e^{\lambda} v_i!}$$
- A watching **session**: all interactions of a user containing watching at least one episode of television and with less than an hour between interactions.
- **Censoring behavior**: sessions are censored when the termination of a session may not have been by user choice

Table 2: List of notations.

Notation	Definition
$\mathbf{x}_i$	Covariate vector of each session i
v_i	The number of views in each session i
h_i	The number of episodes available for session i
c_i	Censorship indicator for each session i
$\lambda_{i,k}$	Poisson parameter for component $k \in \{1, \ldots, K\}$ and corresponding to session i
$\boldsymbol{\beta}_k$	Coefficient vector for component $k \in \{1, \ldots, K\}$
π_k	Mixture weight for component $k \in \{1, \ldots, K\}$

Methods

- Censored Expectation-Maximization Algorithm

Algorithm 1 Censored EM Algorithm: `EM-fit`

Input: Session covariates $\{x_i\}$, consumption observations $\{v_i\}$, censorship thresholds $\{h_i\}$, number of iterations T, number of components K.
Initialize parameters $\Theta^{(0)} = (\boldsymbol{\pi}^{(0)}, \boldsymbol{\beta}^{(0)})$
for $t \in 1 \dots T$ **do**
  E-Step: Compute $\tau_{i,k}^{(t)}$ using Equation 7, for $i = 1 \dots N$, $k = 1 \dots K$.
  M-Step: Compute $\pi_k^{(t)}$ using Equation 9, and $\boldsymbol{\beta}_k^{(t)}$ using Equation 10, for $k = 1 \dots K$
end for
Output: Estimated parameter values, $\widehat{\Theta} = (\boldsymbol{\pi}^{(T)}, \boldsymbol{\beta}^{(T)})$

Methods

- Censored Expectation-Maximization Algorithm

Algorithm 1 Censored EM Algorithm: `EM-fit`

Input: Session covariates $\{x_i\}$, consumption observations $\{v_i\}$, censorship thresholds $\{h_i\}$, number of iterations T, number of components K.

Initialize parameters $\Theta^{(0)} = (\boldsymbol{\pi}^{(0)}, \boldsymbol{\beta}^{(0)})$

for $t \in 1 \dots T$ **do**

E-Step: Compute $\tau_{i,k}^{(t)}$ using Equation 7, for $i = 1 \dots N$, $k = 1 \dots K$.

M-Step: Compute $\pi_k^{(t)}$ using Equation 9, and $\boldsymbol{\beta}_k^{(t)}$ using Equation 10, for $k = 1 \dots K$

end for

Output: Estimated parameter values, $\widehat{\Theta} = (\boldsymbol{\pi}^{(T)}, \boldsymbol{\beta}^{(T)})$

Methods

- Censored Expectation-Maximization Algorithm

Algorithm 1 Censored EM Algorithm: `EM-fit`

Input: Session covariates $\{x_i\}$, consumption observations $\{v_i\}$, censorship thresholds $\{h_i\}$, number of iterations T, number of components K.
Initialize parameters $\Theta^{(0)} = (\boldsymbol{\pi}^{(0)}, \boldsymbol{\beta}^{(0)})$
for $t \in 1 \ldots T$ **do**
  E-Step: Compute $\tau_{i,k}^{(t)}$ using Equation 7, for $i = 1 \ldots N$, $k = 1 \ldots K$.
  M-Step: Compute $\pi_k^{(t)}$ using Equation 9, and $\boldsymbol{\beta}_k^{(t)}$ using Equation 10, for $k = 1 \ldots K$
end for
Output: Estimated parameter values, $\widehat{\Theta} = (\boldsymbol{\pi}^{(T)}, \boldsymbol{\beta}^{(T)})$

Methods

- Censored Expectation-Maximization Algorithm

Algorithm 1 Censored EM Algorithm: `EM-fit`

Input: Session covariates $\{x_i\}$, consumption observations $\{v_i\}$, censorship thresholds $\{h_i\}$, number of iterations T, number of components K.
Initialize parameters $\Theta^{(0)} = (\boldsymbol{\pi}^{(0)}, \boldsymbol{\beta}^{(0)})$
for $t \in 1 \ldots T$ **do**
  E-Step: Compute $\tau_{i,k}^{(t)}$ using Equation 7, for $i = 1 \ldots N$, $k = 1 \ldots K$.
  M-Step: Compute $\pi_k^{(t)}$ using Equation 9, and $\boldsymbol{\beta}_k^{(t)}$ using Equation 10, for $k = 1 \ldots K$
end for
Output: Estimated parameter values, $\widehat{\Theta} = (\boldsymbol{\pi}^{(T)}, \boldsymbol{\beta}^{(T)})$

Methods

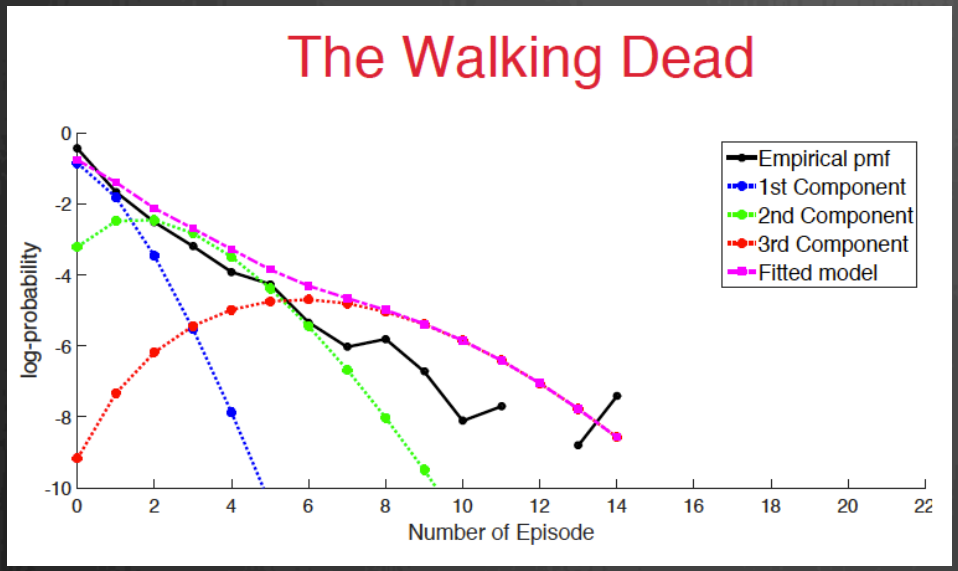
- Predicting the number of episodes in a session

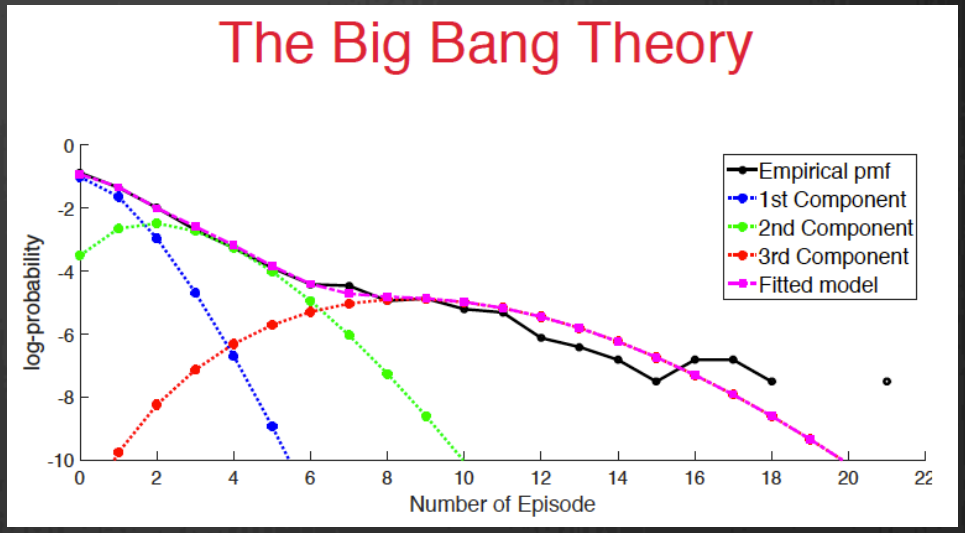
$$\hat{v}_i = \mathbb{E}(v; \boldsymbol{\beta}, \boldsymbol{\pi}) = \sum_{k=1}^{K} \pi_k \mathbb{E}(v; \boldsymbol{\beta}_k)$$

- Predicting if the user will watch the next episode

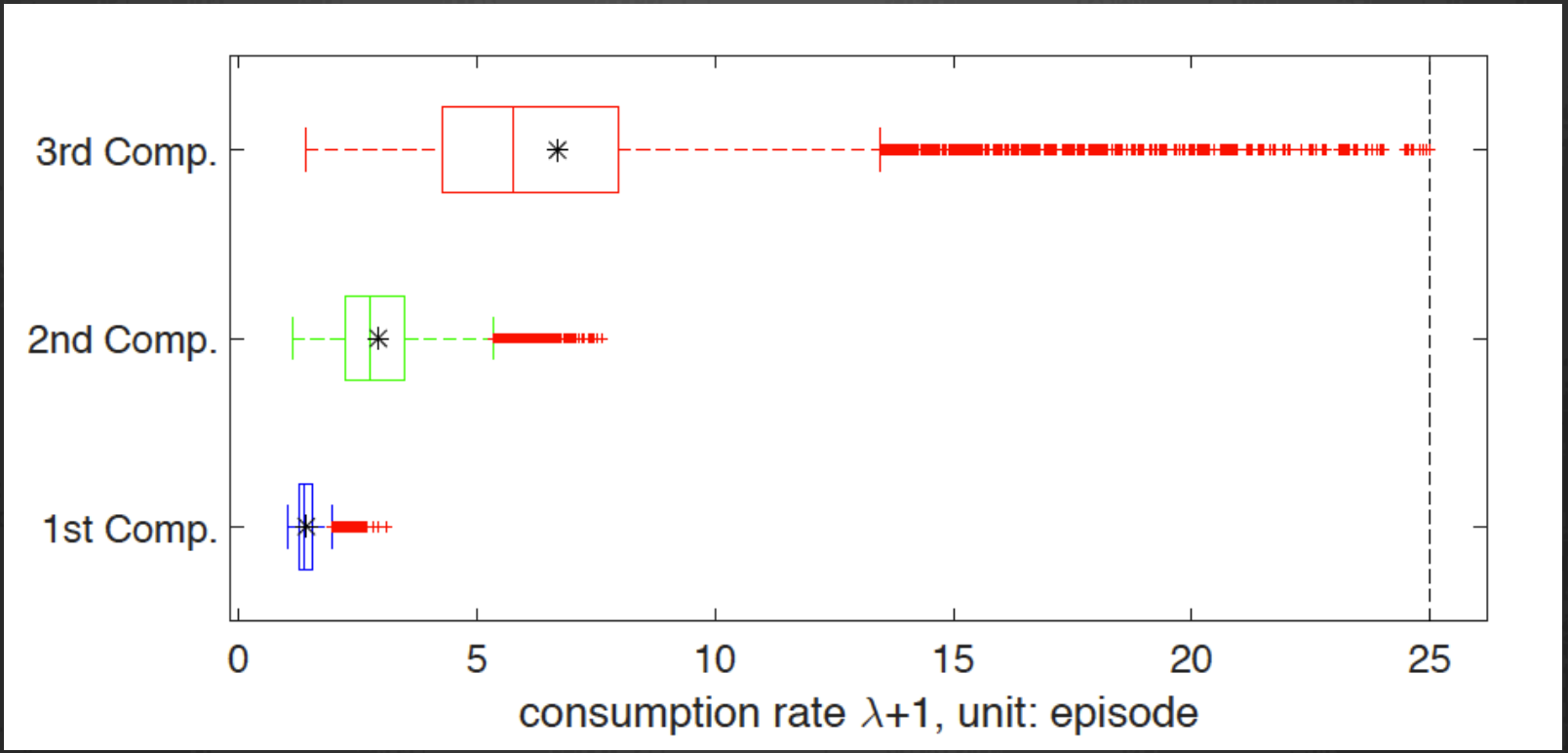
$$\begin{aligned}\Pr(v > v_c | v_c; \boldsymbol{\beta}, \boldsymbol{\pi}) &= \sum_{z=1}^{K} p_z(v > v_c | v_c; \boldsymbol{\beta}, \boldsymbol{\pi}) p(z | v_c; \boldsymbol{\beta}, \boldsymbol{\pi}) \\ &= \sum_{z=1}^{K} \frac{p_z(v > v_c; \boldsymbol{\beta}, \boldsymbol{\pi})}{p_z(v \geq v_c; \boldsymbol{\beta}, \boldsymbol{\pi})} p(z | v_c; \boldsymbol{\beta}, \boldsymbol{\pi})\end{aligned}$$

Results


The Walking Dead
log-probability
Number of Episode
Empirical pmf
1st Component
2nd Component
3rd Component
Fitted model


The Big Bang Theory
log-probability
Number of Episode
Empirical pmf
1st Component
2nd Component
3rd Component
Fitted model

Results


3rd Comp.
2nd Comp.
1st Comp.
0
5
10
15
20
25
consumption rate λ+1, unit: episode

Conclusion

- Binge watching depends on many factors
- Algorithm very similar to K-means to determine a different model
- Analysis of binge consumptions of other types of products

Unsupervised detection and tracking of moving objects for video surveillance applications

Authors: Issam Elafi, Mohamed Jedra, Noureddine Zahid

Presented by: Alice Yang

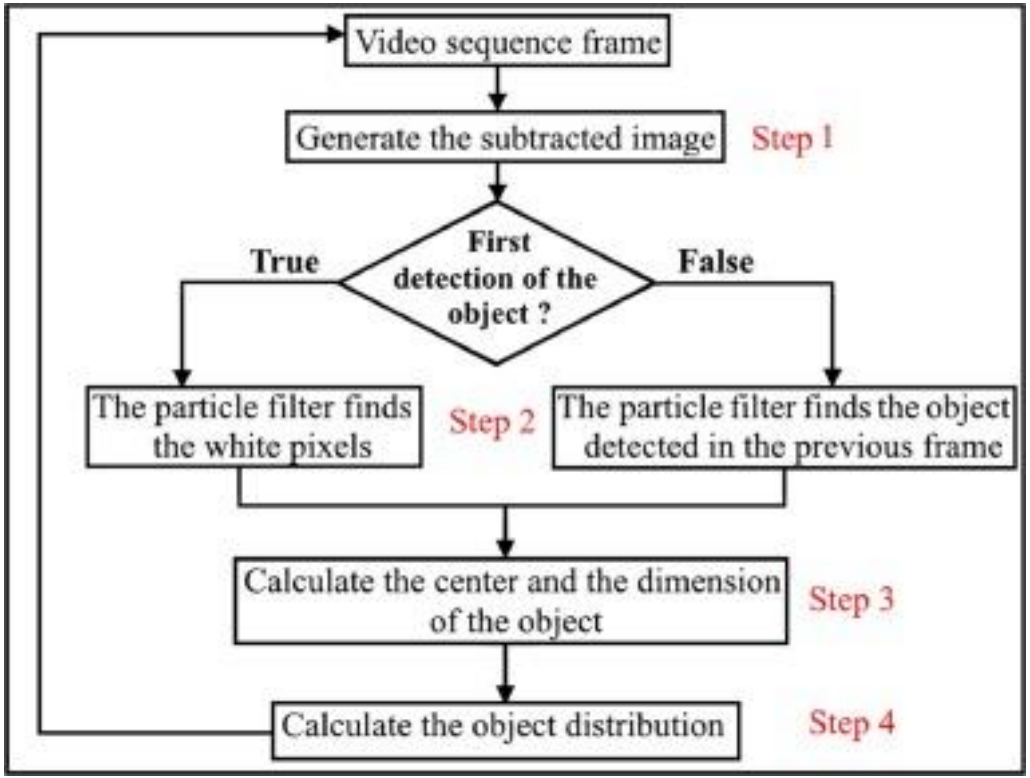
Motivation

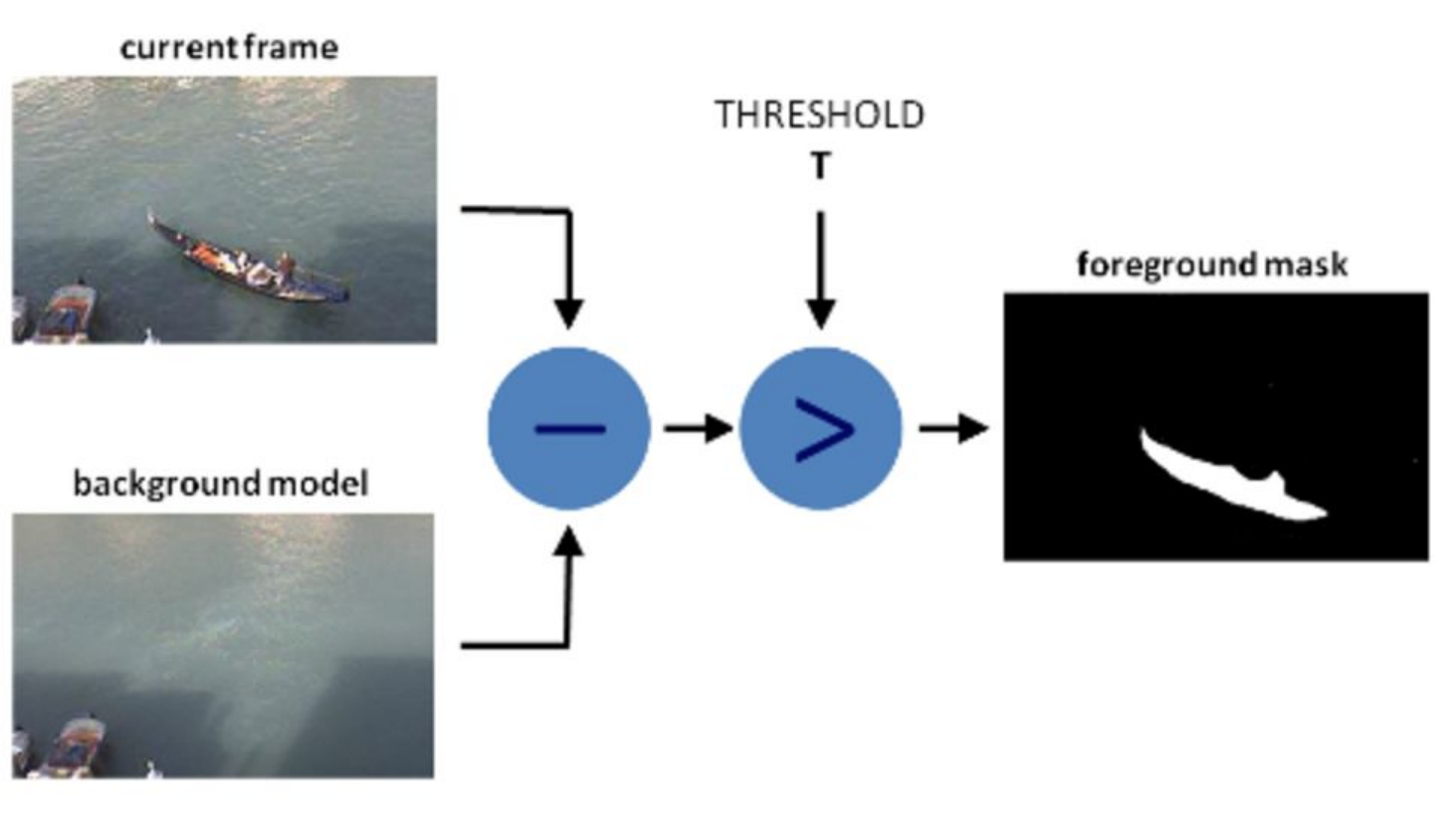
Key Terminology

- **Detection Method: Background Subtraction**
  - based on the measurement of appearance change between two consecutive frames

- **Tracking Method: Particle Filtering**
  - Estimate the state of the system at time *t*, taking into account its state at *t*-1.

Method Overview




current frame
THRESHOLD
T
foreground mask
background model

Particle Filtering

State vector for image: $\mathbf{s = \{x, y, v_x, v_y\}}$

A: deterministic component

w: randomly chosen Gaussian noise

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

State vector at time t: $\mathbf{s_t = As_{t-1} + w_{t-1}}$

- Given the sample set s_{t-1} and the image I_{SB} generated using the subtraction background.

- *nf = 1*

For the *nf* particle filters **do**:

- **Propagation:** each particle is propagated from the set s_{t-1}:

$$s_{t-1}^{(N)} = A s_{t-1}^{(N)} + w_{t-1}^{(N)}$$

where $w_{t-1}^{(N)}$ is a Gaussian random variable and *N* is the number of particles

- **Evaluation:** calculate the weight of particles.

- **If** (the first detection of the object):

$$\omega_n = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(cp-tp)^2}{2\sigma^2}}$$

nf=nf+1

Else

$$\omega_n = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(1-\rho[p_s,q])}{2\sigma^2}}$$

End if

- Compute the normalized weights.

- **Resampling:** select the *N* next samples using the Sampling Importance Re-sampling algorithm (SIR) [49]

- **Estimation**: estimate the average state of the set $s_t^{(N)}$.

- **Removal**: remove the object from the initial frame I_{SB}

- Estimate the dimension of the object and calculate the histogram distribution.

- **If (**the object no longer exists**)**

nf=nf−1

End if

- Create a black square in the same zone of the object.

End for

Formulas

- Weights of each particle for a new detected object:

$$\omega_n = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(cp-tp)^2}{2\sigma^2}} \; tp \in [150, 255]$$

- Weights of each particle for an already detected object:

$$\omega_n \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(1-\rho[ps,q])}{2\sigma^2}}$$

Formulas

- Find center of object

$$E = \sum_{n=1}^{N} \omega_n * s_t^n$$

- Calculate dimension of object

Experiment

6 Challenging Sequences

- Walk1 (Illumination)
- Browse1 (little objects, background clusters)
- BrowseWhileWaitting1 (scale variation)
- OneStopEnter2cor (scale variation)
- OneStopEnter2front (deformation)
- Walking (size variation)

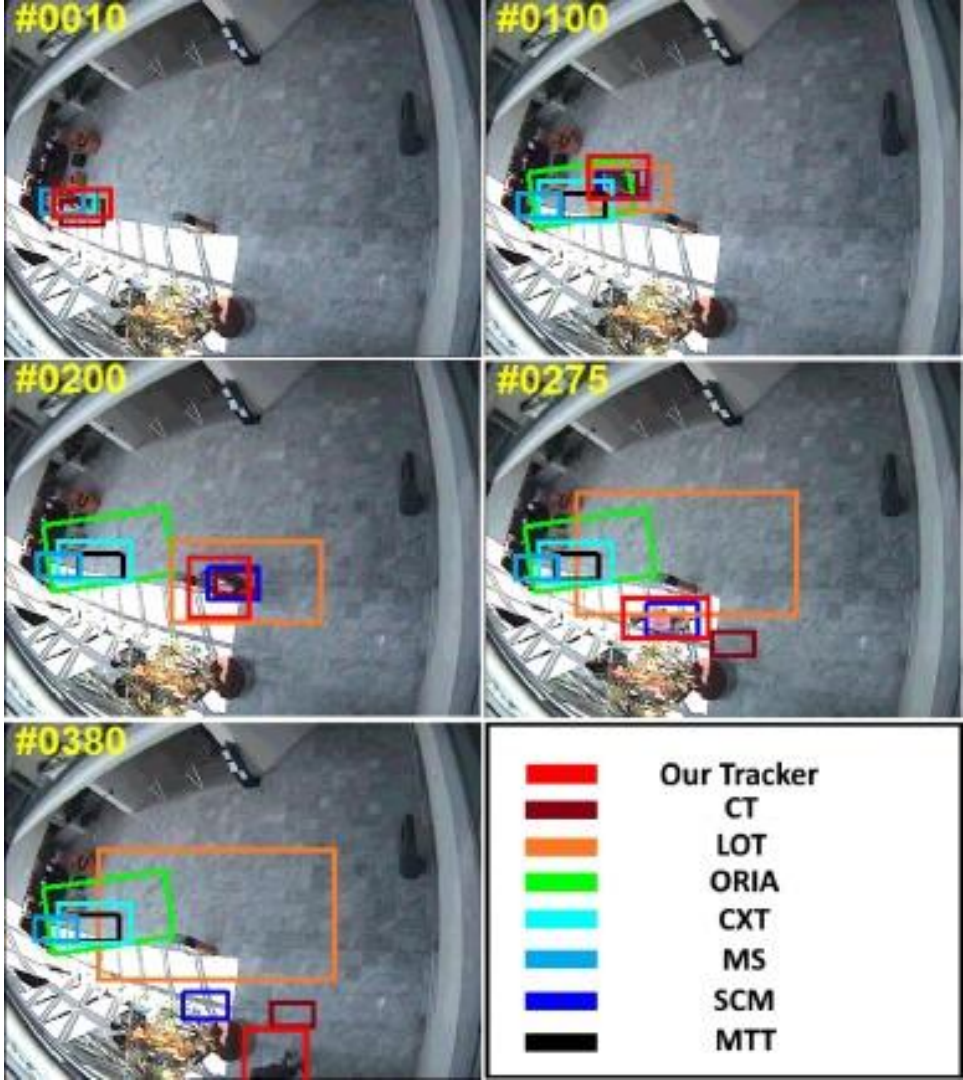
Datasets: CAVIAR dataset and Walking from the VTB dataset.

7 Other Algorithms

- Online Robust Image Alignment
- Locally Orderless Tracking
- Sparsity-based Collaborative Model
- Fast Compressive Tracking
- Multi-task Sparsh Learning
- Context Tracker
- Online Selection of Discriminative Tracking Features

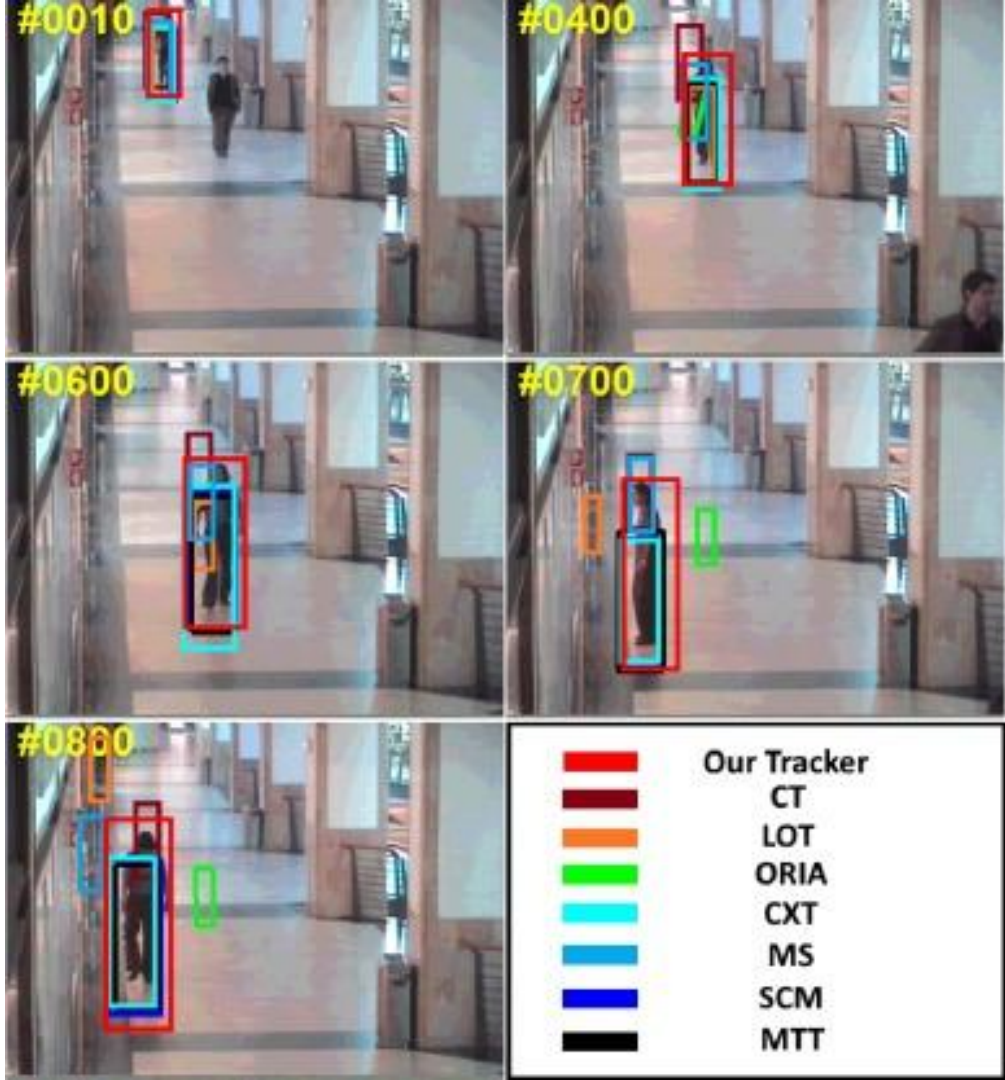
Walk 1

(Illumination)



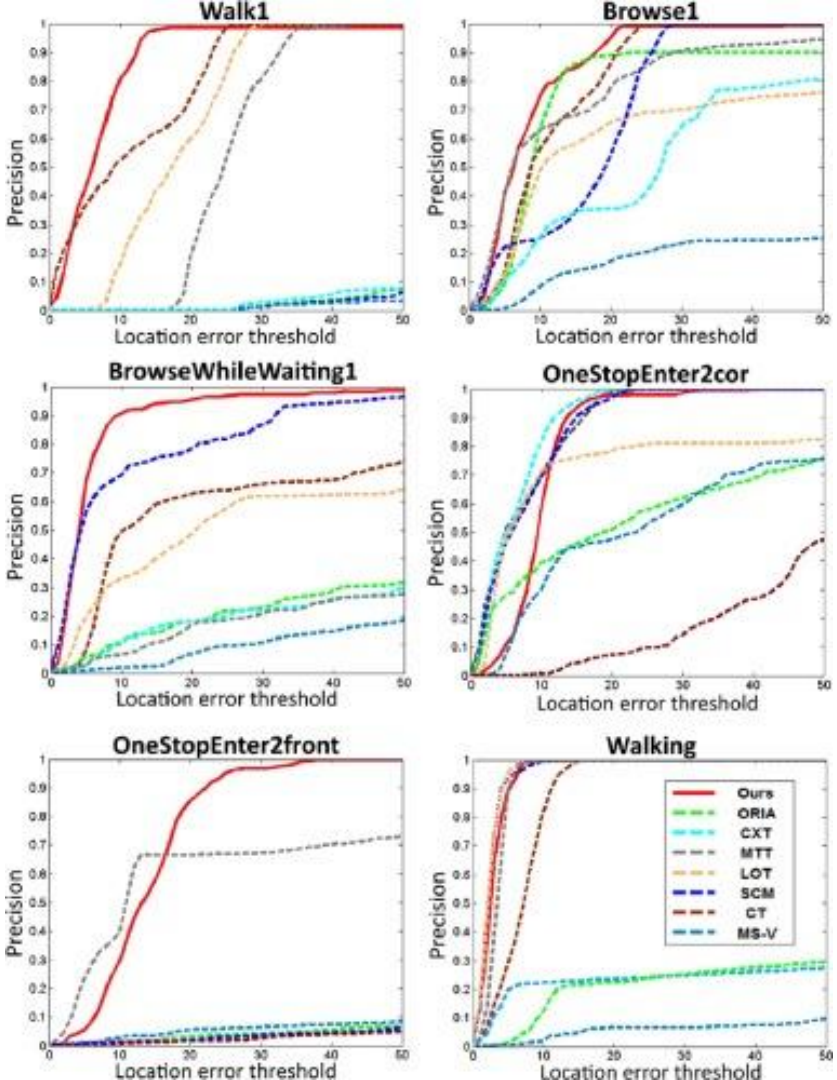
OneStopEnter2cor

(Scale Variation)



Results

Local error: Euclidian distance between centroids of ground truth window and tracked window



Conclusions

Multiple objects

No prior needed

Works well compared to other algorithms

Communication Efficient Distributed Kernel Principal Component Analysis

M.F. Balcan, Y. Liang , L.Song , D. Woodruff and B.Xie

Presented by Isha Raut

KEY TERMINOLOGY & CONCEPTS

Distributed Computing

Distributed systems are groups of networked computers, which have the same goal for their work. Machine learning is used in a distributed setting often when there is very large amounts of data.

Kernels

A kernel is a similarity function.A kernel is just a transformation of your input data that allows you to process it more easily. kernel methods require only a user-specified kernel, i.e., a similarity function over pairs of data points in raw representation.

Communication costs

One of the major overheads in the execution of parallel programs arises from communication of information between processing elements. The cost of communication is dependent on a variety of features. If it takes a long time for communication it will create a bottleneck to the nonlinear feature extraction pipeline.

Kernel PCA

Kernel principal component analysis (kernel PCA) is an extension of principal component analysis (PCA) using techniques of kernel methods. Using a kernel, the originally linear operations of PCA are performed in a reproducing kernel Hilbert space.

SVD and Leverage scores

Singular Value decomposition (SVD) is a factorization of a real or complex matrix. Leverage is a measure of how far away the independent variable values of an observation are from those of the other observations.

Subspace embeddings

Subspace Embedding S for a matrix M is so the norm of any vectors in the column space of M is approximately preserved. Subspace embedding is basically approximating the subspace of the original matrix in a lower dimension.

MOTIVATION

KPCA is an algorithm used to extract nonlinear features from complex data sets. It is currently very costly to conduct Kernel PCA in a distributed setting

This paper aims to introduce a communication efficient method of generating a small representative subset of data then perform KPCA on it.

The proposed algorithm is tested with multiple datasets and the results show high quality KPCA results with low communication costs

CHALLENGES

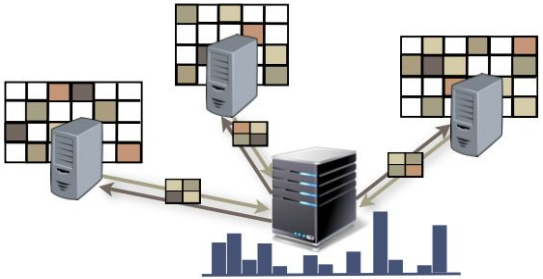
In related work and KPCA in general there are many technical challenges as follows:

- Computing LS is very expensive , not many fast algorithms in the distributed setting
- Unclear how to compute its communication efficient LS ; Existing ways lead to communication linear in the number of data points
- Sampling only with LS do not give good approximations
- There needs to be an algorithm to approximate a good low rank approximation in their span

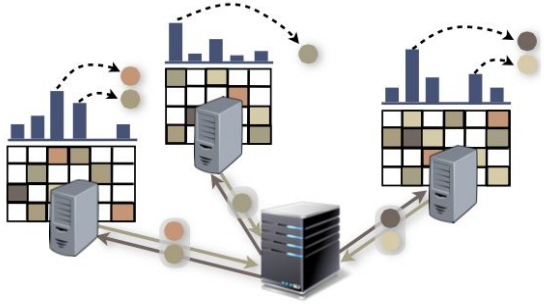
MAIN ALGORITHM

The main distributed KPCA algorithm has four parts to it , each part tackling one of the challenges faced while conducting Kernel Principal Component Analysis

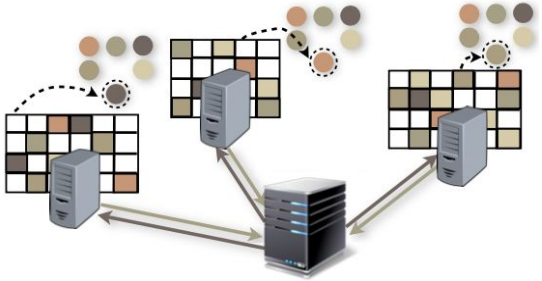
ALGORITHM OVERVIEW



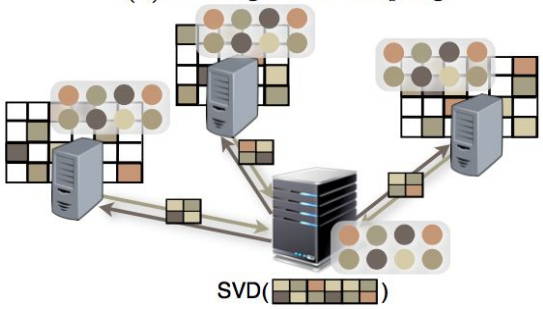
(a) Compress data and compute leverage scores



(b) Leverage score sampling



(c) Adaptive sampling



(d) Project data and compute KPCA

ALGORITHM OVERVIEW

Algorithm 4 Distributed Kernel PCA:
$L = \textbf{disKPCA}(\{A^i\}_{i=1}^s, k, \epsilon, \Delta)$

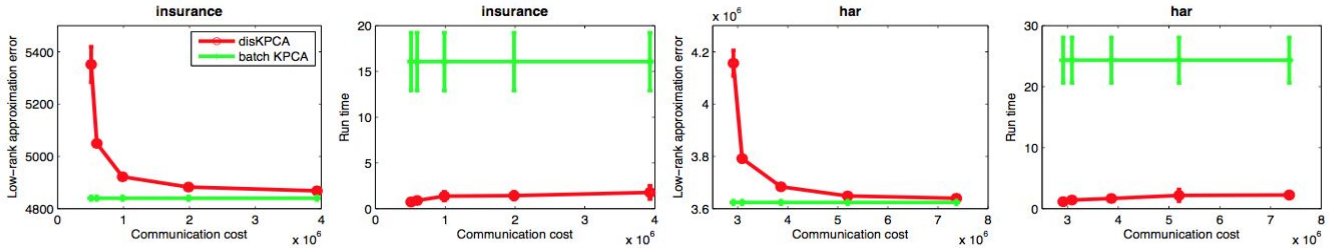
1: Each worker i: do a $(1/4, \Delta)$-good subspace embedding $E^i = S(\phi(A^i)) \in \mathbb{R}^{t \times n_i}, t = O(k)$;
2: Compute the leverage scores: $\{\tilde{\ell}_j^i\} = \textbf{disLS}(\{E^i\}_{i=1}^s, k)$;
3: Sample points: $Y = \textbf{RepSample}(\{A^i\}_{i=1}^s, \{\tilde{\ell}_j^i\}, k, \epsilon)$;
4: Output $L = \textbf{disLR}(\{A^i\}_{i=1}^s, Y, k, \epsilon, \Delta)$.

EXPERIMENTS

- Used 10 datasets from UCI repository to evaluate the algorithm
- Used 2 small datasets for benchmark
- Each data partitioned on different workers ; depending on the size of each data sets there was either 5 to 200
- Testing out the proposed KPCA algorithm as well as standard batch KPCA

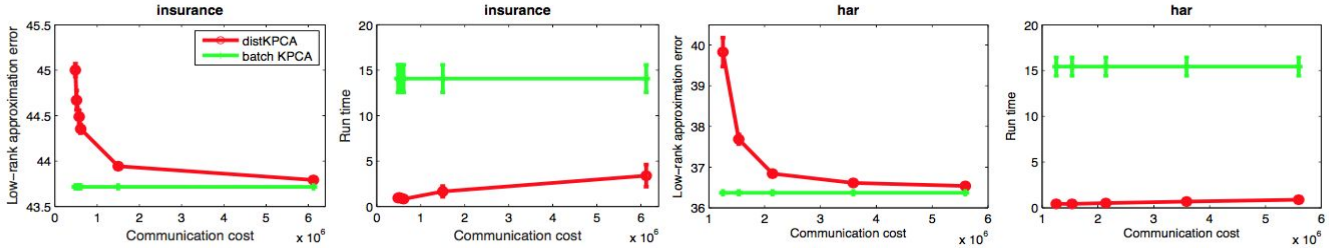
Dataset	d	n	s
bow	100,000	8,000,000	200
higgs	28	11,000,000	200
mnist8m	784	8,000,000	100
susy	18	5,000,000	100
yearpredmsd	90	463,715	10
ctslice	384	53,500	10
20news	61,118	11,269	5
protein	9	41,157	5
har	561	10,299	5
insurance	85	9,822	5

RESULTS



(a) error on insurance (b) runtime on insurance (c) error on har (d) runtime on har

Figure 2: KPCA for polynomial kernels on small datasets: low-rank approximation error and runtime



(a) error on insurance (b) runtime on insurance (c) error on har (d) runtime on har

Figure 3: KPCA for Gaussian kernels on small datasets: low-rank approximation error and runtime

RESULTS

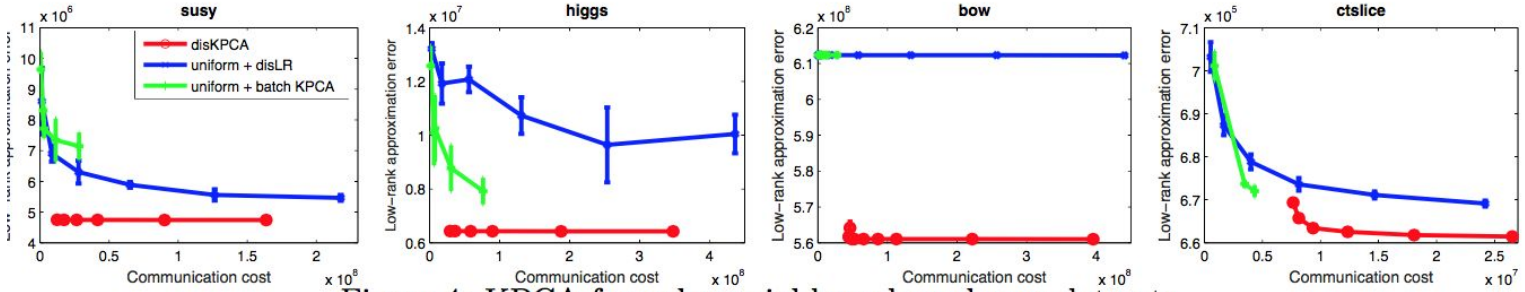


Figure 4: KPCA for polynomial kernels on larger datasets

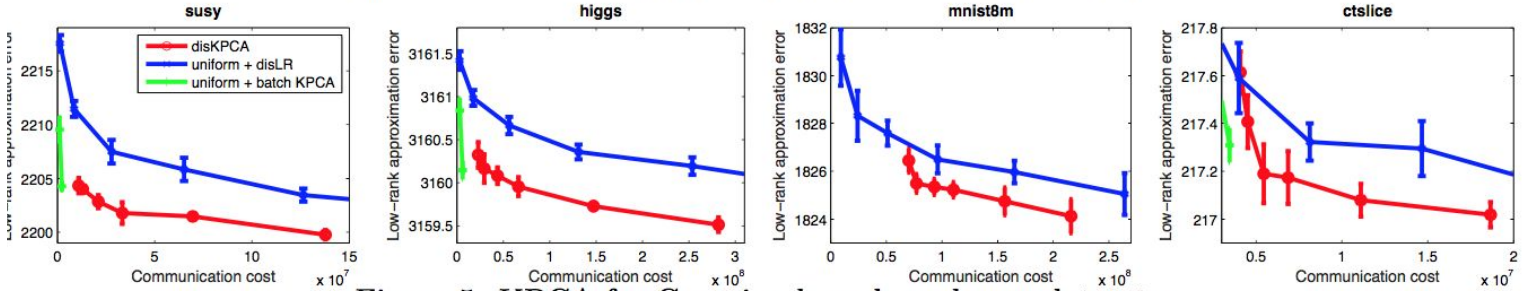
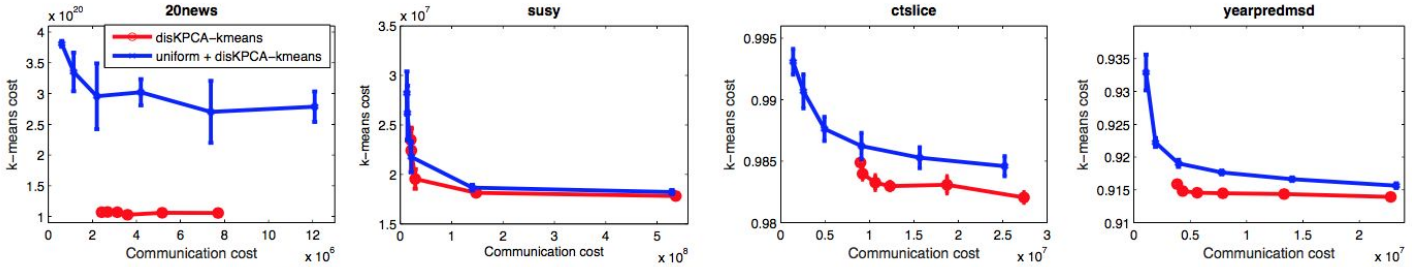


Figure 5: KPCA for Gaussian kernels on larger datasets

DISTRIBUTED SPECTRAL CLUSTERING



(a) Polynomial kernels (b) Gaussian kernels

Figure 8: KPCA + k-means clustering

- Project data on top k PC and apply distributed k means clustering algorithm
- Evaluated by the k means objective
- The proposed algorithm fares well achieves best trade off between communication and error
- Means that this method requires sampling less data to achieve the same loss

CONCLUSIONS

From this algorithm they make contributions such as :
(i) Subspace embedding technique for many kernels

(ii) Distributed algorithm for computing GLS with low communication

(iii) Distributed algorithm for kernel column subset selection

Along with a communication efficient distributed algorithm for KPCA with good approximation and communication and error tradeoff.

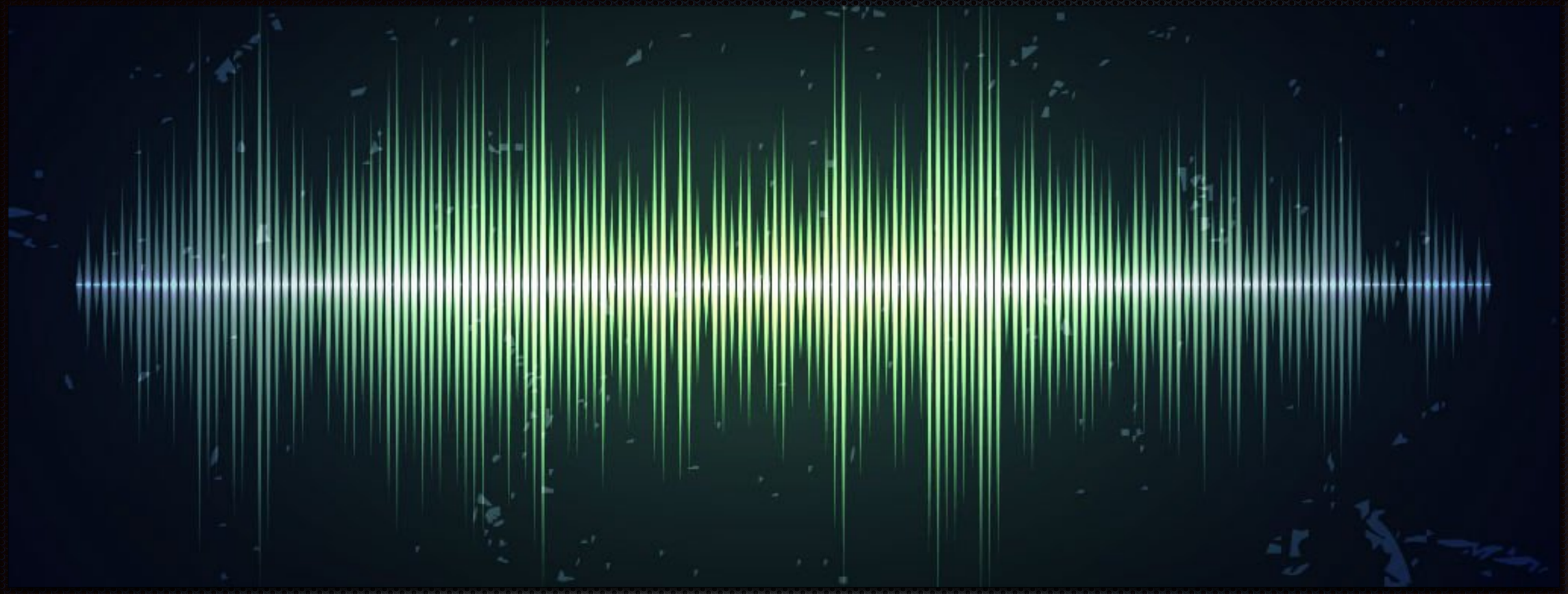
I chose this paper so I could see PCA in an applied setting and the problems it had with Big Data.
It also showed applications of linear algebra and bit of clustering

THANKS!

Any questions?

You can find me at

raut22i@mtholyoke.edu



Assisted keyword indexing for lecture videos using unsupervised keyword spotting

Manish Kanadje, Zachary Millera, Anurag Agarwalb, Roger Gaborskia, Richard Zanibbia, Stephanie Ludi

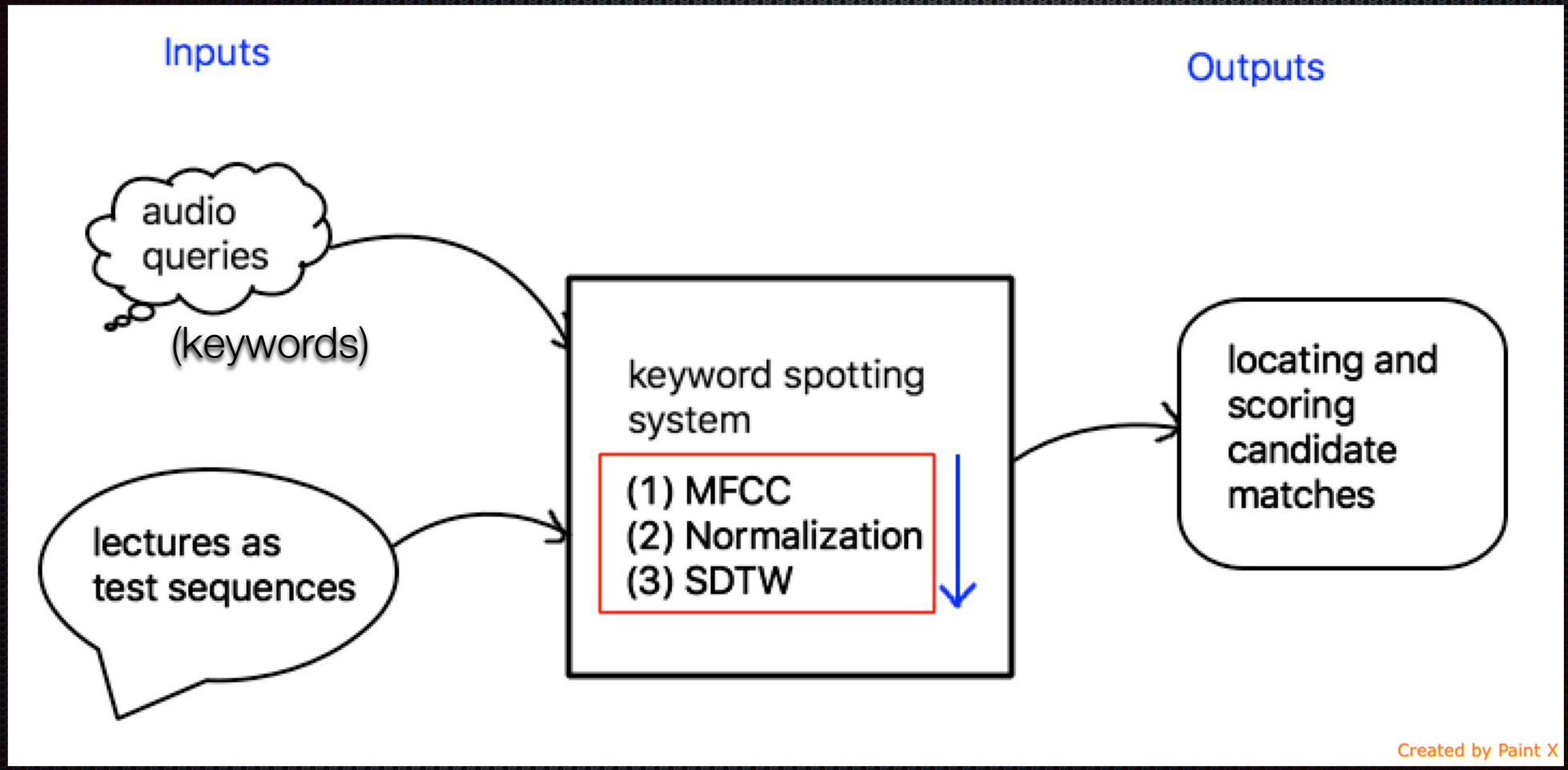
Motivation

- Consider the following scenario:

Solution

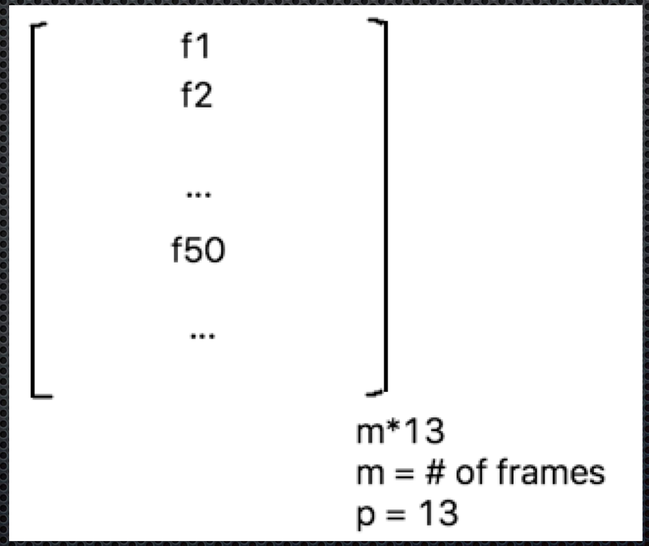
- Video recordings of classroom lectures
- But hard to access and not well indexed, you do a lot of manual scrubbing to find the right place to start. Tedious!
- Wouldn’t it be nice to have indexed videos with keywords so you can jump right to the spot?

Unsupervised Keyword Spotting System



1. MFCC — Mel Frequence Cepstral Coefficients extraction

- A way to represent speech audio mathematically
- Transform and convert raw frames of audio into vectors, extracting features
- raw speech audio ⟶

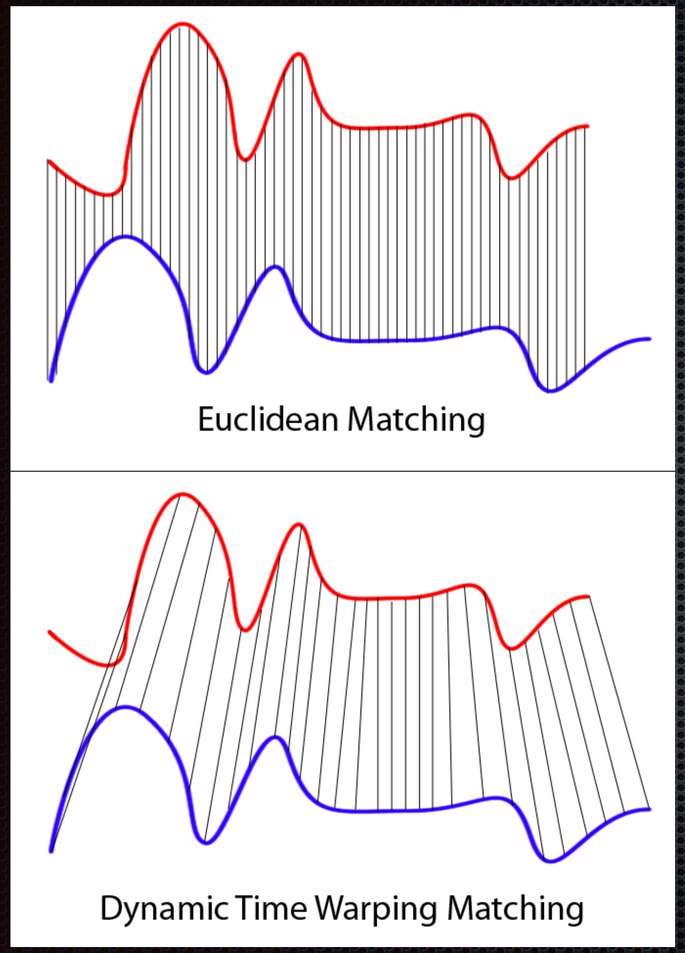


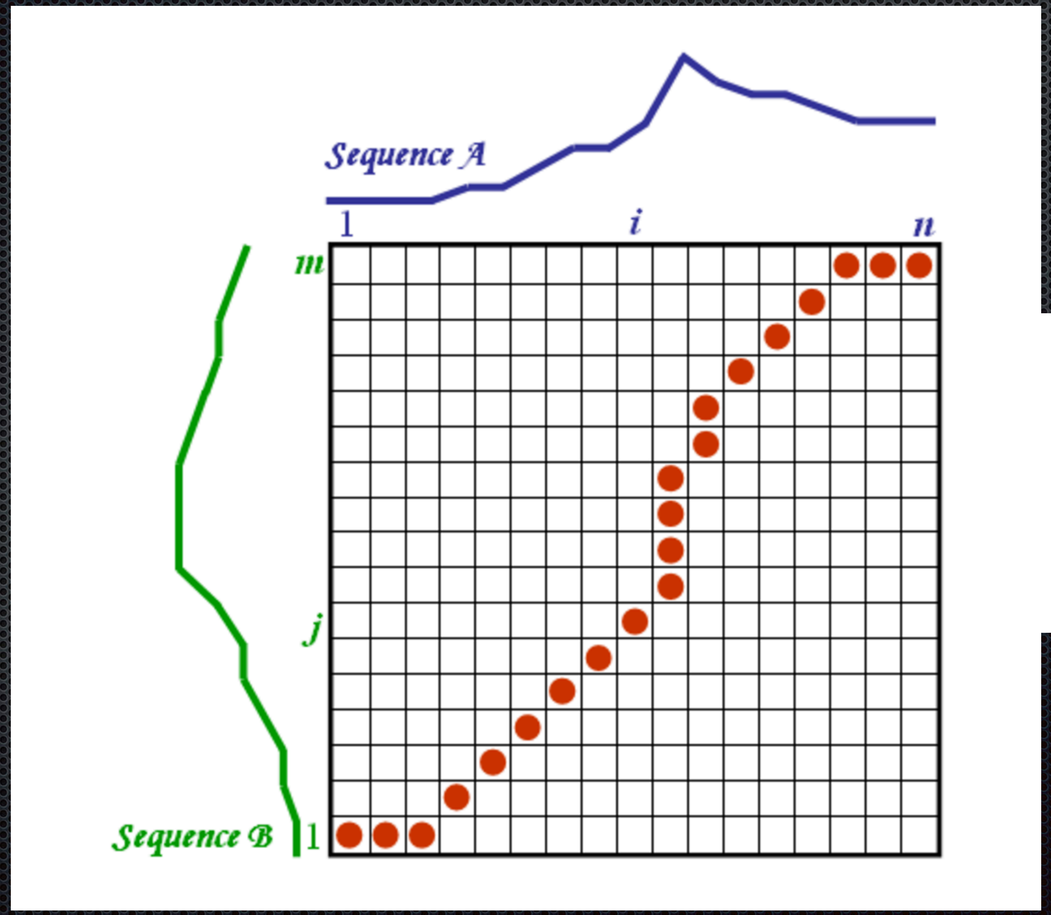
2. Feature Normalization

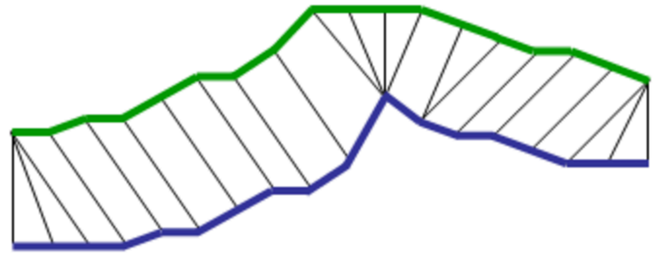
- Designed to reduce the impact of changes in the **recording environment**: laptop recording vs microphone input, in office vs in classroom, other noises. **Increase precision.**
- a. “whitening transform”: $$w_i = \frac{f_i - \mu_i}{\sigma_i}$$
- b. whitened MFCC feature vectors are converted to unit vectors

3. SDTW — Segmental Dynamic Time Warping

- A little bit about DTW first…



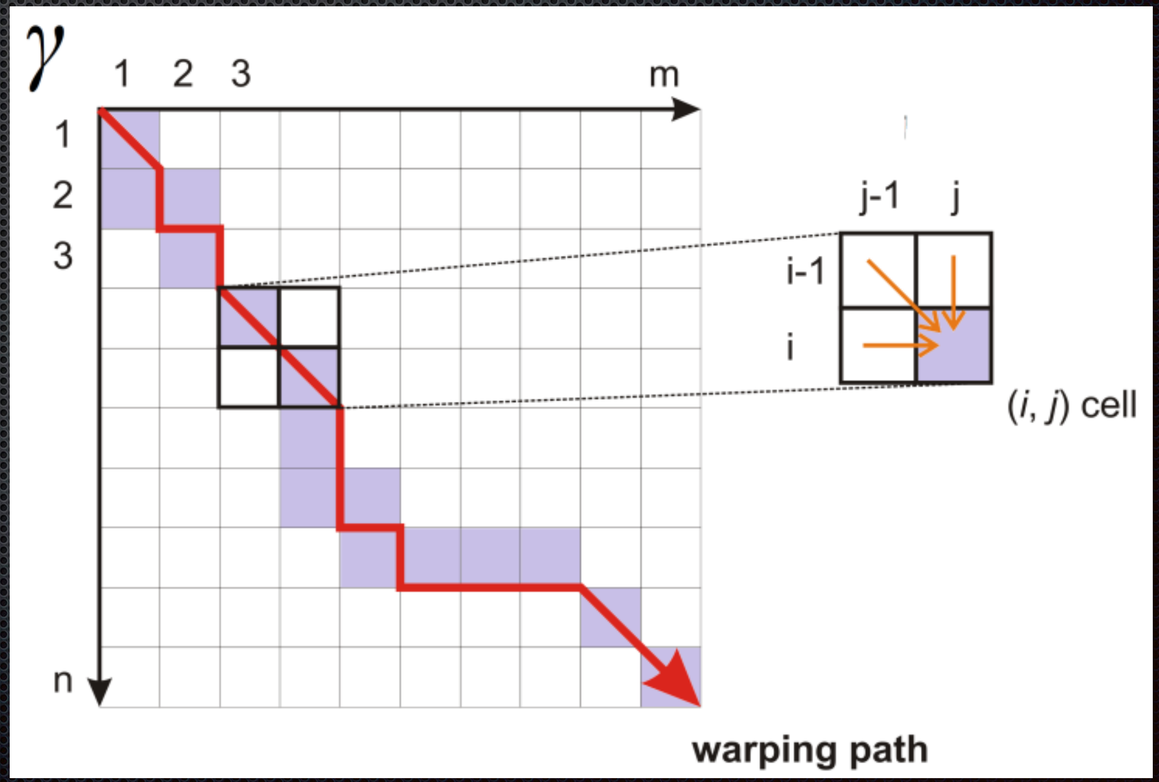




3. SDTW — Segmental Dynamic Time Warping

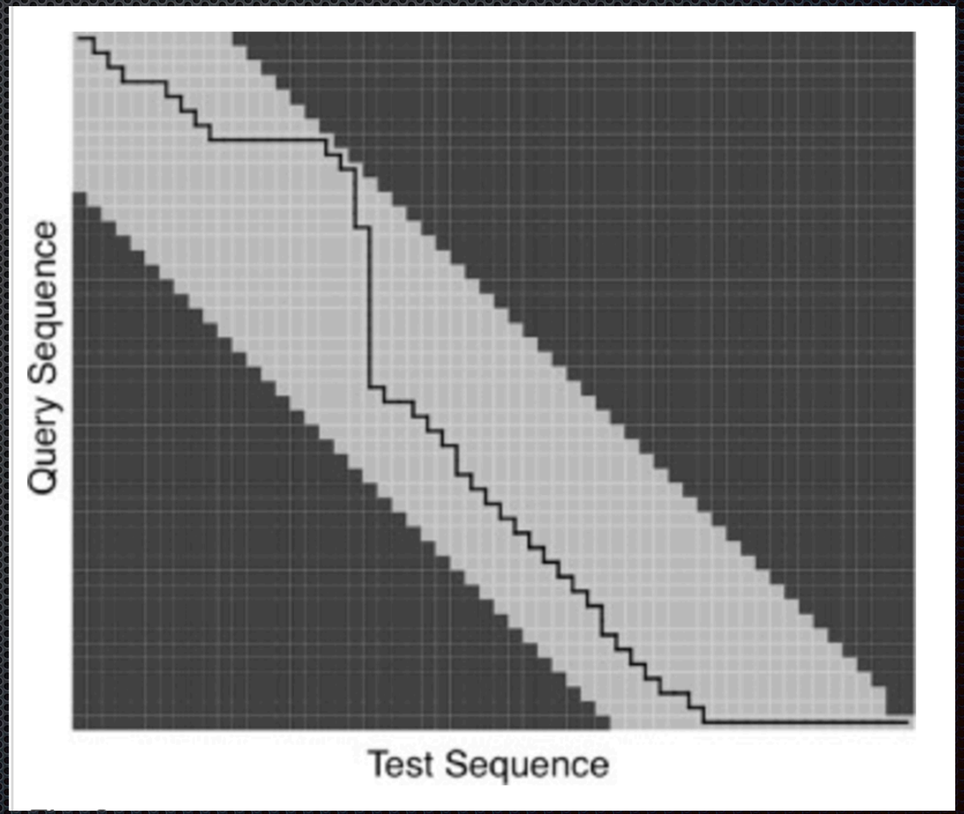
- Determining the minimum warping path:
  - using the Euclidean distance between Q_i and S_j
  - Computed recursively:

$$C_{i,j} = d(Q_i, S_j) + min \begin{cases} C_{i,j-1} \\ C_{i-1,j-1} \\ C_{i-1,j} \end{cases}$$



3. SDTW — Segmental Dynamic Time Warping

- **Segmental** DTW restricts the warping path with a radius parameter, r
- Matching is then performed upon different segments of the test sequence, each starting at a different frame



Key Evaluation Metrics In the Experiments

- Precision at 10
- a “hit” is chosen by human among 5 categories: ***Exact***, ***Insertion*** (extra sounds), *Deletion* (missing sounds), *Both*, *No Match* (false positive)
- ***Similar***

Guessing the results…

- “are homogenous” vs “non-homogenous”
- “multiplicative”, “multiply” or “multiplication”?
- Which kind of queries would achieve the highest Precision?
- “reduce”? “produce”?? “introduce”??? 🤔
- laptop queries vs in-lecture

Results

- Average Precision at 10 of **71.5%** and **79.5%** for laptop recorded and in-lecture queries for RIT lectures
- Average Precision at 10 of **89.5%** for MIT lectures recorded on lapel microphone
- pretty accurate and kind of robust, but audio query input expected to be given

	RIT		RIT	MIT
	Laptop	Requery	Manual	
Query	Exp 1	Exp 2	Exp 3	Exp 4
Augmented	5	10	10	3
Dimensional	9	8	10	10
Row-reduced	10	10	10	
(Reduced row)				9
Solution	10	10	10	10
System of equations	10	10	10	8
Transpose	9	10	10	9
Zero vector	10	10	10	9
Echelon form of a matrix	10	10	9	
(Echelon form)				10
Independent	8	4	9	10
Multiplicative	4	3	8	
(Multiplication)				10
System	9	8	7	10
Variables	6	4	6	10
Coefficients	8	9	5	8
Reduce	3	10	5	4
Orthogonal	1	3	4	10
Mean (μ)	7.15	7.80	7.95	8.95
Stdev (σ)	2.96	2.93	2.56	1.97

Questions?

DOPELEARNING: A COMPUTATIONAL APPROACH TO RAP LYRICS GENERATION

Presentation by : Karen Diaz

Motivation Behind paper

- **Objective:** Study the problem of computational creation of rap lyrics
- Interest is motivated by two different perspectives
- 1) Interested in analyzing the formal structure of these lyrics and in developing a model that can lead to generating artistic work.
- 2) Interested in the demand for increase for systems that interact with humans in non-mechanical and pleasant ways

Background Information

- While the study of human generated lyrics is of interest to academics in fields such as linguistics and music, artificial rhyme generation is also relevant for various subfields of computer science
  - *computational creativity, information extraction, and natural language processing.*
  - *the proposed framework could even form the basis for several other text-synthesis problems, such as generation of text or conversation responses. Practical extended applications include automation of tasks, such as customer service, sales, or even news reporting.*
- The work created in this algorithm falls under the kind of creativity known as “combinatorial creativity” where creative results are produced as novel combinations of familiar ideas.

Key Words

- RankSVM
- Neural Network
- NN5
- EndRhyme
- EndRhyme-1
- OtherRhyme
- LineLength
- BOW
- BOW5
- LSA

Online Demo

- Another interesting thing to point out is that the lyrics generator has been deployed as an online tool called DeepBeat
- http://deepbeat.org/

Method

Algorithm 1: Lyrics generation algorithm DeepBeat.

Input: Seed line ℓ_1, length of the lyrics n.
Output: Lyrics $L = (\ell_1, \ell_2, \ldots, \ell_n)$.
$L[1] \leftarrow \ell_1$; // Initialize a list of lines.
for $i \leftarrow 2$ **to** n **do**
  $C \leftarrow$ `retrieve_candidates`$(L[i-1])$; // Sec. 5.2.1
  $\hat{c} \leftarrow NaN$;
  foreach $c \in C$ **do**
    /* Check relevance and feasibility of the candidate. */
    if rel $(c, L) >$ rel $(\hat{c}, L)$ & `rhyme_ok`(c, L) **then**
      $\hat{c} \leftarrow c$;
  $L[i] \leftarrow \hat{c}$;
return L;

ℓ 1 = seed line
n = length of the lyrics
L = lyrics output by the algorithm
C = candidate next lines
c = candidate line in C

Features

- Captures rhyming of lyrics
  - *EndRhyme*
  - *EndRhyme-1*
  - *OtherRhyme*
- Captures structural similarity
  - *LineLength*
  - *BOW*
  - *BOW5*
  - *LSA*
- Captures semantic similarity
  - *NN5*

Method

Algorithm 1: Lyrics generation algorithm DeepBeat.

Input: Seed line ℓ_1, length of the lyrics n.
Output: Lyrics $L = (\ell_1, \ell_2, \ldots, \ell_n)$.
$L[1] \leftarrow \ell_1$; // Initialize a list of lines.
for $i \leftarrow 2$ **to** n **do**
  $C \leftarrow$ `retrieve_candidates`$(L[i-1])$; // Sec. 5.2.1
  $\hat{c} \leftarrow NaN$;
  foreach $c \in C$ **do**
    /* Check relevance and feasibility of the candidate. */
    if rel $(c, L) >$ rel $(\hat{c}, L)$ & `rhyme_ok`(c, L) **then**
      $\hat{c} \leftarrow c$;
  $L[i] \leftarrow \hat{c}$;
return L;

ℓ 1 = seed line
n = length of the lyrics
L = lyrics output by the algorithm
C = candidate next lines
c = candidate line in C

Neural network model

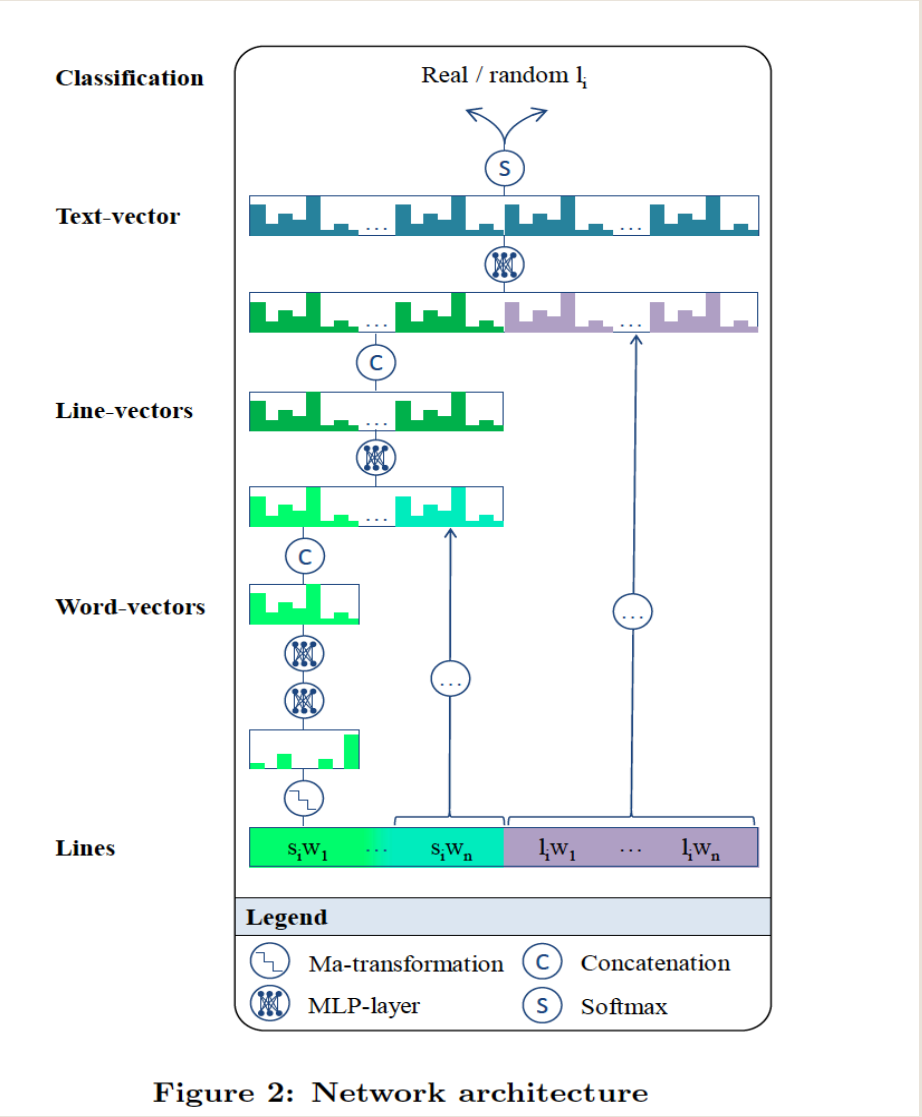


Figure 2: Network architecture

Results

Table 3: Next line prediction results for $k = 300$ candidate lines. MRR stands for mean reciprocal rank and Rec@N for recall when retrieving top N lines.

Feature(s)	Mean rank	MRR	Rec@1	Rec@5	Rec@30	Rec@150
Random	150.5	0.021	0.003	0.017	0.010	0.500
LineLength	117.6	0.030	0.002	0.029	0.177	0.657
EndRhyme	103.2	**0.140**	**0.077**	**0.181**	**0.344**	0.480
EndRhyme-1	126.0	0.075	0.037	0.092	0.205	0.347
OtherRhyme	123.3	0.047	0.016	0.055	0.190	0.604
BOW	112.4	0.116	0.074	0.138	0.280	0.516
BOW5	99.1	0.110	0.065	0.129	0.314	0.708
LSA	111.3	0.089	0.051	0.107	0.262	0.662
NN5	**84.7**	0.067	0.020	0.083	0.319	**0.793**
FastFeats	73.5	0.224	0.160	0.272	0.476	0.802
FastFeatsNN5	61.2	**0.244**	**0.172**	**0.306**	0.524	0.853
All features	**60.8**	0.243	0.169	0.304	**0.527**	**0.855**

Results

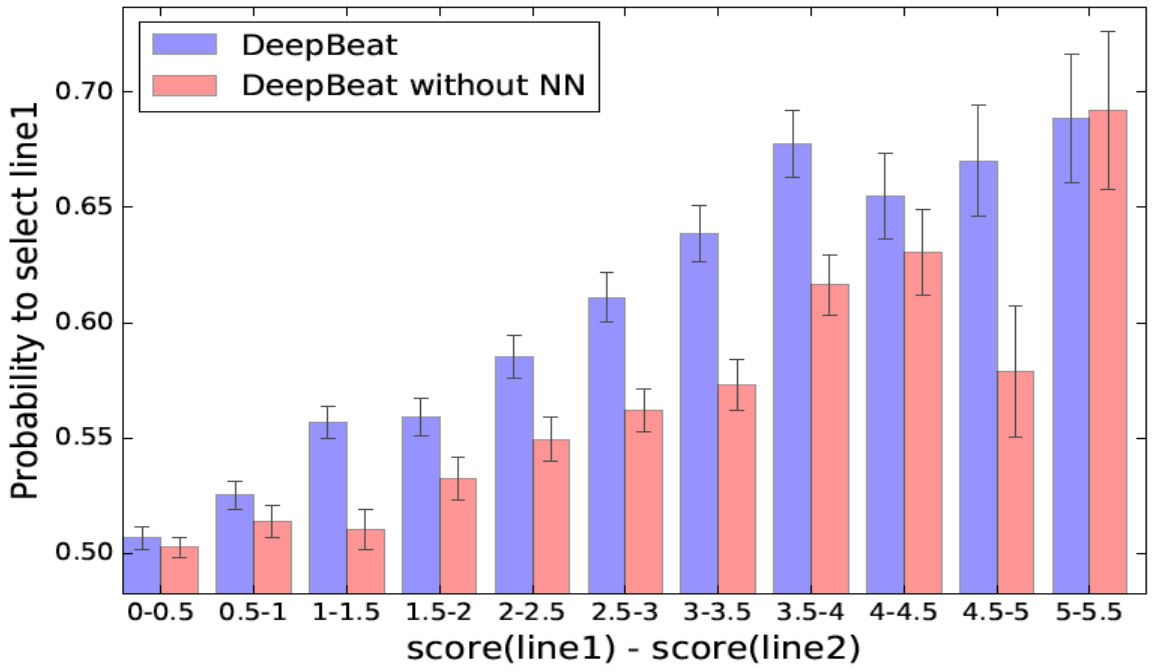


Figure 4: Probability of a `deepbeat.org` user to select a line with a higher score from a pair of lines given the (binned) score difference of the lines. User preferences correlate with the scores assigned by DeepBeat.