Probabilistic Time Series Classification

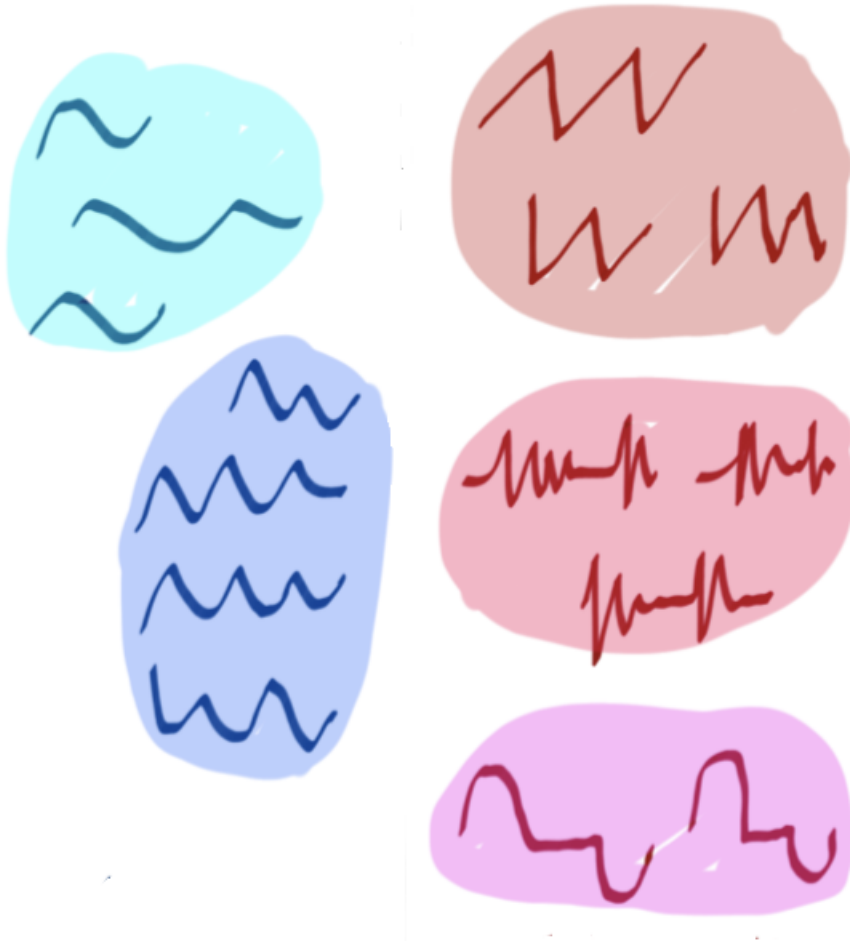
Y. Cem Sübakan

Boğaziçi University

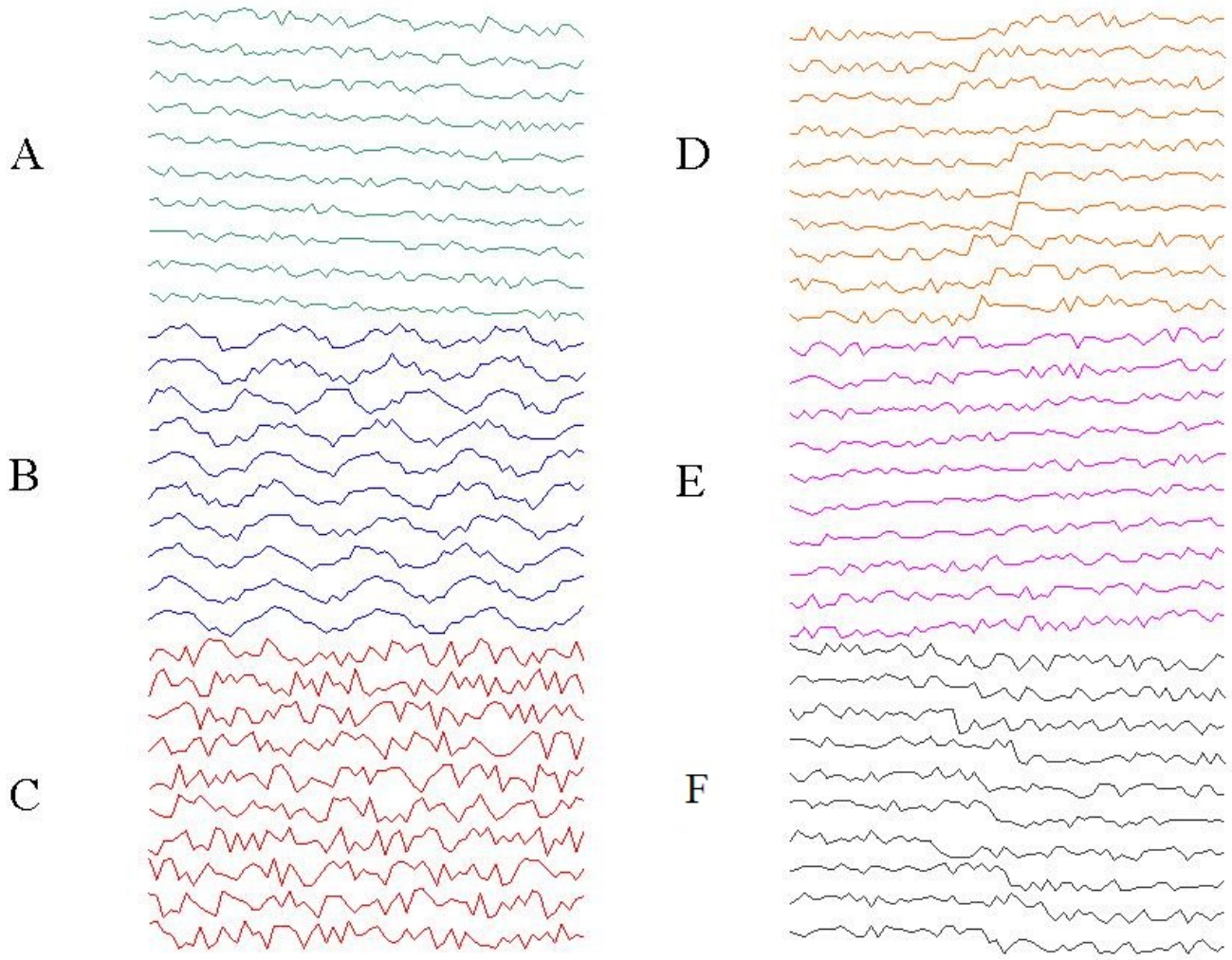
25.06.2013

Problem Statement

- The goal is to assign "similar" sequences to same classes.



Problem Statement



- It is not a trivial task. Variability within each class

Problem Statement

- Data in real life applications is even more challenging.

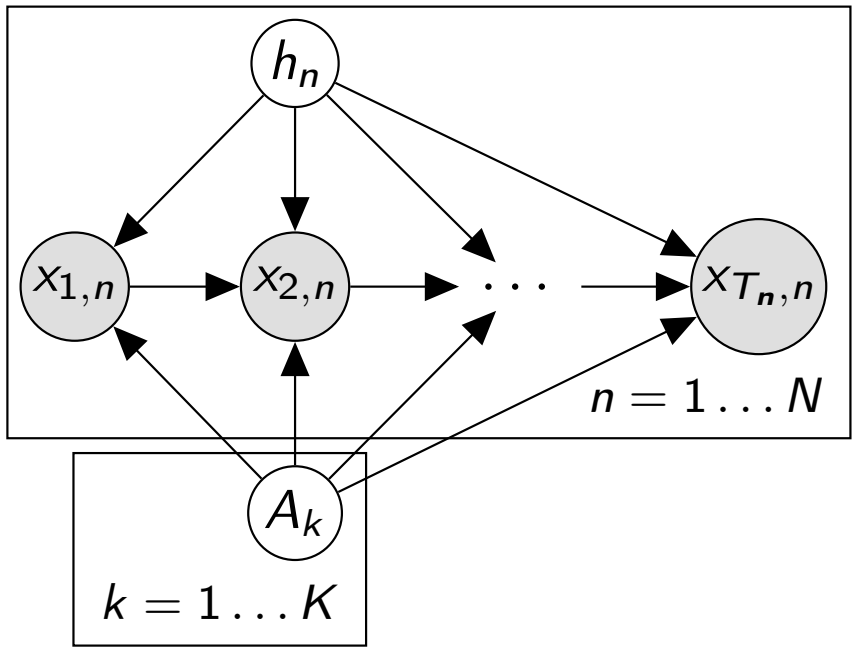
- So, we make a computer program "learn" how to classify sequences.

Goal of this thesis

- The goal of this thesis is to understand and develop learning algorithms for supervised and unsupervised time series classification models.
- We propose two novel learning algorithms for mixture of Markov models and mixture of HMMs .

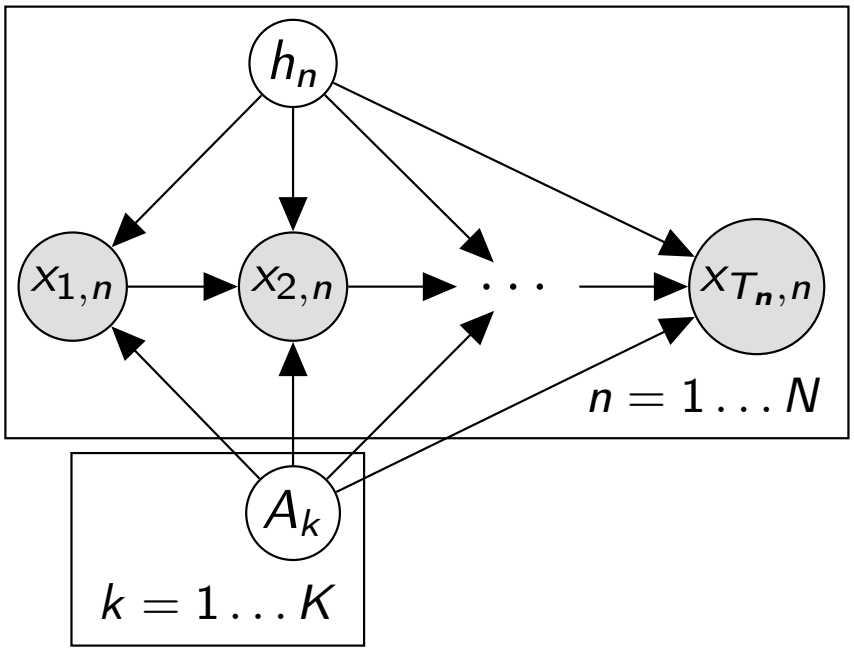
Goal of this thesis

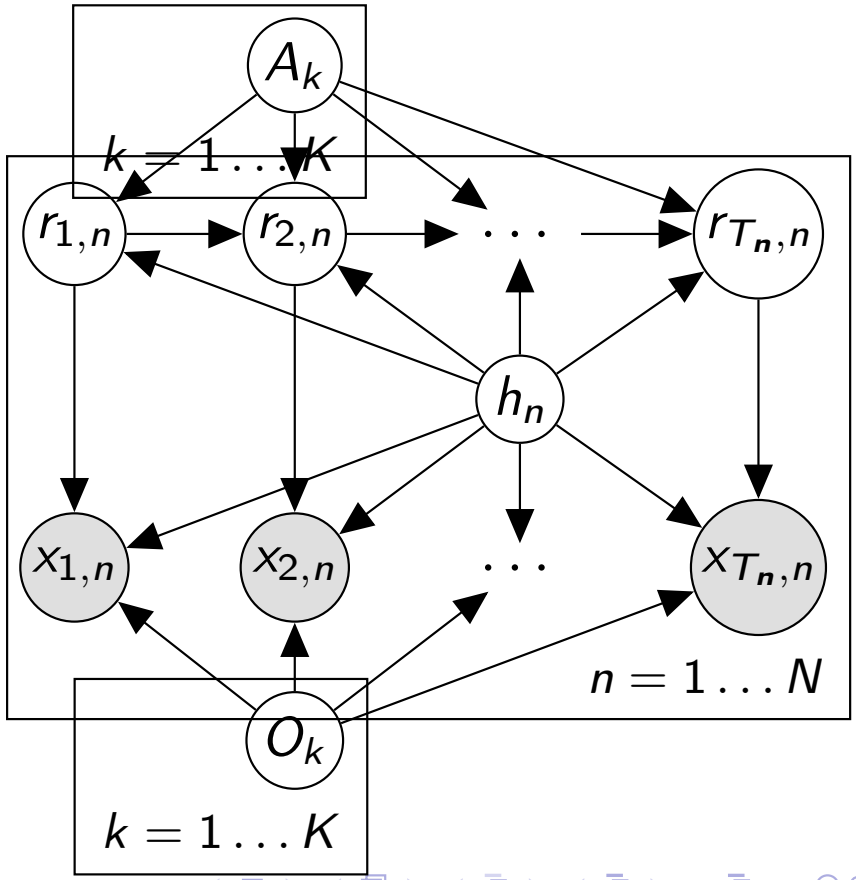
- The goal of this thesis is to understand and develop learning algorithms for supervised and unsupervised time series classification models.
- We propose two novel learning algorithms for mixture of Markov models and mixture of HMMs .



Goal of this thesis

- The goal of this thesis is to understand and develop learning algorithms for supervised and unsupervised time series classification models.
- We propose two novel learning algorithms for mixture of Markov models and mixture of HMMs .





Achievements

- A new spectral learning algorithm for learning mixture of Markov models
  - Based on learning mixture of Dirichlet distributions

Achievements

- A new spectral learning algorithm for learning mixture of Markov models
  - Based on learning mixture of Dirichlet distributions
- A new spectral learning based algorithm for learning mixture of HMMs
  - Faster compared to the conventional EM approach
  - Possible to generalize to infinite mixture of HMMs

Achievements

- A new spectral learning algorithm for learning mixture of Markov models
  - Based on learning mixture of Dirichlet distributions
- A new spectral learning based algorithm for learning mixture of HMMs
  - Faster compared to the conventional EM approach
  - Possible to generalize to infinite mixture of HMMs
- Survey of Discriminative versions of Markov models and HMMs

Outline

EM for latent variable models

- The optimization problem to train a latent variable model:

$$\theta^* = \arg\max_{\theta} p(x|\theta)$$
$$= \arg\max_{\theta} \sum_{h} p(x, h|\theta)$$

  where, x are the observations (sequences), h are the latent variables (cluster indicators/latent state sequences), θ is the model parameter.
- The summation is over all possible combinations of h. An intractable summation in practice.

EM for latent variable models

- The idea is to have a lower bound on $\log p(x|\theta)$, which is easier to maximize.

EM for latent variable models

- The idea is to have a lower bound on $\log p(x|\theta)$, which is easier to maximize.

$$\log p(x|\theta) = \log \sum_h p(x, h|\theta) = \log \sum_h p(x, h|\theta) \frac{q(h)}{q(h)}$$
$$= \log \sum_h \mathbb{E}_{q(h)} \left[\frac{p(x, h|\theta)}{q(h)} \right]$$

EM for latent variable models

- The idea is to have a lower bound on $\log p(x|\theta)$, which is easier to maximize.

$$\log p(x|\theta) = \log \sum_h p(x,h|\theta) = \log \sum_h p(x,h|\theta) \frac{q(h)}{q(h)}$$
$$= \log \sum_h \mathbb{E}_{q(h)} \left[\frac{p(x,h|\theta)}{q(h)} \right]$$

Then, using Jensen's inequality;

$$\log \sum_h \mathbb{E}_{q(h)} \left[\frac{p(x,h|\theta)}{q(h)} \right] \geq \mathbb{E}_{q(h)} \left[\log \frac{p(x,h|\theta)}{q(h)} \right]$$
$$\log p(x|\theta) \geq \underbrace{\mathbb{E}_{q(h)}[\log p(x,h|\theta)] - \mathbb{E}_{q(h)}[\log q(h)]}_{\mathcal{Q}(\theta):=}$$

$\mathcal{Q}(\theta)$ is easier to maximize than $p(x|\theta)$.

EM for latent variable models

EM Algorithm

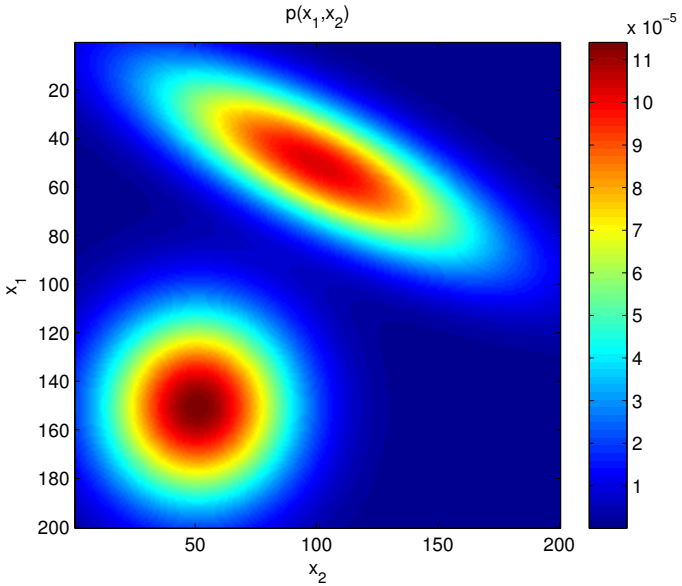
Initialize θ then, at each iteration;
E-Step: Compute the lower bound $\mathcal{Q}(\theta)$ using $q(h) = p(h|x, \theta)$
M-Step: $\theta^* = \arg\max_\theta \mathcal{Q}(\theta)$

EM for latent variable models

EM Algorithm

Initialize θ then, at each iteration;
E-Step: Compute the lower bound $\mathcal{Q}(\theta)$ using $q(h) = p(h|x, \theta)$
M-Step: $\theta^* = \arg\max_\theta \mathcal{Q}(\theta)$

Let us consider the following toy example:



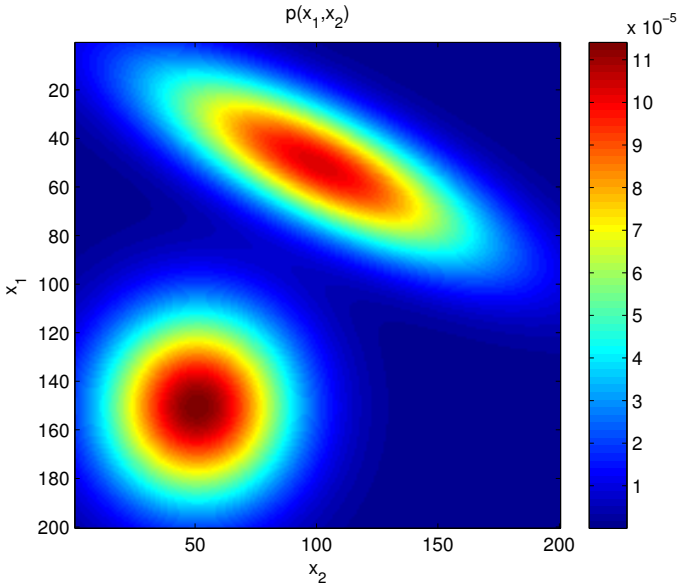
Joint distribution $p(x_1, x_2)$

EM for latent variable models

EM Algorithm

Initialize θ then, at each iteration;
E-Step: Compute the lower bound $\mathcal{Q}(\theta)$ using $q(h) = p(h|x, \theta)$
M-Step: $\theta^* = \arg\max_\theta \mathcal{Q}(\theta)$

Let us consider the following toy example:



Joint distribution $p(x_1, x_2)$

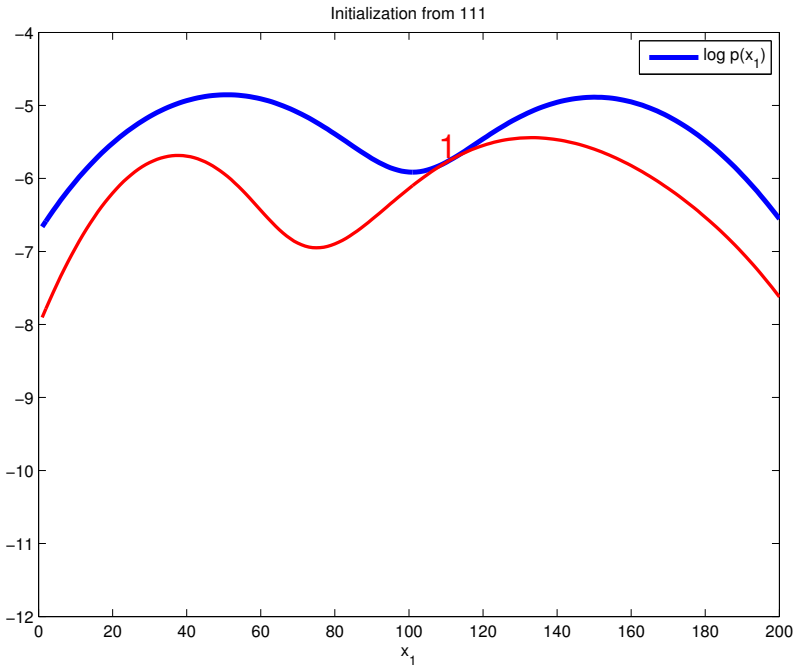
We take $h = x_2$, $\theta = x_1$. The problem is:

$$x_1^* = \arg\max_{x_1} \sum_{x_2} p(x_1, x_2)$$

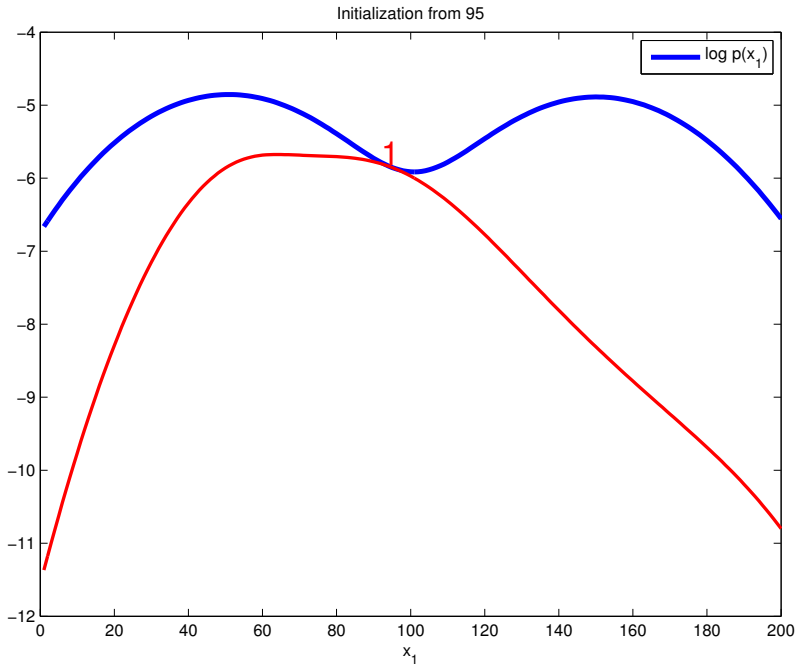
EM for latent variable models

EM Algorithm

Initialize θ then, at each iteration;
E-Step: Compute the lower bound $\mathcal{Q}(\theta)$ using $q(h) = p(h|x, \theta)$
M-Step: $\theta^* = \arg\max_\theta \mathcal{Q}(\theta)$



Initialization $x_1^1 = 111$

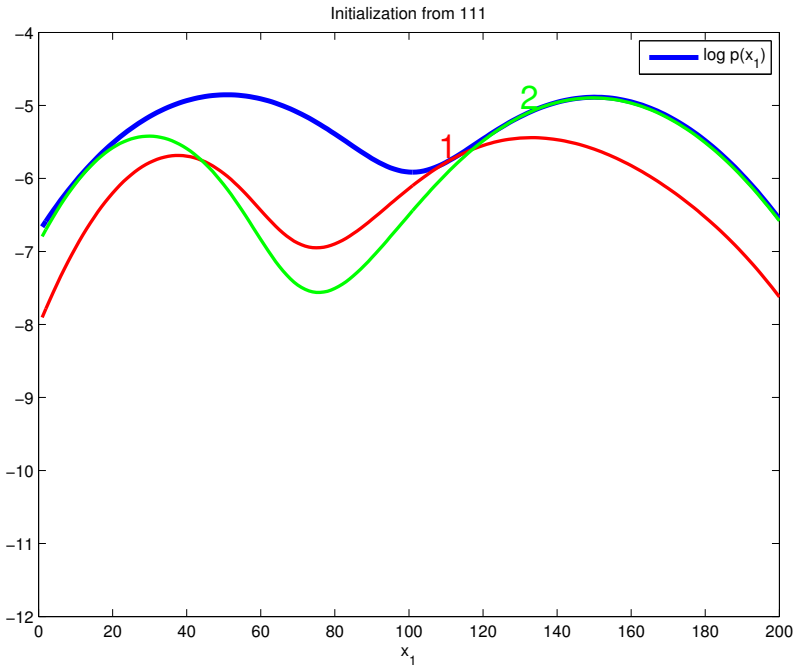


Initialization $x_1^1 = 95$

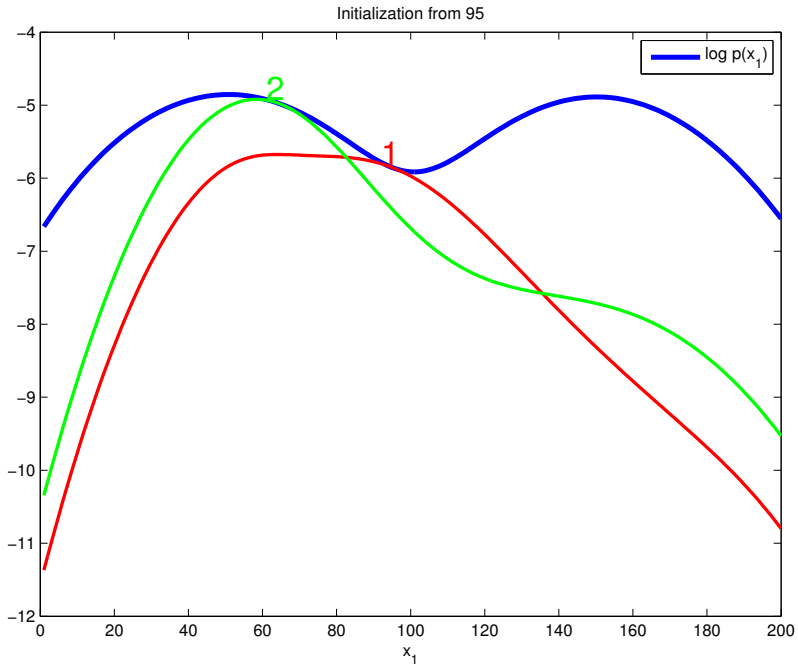
EM for latent variable models

EM Algorithm

Initialize θ then, at each iteration;
E-Step: Compute the lower bound $\mathcal{Q}(\theta)$ using $q(h) = p(h|x, \theta)$
M-Step: $\theta^* = \arg\max_\theta \mathcal{Q}(\theta)$



Initialization $x_1^1 = 111$

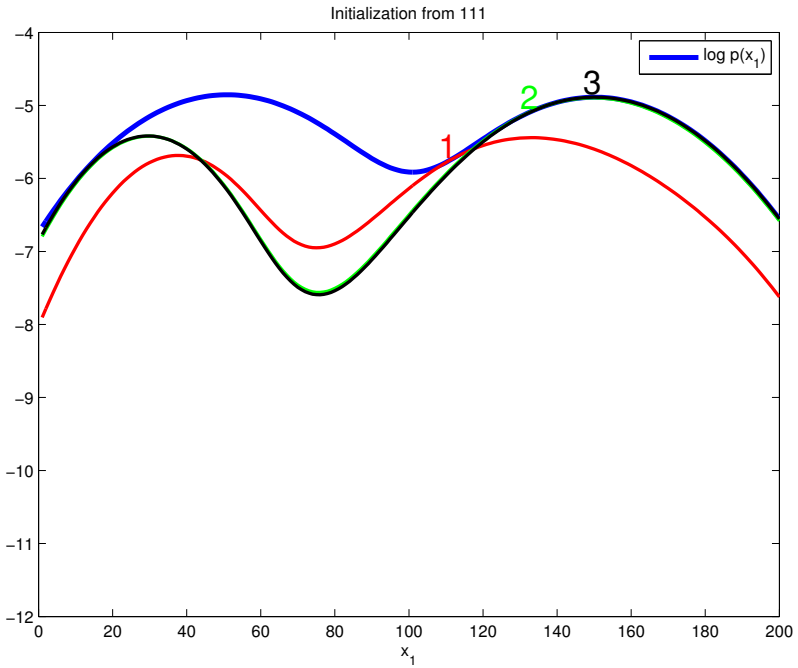


Initialization $x_1^1 = 95$

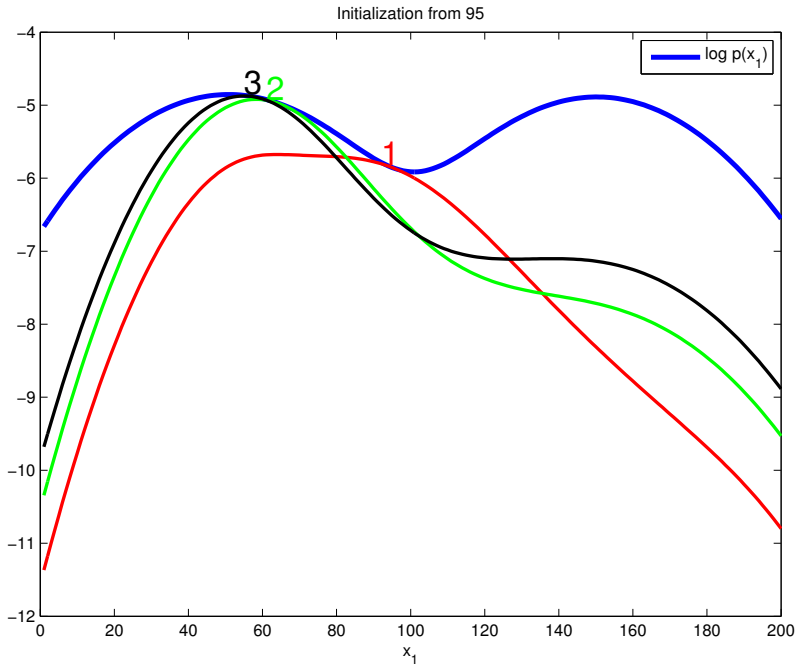
EM for latent variable models

EM Algorithm

Initialize θ then, at each iteration;
E-Step: Compute the lower bound $\mathcal{Q}(\theta)$ using $q(h) = p(h|x, \theta)$
M-Step: $\theta^* = \arg\max_\theta \mathcal{Q}(\theta)$



Initialization $x_1^1 = 111$

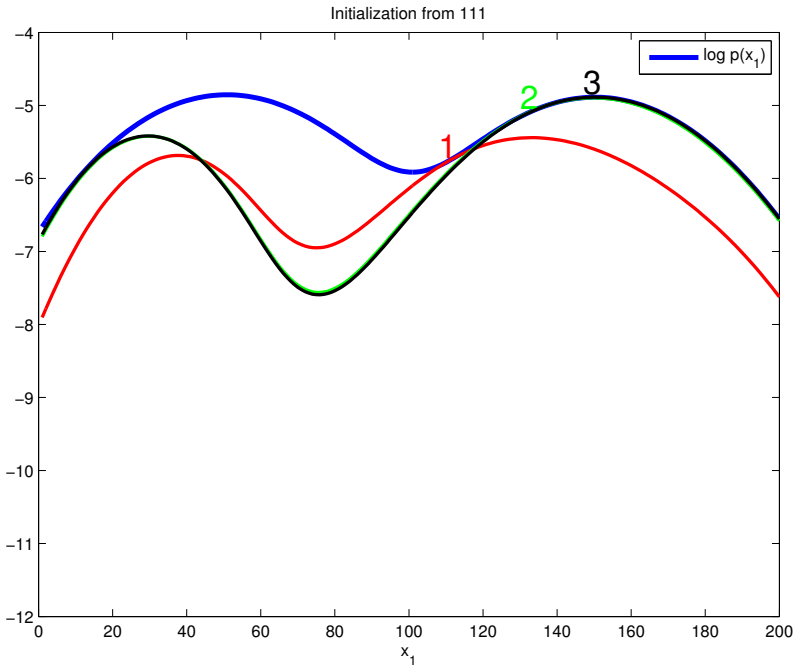


Initialization $x_1^1 = 95$

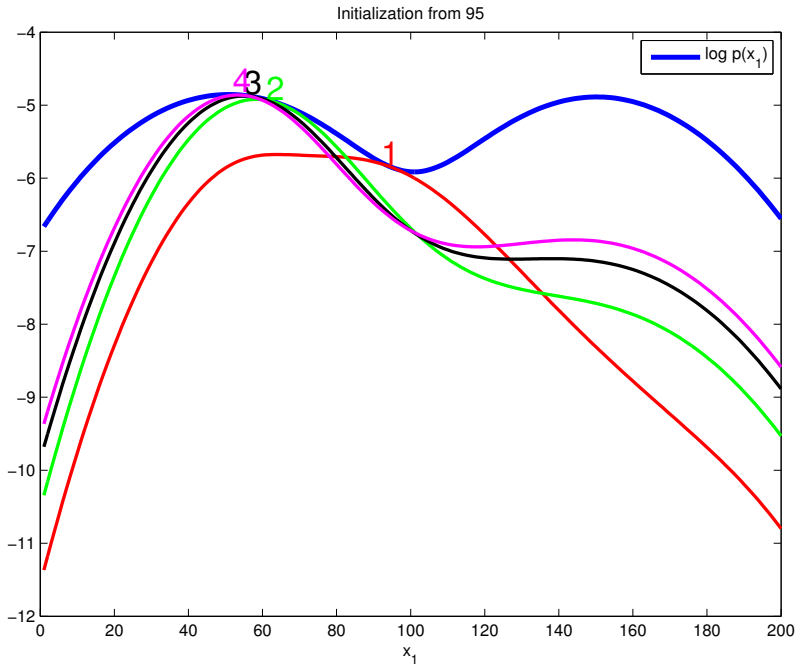
EM for latent variable models

EM Algorithm

Initialize θ then, at each iteration;
E-Step: Compute the lower bound $\mathcal{Q}(\theta)$ using $q(h) = p(h|x, \theta)$
M-Step: $\theta^* = \arg\max_\theta \mathcal{Q}(\theta)$



Initialization $x_1^1 = 111$

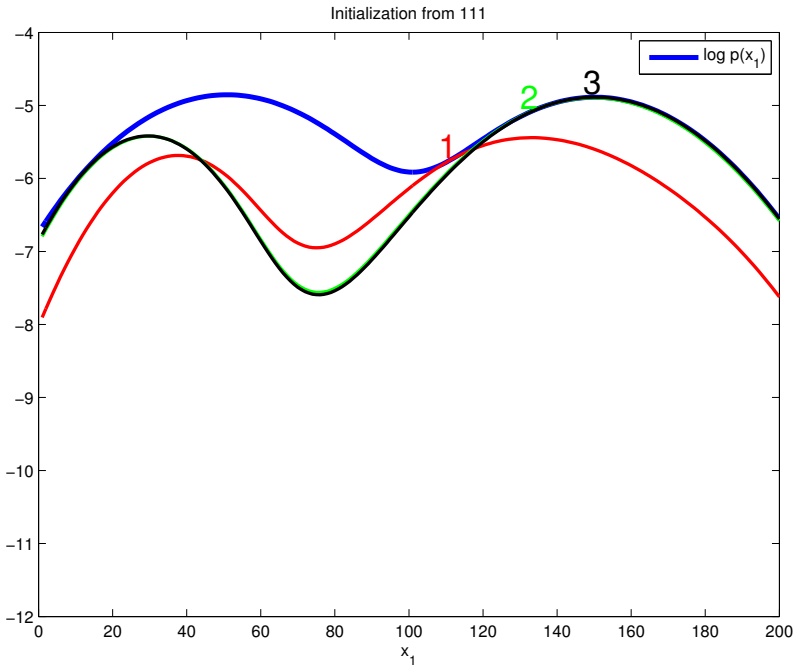


Initialization $x_1^1 = 95$

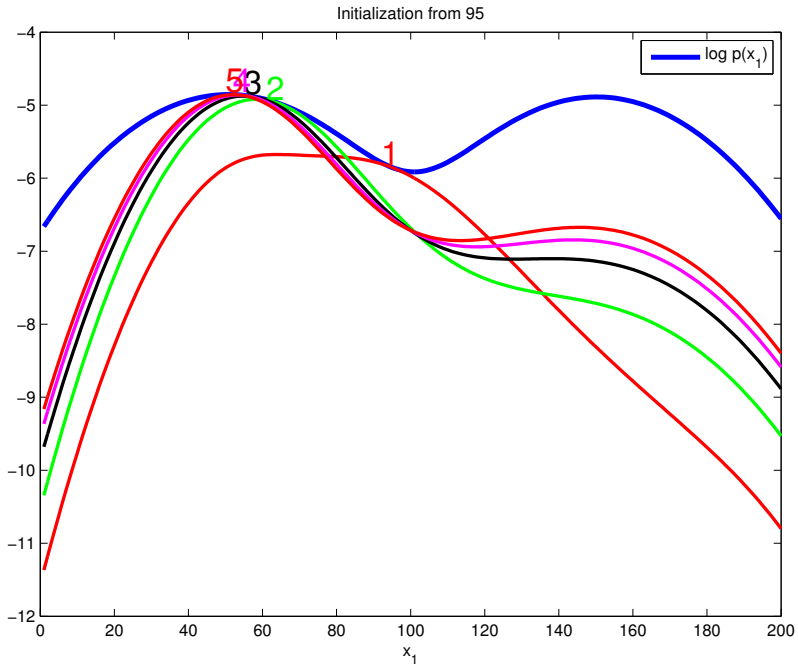
EM for latent variable models

EM Algorithm

Initialize θ then, at each iteration;
E-Step: Compute the lower bound $\mathcal{Q}(\theta)$ using $q(h) = p(h|x, \theta)$
M-Step: $\theta^* = \arg\max_\theta \mathcal{Q}(\theta)$



Initialization $x_1^1 = 111$

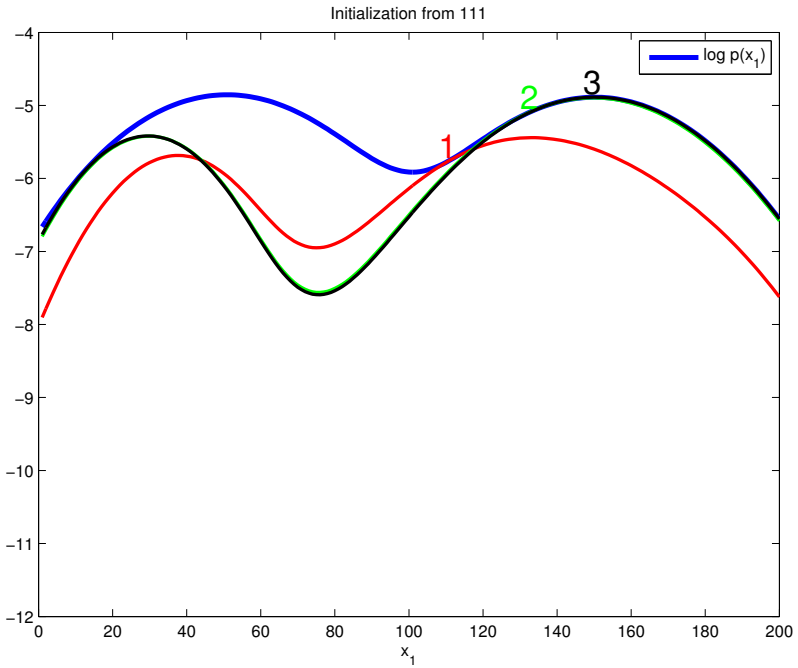


Initialization $x_1^1 = 95$

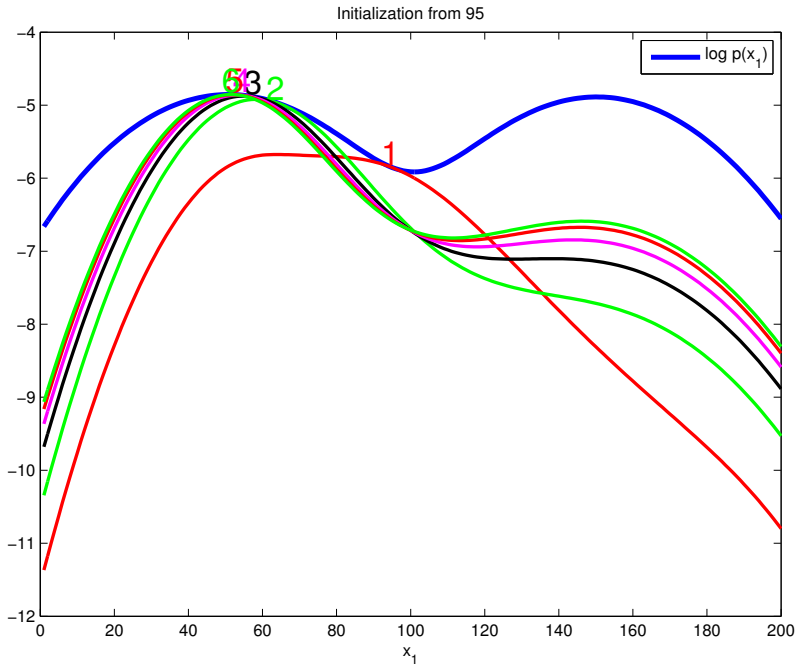
EM for latent variable models

EM Algorithm

Initialize θ then, at each iteration;
E-Step: Compute the lower bound $\mathcal{Q}(\theta)$ using $q(h) = p(h|x, \theta)$
M-Step: $\theta^* = \arg\max_\theta \mathcal{Q}(\theta)$



Initialization $x_1^1 = 111$



Initialization $x_1^1 = 95$

EM for latent variable models

- We see that EM is an iterative algorithm, sensitive to initialization.

EM for latent variable models

- We see that EM is an iterative algorithm, sensitive to initialization.
- In next section, we introduce an alternative local optima free, non-iterative learning method for LVMs, based on method of moments.

Outline

Method of Moments vs ML

- Suppose we have the i.i.d. data $x_{1:N}$, generated from $\mathcal{G}(x; a, b)$.

Method of Moments vs ML

- Suppose we have the i.i.d. data $x_{1:N}$, generated from $\mathcal{G}(x; a, b)$.
- A maximum likelihood estimator for a and b would require to solve,

$$\log(a) - \psi(a) = \log\left(\frac{1}{N}\sum_{n=1}^{N} x_n\right) - \frac{1}{N}\sum_{n=1}^{N}\log(x_n)$$

to find an estimate $\hat{a}$. Then,

$$\hat{b} = \frac{1}{\hat{a}N}\sum_{n=1}^{N} x_n$$

Method of Moments vs ML

- Suppose we have the i.i.d. data $x_{1:N}$, generated from $\mathcal{G}(x; a, b)$.
- A maximum likelihood estimator for a and b would require to solve,

$$\log(a) - \psi(a) = \log\left(\frac{1}{N}\sum_{n=1}^{N} x_n\right) - \frac{1}{N}\sum_{n=1}^{N}\log(x_n)$$

to find an estimate $\hat{a}$. Then,

$$\hat{b} = \frac{1}{\hat{a}N}\sum_{n=1}^{N} x_n$$

- So, a ML estimate for $\mathcal{G}(a, b)$ would require to use an iterative method.

Method of Moments vs ML

- Alternatively, we can use method of moments to estimate a and b:

Method of Moments vs ML

- Alternatively, we can use method of moments to estimate a and b:

$$M_1 := \mathbb{E}[x] = ab$$
$$M_2 := \mathbb{E}[x^2] = ab^2 + a^2 b^2$$

Then,

Method of Moments vs ML

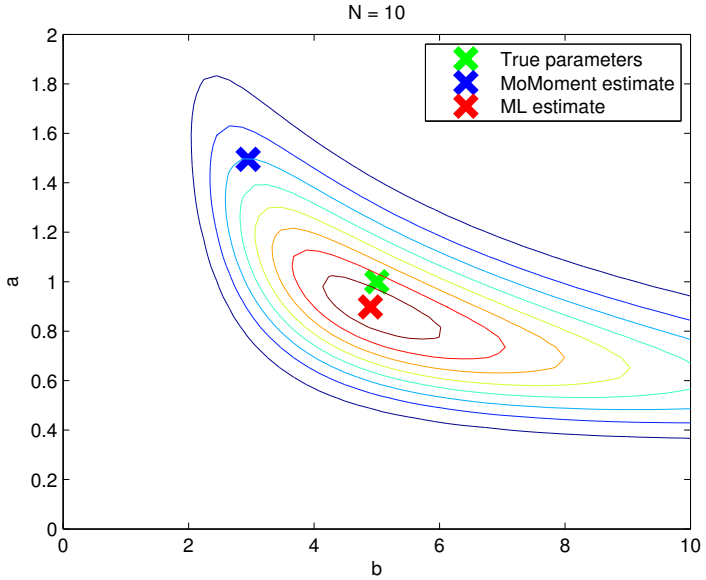
- Alternatively, we can use method of moments to estimate a and b:

$$M_1 := \mathbb{E}[x] = ab$$
$$M_2 := \mathbb{E}[x^2] = ab^2 + a^2 b^2$$

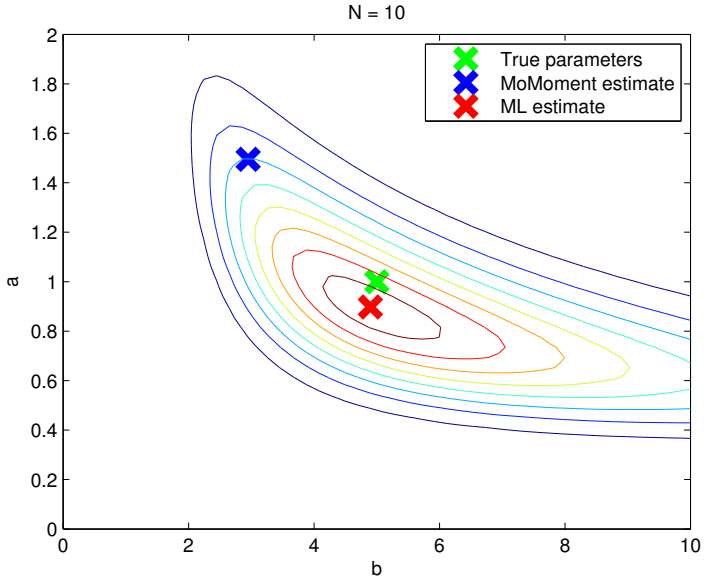
Then,

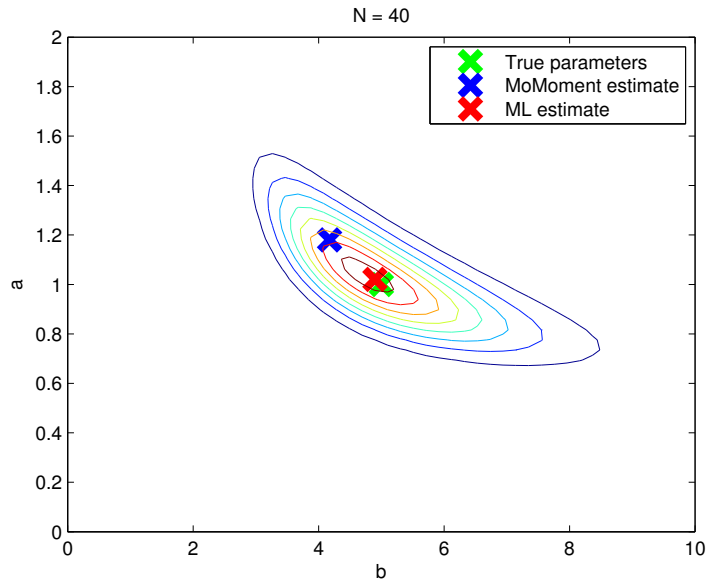
$$\hat{a} = \frac{M_1^2}{M_2 - M_1^2}$$
$$\hat{b} = \frac{M_2 - M_1^2}{M_1}$$

Method of Moments vs ML

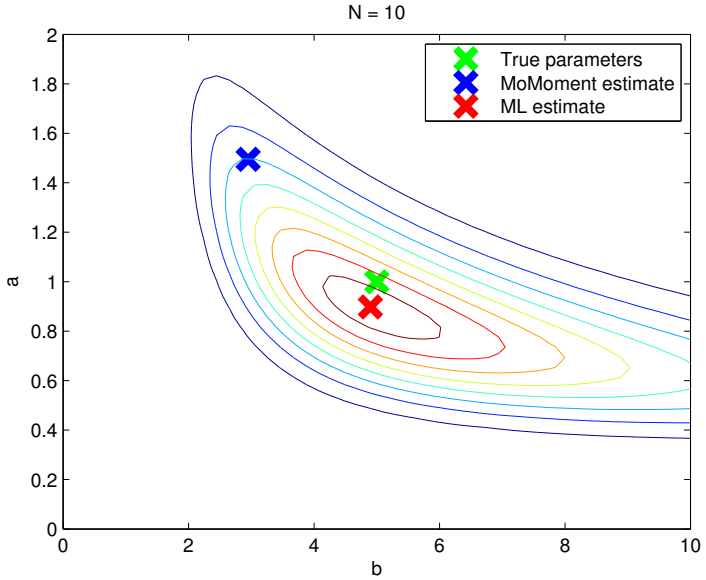


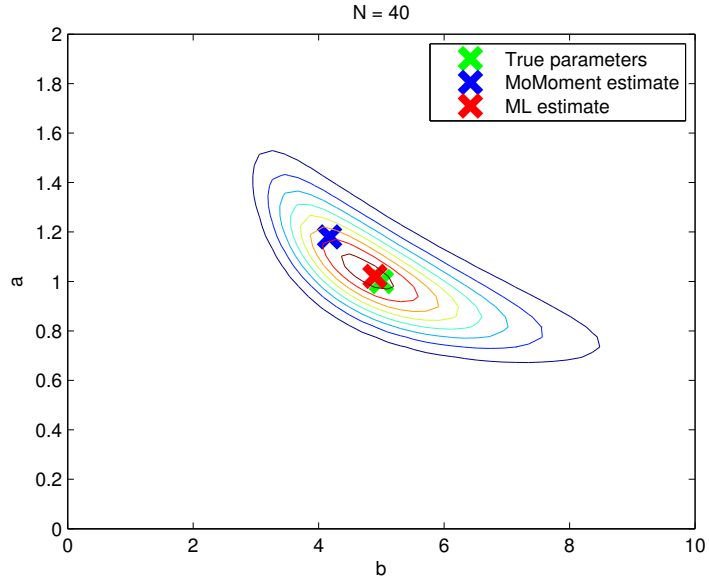
Method of Moments vs ML

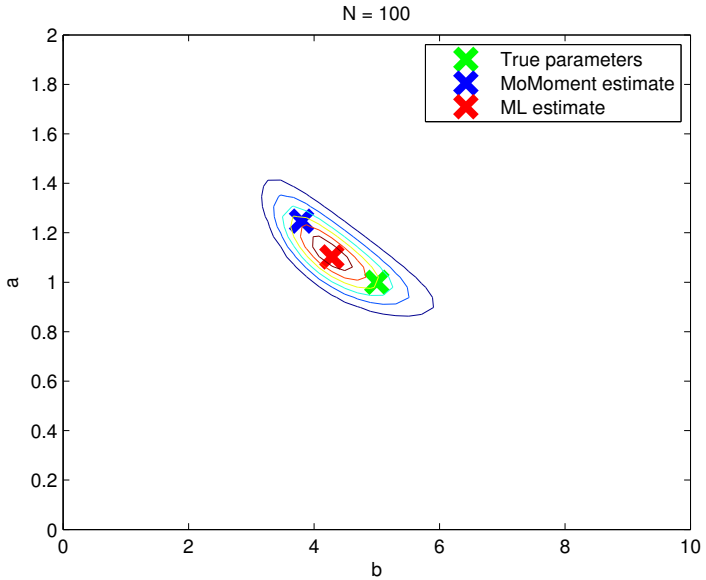




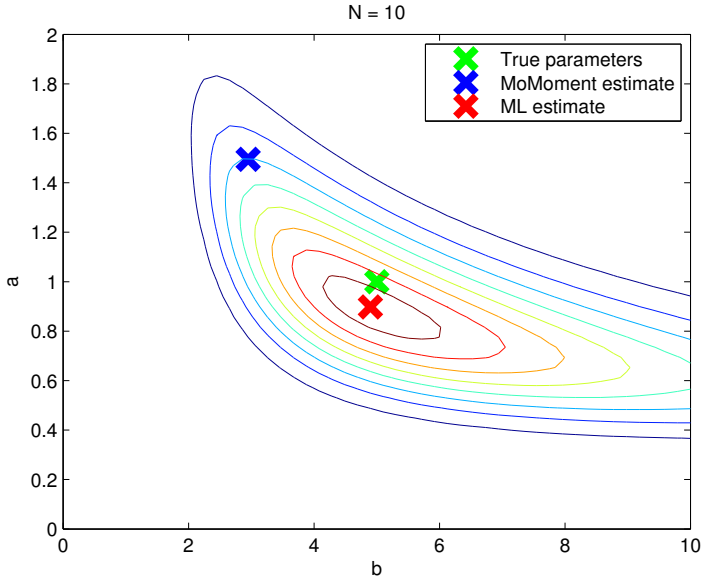
Method of Moments vs ML

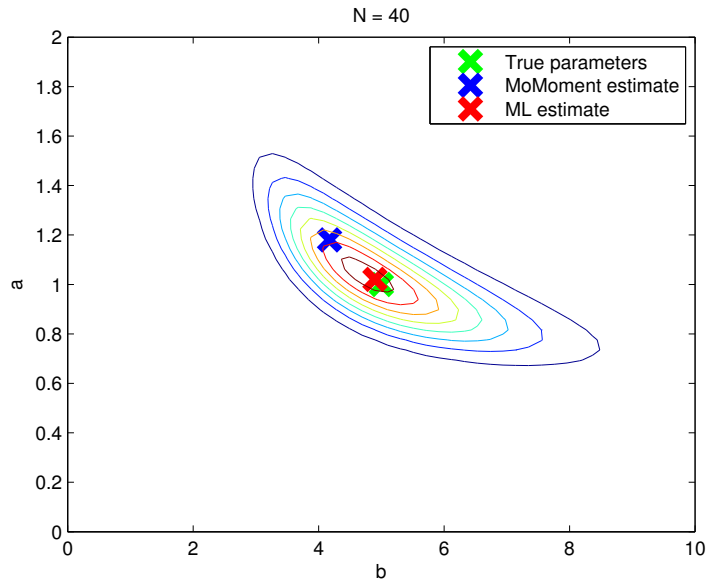


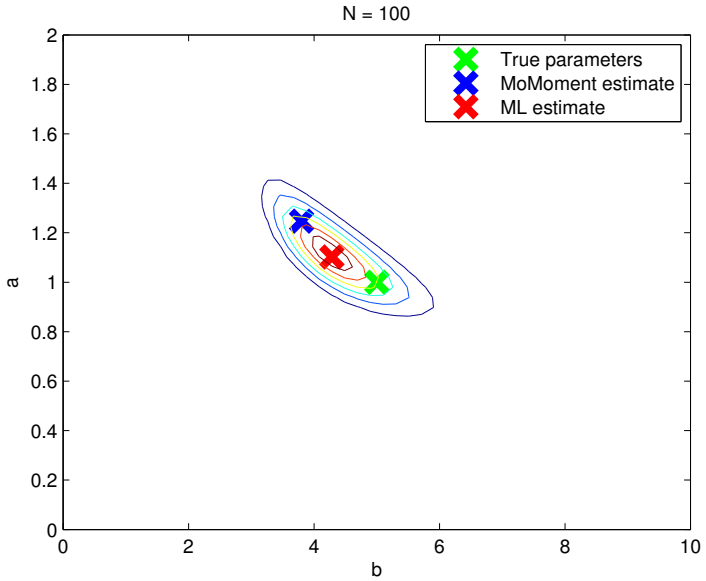


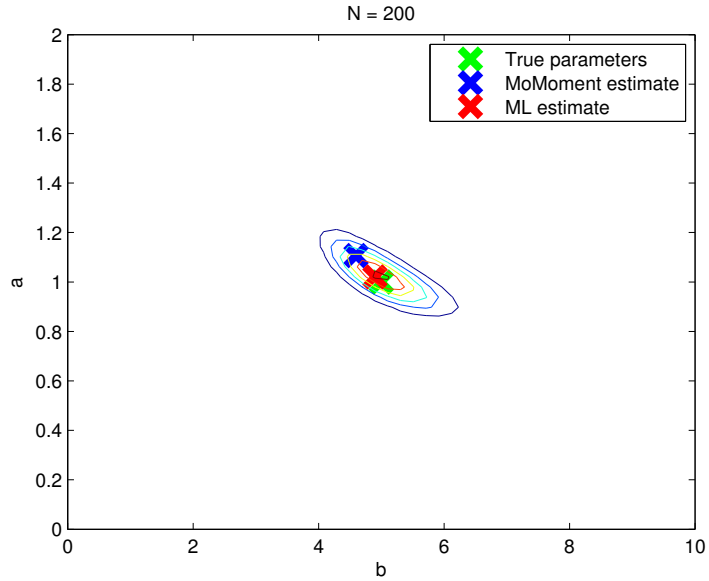


Method of Moments vs ML









Method of Moments vs ML

- Method of Moments yield simpler solutions than ML.

Method of Moments vs ML

- Method of Moments yield simpler solutions than ML.
- Method of Moments solutions are not iterative. ML solutions are.

Method of Moments vs ML

- Method of Moments yield simpler solutions than ML.
- Method of Moments solutions are not iterative. ML solutions are.
- ML solutions are more efficient, guaranteed to return parameter estimates in feasible range.

Method of Moments vs ML

- Method of Moments yield simpler solutions than ML.
- Method of Moments solutions are not iterative. ML solutions are.
- ML solutions are more efficient, guaranteed to return parameter estimates in feasible range.
- ML solutions may get stuck in local maxima, require good initializations. Method of moments do not.

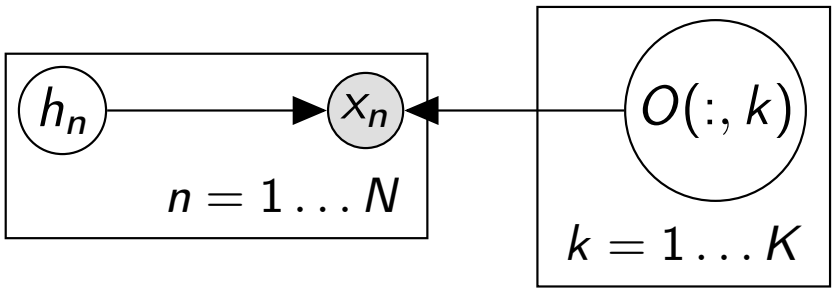
Spectral Learning for Latent Variable Models

- Spectral learning is an instance of method of moments.
- Model parameters are solved from a function of some observable moments.

Spectral Learning for Latent Variable Models

- Spectral learning is an instance of method of moments.
- Model parameters are solved from a function of some observable moments.

Let us consider a simple mixture model:



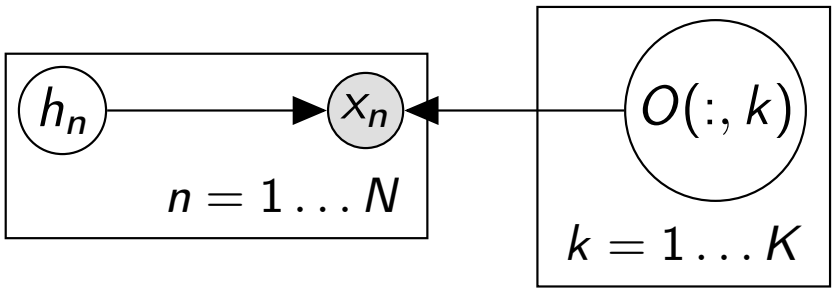
$$h_n \sim \text{Discrete}(p(h_n))$$
$$x_n|h_n \sim p(x_n|h_n, O(:,h_n))$$

where, $x \in \mathbb{R}^L$ observations,
$h_n \in \{1, \dots, K\}$ cluster indicators,
$\mathbb{E}[x|h] = O \in \mathbb{R}^{L \times K}$ model parameters,
e.g. for Poisson, $O(i,k) = \lambda_{i,k}$.

Spectral Learning for Latent Variable Models

- Spectral learning is an instance of method of moments.
- Model parameters are solved from a function of some observable moments.

Let us consider a simple mixture model:

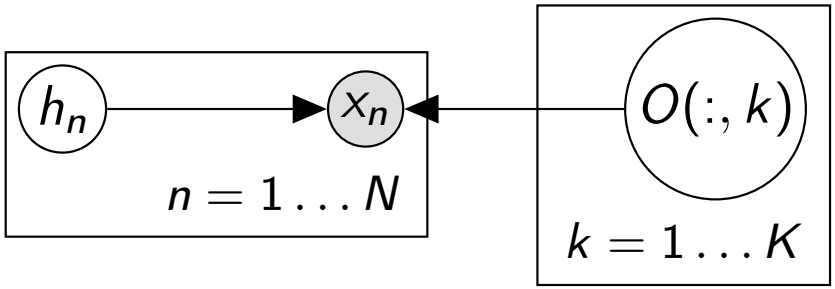


$$h_n \sim \text{Discrete}(p(h_n))$$
$$x_n|h_n \sim p(x_n|h_n, O(:,h_n))$$

where, $x \in \mathbb{R}^L$ observations,
$h_n \in \{1, \dots, K\}$ cluster indicators,
$\mathbb{E}[x|h] = O \in \mathbb{R}^{L \times K}$ model parameters,
e.g. for Poisson, $O(i,k) = \lambda_{i,k}$.

- Given $x_{1:N}$, one can use EM to learn O. Alternatively, we can use spectral learning to learn O, in a non iterative fashion.

Spectral Learning for Latent Variable Models



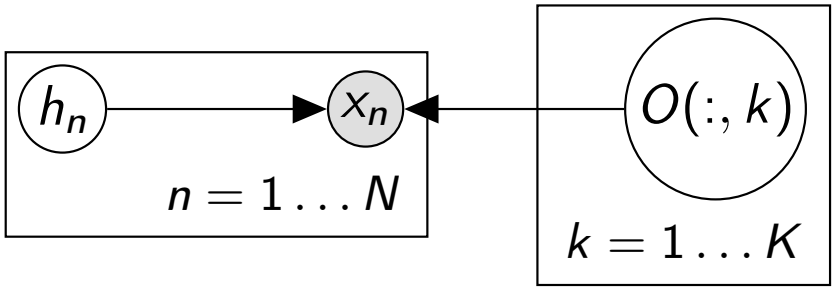
$$h_n \sim \text{Discrete}(p(h_n))$$
$$x_n|h_n \sim p(x_n|h_n, O(:,h_n))$$

Given that $L \geq K$, and O has K linearly independent columns then,

$$\begin{aligned} B_i :=& (U^T \mathbb{E}[x \otimes x \otimes x_i] V)(U^T \mathbb{E}[x \otimes x] V)^{-1} \\ =& (U^T O)\text{diag}(O(i,:))(U^T O)^{-1} \end{aligned}$$

where, $\mathbb{E}[x \otimes x] = U \Sigma V^T$.

Spectral Learning for Latent Variable Models



$$h_n \sim \text{Discrete}(p(h_n))$$
$$x_n|h_n \sim p(x_n|h_n, O(:,h_n))$$

Given that $L \geq K$, and O has K linearly independent columns then,

$$\begin{aligned} B_i :=& (U^T \mathbb{E}[x \otimes x \otimes x_i] V)(U^T \mathbb{E}[x \otimes x] V)^{-1} \\ =& (U^T O)\text{diag}(O(i,:))(U^T O)^{-1} \end{aligned}$$

where, $\mathbb{E}[x \otimes x] = U \Sigma V^T$.

- B_i can be expressed in terms of observable moments $\mathbb{E}[x \otimes x]$ and $\mathbb{E}[x \otimes x \otimes x_i]$.
- So, model parameters can be simply be read from the eigenvectors of B_i.

Spectral Learning for Latent Variable Models

Proof:

$$U^T \mathbb{E}[x \otimes x \otimes x_i] V = U^T O \text{diag}(O(i,:)) \text{diag}(p(h)) O^T V$$

Spectral Learning for Latent Variable Models

Proof:

$$\begin{aligned} U^T \mathbb{E}[x \otimes x \otimes x_i] V &= U^T O \text{diag}(O(i,:)) \text{diag}(p(h)) O^T V \\ &= U^T O \text{diag}(O(i,:)) (U^T O)^{-1} U^T O \text{diag}(p(h)) O^T V \end{aligned}$$

Spectral Learning for Latent Variable Models

Proof:

$$
\begin{aligned}
U^T \mathbb{E}[x \otimes x \otimes x_i] V &= U^T O \text{diag}(O(i,:)) \text{diag}(p(h)) O^T V \\
&= U^T O \text{diag}(O(i,:)) (U^T O)^{-1} U^T O \text{diag}(p(h)) O^T V \\
&= U^T O \text{diag}(O(i,:)) (U^T O)^{-1} \underbrace{U^T O \text{diag}(p(h)) O^T V}_{U^T \mathbb{E}[x \otimes x] V}
\end{aligned}
$$

Spectral Learning for Latent Variable Models

Proof:

$$\begin{aligned} U^T \mathbb{E}[x \otimes x \otimes x_i] V &= U^T O \text{diag}(O(i,:)) \text{diag}(p(h)) O^T V \\ &= U^T O \text{diag}(O(i,:)) (U^T O)^{-1} U^T O \text{diag}(p(h)) O^T V \\ &= U^T O \text{diag}(O(i,:)) (U^T O)^{-1} \underbrace{U^T O \text{diag}(p(h)) O^T V}_{U^T \mathbb{E}[x \otimes x] V} \end{aligned}$$

so,

$$(U^T O) \text{diag}(O(i,:)) (U^T O)^{-1} = (U^T \mathbb{E}[x \otimes x \otimes x_i] V)(U^T \mathbb{E}[x \otimes x] V)^{-1}$$

□

Spectral Learning for Latent Variable Models

- This algorithm is based on the work of Anandkumar et al. 2012.

Spectral Learning for Latent Variable Models

- This algorithm is based on the work of Anandkumar et al. 2012.
- One can also modify this algorithm slightly to learn HMMs.

Spectral Learning for Latent Variable Models

- This algorithm is based on the work of Anandkumar et al. 2012.
- One can also modify this algorithm slightly to learn HMMs.
- The general goal is to express the observable moments so that, we obtain an eigen-decomposition form, from which we can read the parameters.

Spectral Learning for Latent Variable Models

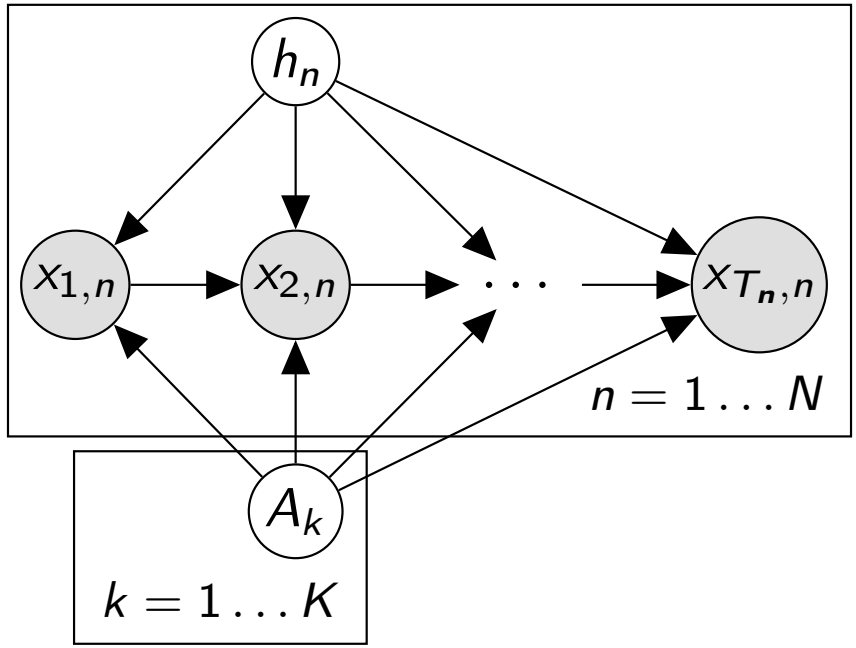
- This algorithm is based on the work of Anandkumar et al. 2012.
- One can also modify this algorithm slightly to learn HMMs.
- The general goal is to express the observable moments so that, we obtain an eigen-decomposition form, from which we can read the parameters.
- This is a fast, and local optima free algorithm to learn LVMs.

Spectral Learning for Latent Variable Models

- This algorithm is based on the work of Anandkumar et al. 2012.
- One can also modify this algorithm slightly to learn HMMs.
- The general goal is to express the observable moments so that, we obtain an eigen-decomposition form, from which we can read the parameters.
- This is a fast, and local optima free algorithm to learn LVMs.
- However, this scheme requires high order moments to learn a temporally connected LVM such as mixture of Markov models.

Outline

Mixture of Markov models



$$h_n \sim \text{Discrete}(p(h_n))$$

$$x_{t,n} | \; x_{t-1,n}, h_n, A \sim \text{Discrete}(A(:, x_{t-1,n}, h_n))$$

Spectral Learning for Mixture of Markov models

- We show that a spectral learning algorithm as in Anandkumar, et al. 2012, requires observable moments up to order five.

Spectral Learning for Mixture of Markov models

- We show that a spectral learning algorithm as in Anandkumar, et al. 2012, requires observable moments up to order five.
- We propose an alternative scheme based on hierarchical Bayesian modeling:

$$\begin{aligned} p(A_{h_n}|\mathbf{x}_n, h_n) \propto & \quad p(\mathbf{x}_n|A_{h_n}, h_n)p(A_{h_n}) \\ = & \quad \text{Dirichlet}(c^n_{1,1} + \beta - 1, c^n_{1,2} + \beta - 1, \ldots, c^n_{L,L} + \beta - 1) \end{aligned}$$

Spectral Learning for Mixture of Markov models

- We show that a spectral learning algorithm as in Anandkumar, et al. 2012, requires observable moments up to order five.
- We propose an alternative scheme based on hierarchical Bayesian modeling:

$$\begin{aligned} p(A_{h_n}|\mathbf{x}_n, h_n) \propto & \quad p(\mathbf{x}_n|A_{h_n}, h_n)p(A_{h_n}) \\ = & \quad \text{Dirichlet}(c^n_{1,1} + \beta - 1, c^n_{1,2} + \beta - 1, \ldots, c^n_{L,L} + \beta - 1) \end{aligned}$$

- Assuming a uniform Dirichlet prior $p(A_{h_n})$, i.e. taking $\beta = 1$, the posterior of A_{h_n} becomes $\text{Dirichlet}(c^n_{1,1}, c^n_{1,2}, \ldots, c^n_{L,L})$.

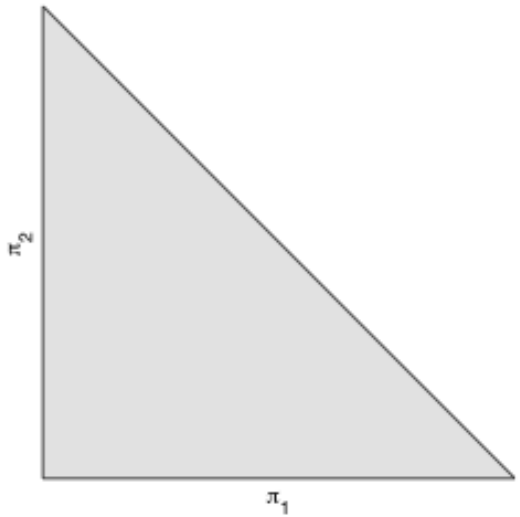
Spectral Learning for Mixture of Markov models

- We show that a spectral learning algorithm as in Anandkumar, et al. 2012, requires observable moments up to order five.
- We propose an alternative scheme based on hierarchical Bayesian modeling:

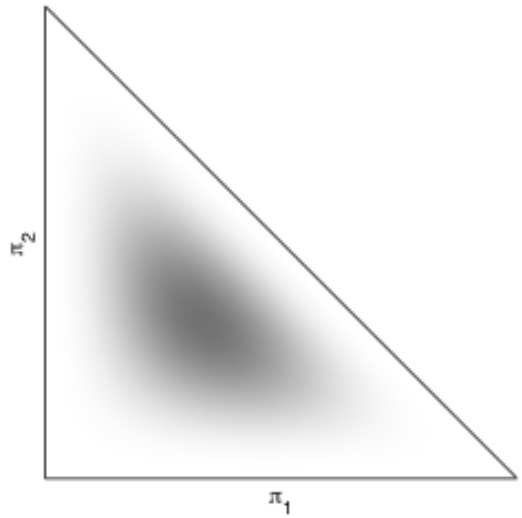
$$\begin{aligned} p(A_{h_n}|\mathbf{x}_n, h_n) \propto & \quad p(\mathbf{x}_n|A_{h_n}, h_n)p(A_{h_n}) \\ = & \quad \text{Dirichlet}(c^n_{1,1} + \beta - 1, c^n_{1,2} + \beta - 1, \ldots, c^n_{L,L} + \beta - 1) \end{aligned}$$

- Assuming a uniform Dirichlet prior $p(A_{h_n})$, i.e. taking $\beta = 1$, the posterior of A_{h_n} becomes $\text{Dirichlet}(c^n_{1,1}, c^n_{1,2}, \ldots, c^n_{L,L})$.
- So, we can represent a sequence only via empirical transition counts matrix $s_n \sim \text{Dirichlet}(c^n_{1,1}, c^n_{1,2}, \ldots, c^n_{L,L}) = p(A_{h_n}|\mathbf{x}_n, h_n)$.

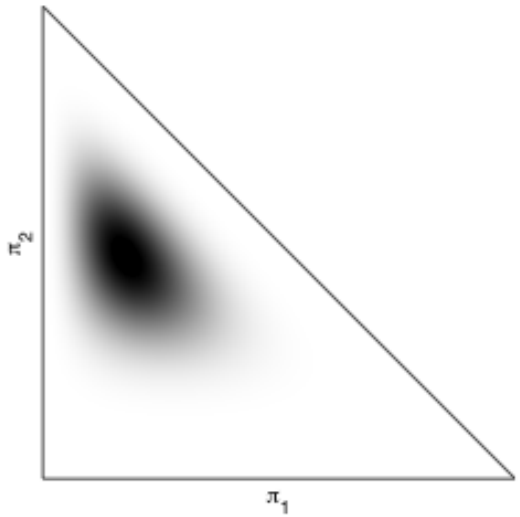
Spectral Learning for Mixture of Markov models - Dirichlet distribution



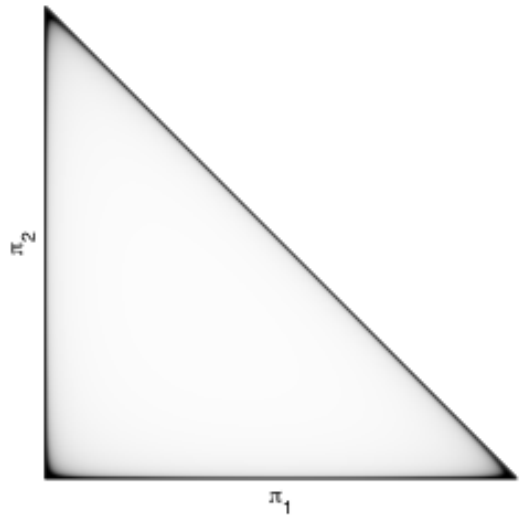
$\pi \sim \mathrm{Dir}(1,1,1)$



$\pi \sim \mathrm{Dir}(4,4,4)$

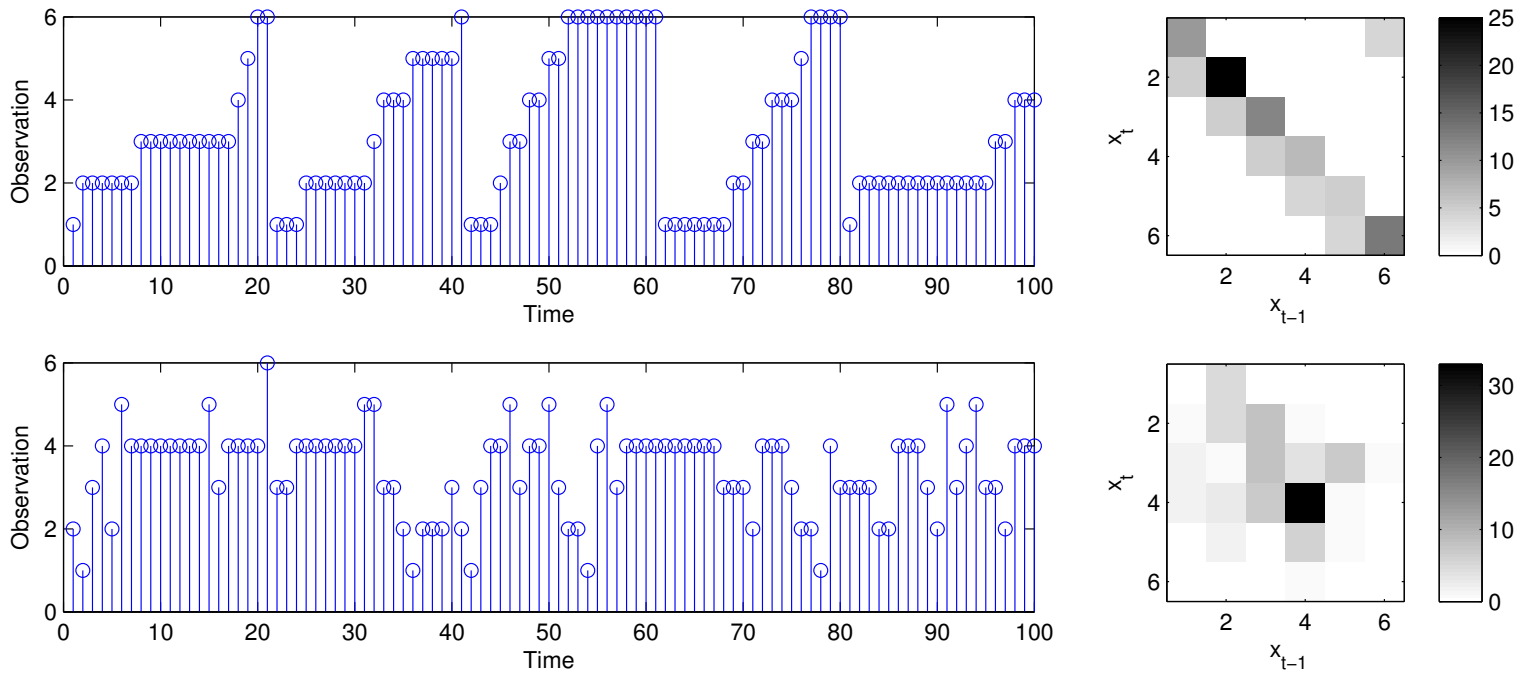


$\pi \sim \mathrm{Dir}(4,9,7)$

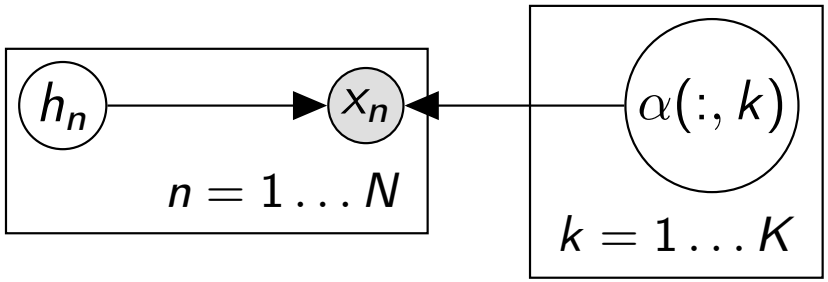


$\pi \sim \mathrm{Dir}(0.2,0.2,0.2)$

Spectral Learning for Mixture of Markov models - Dirichlet posterior

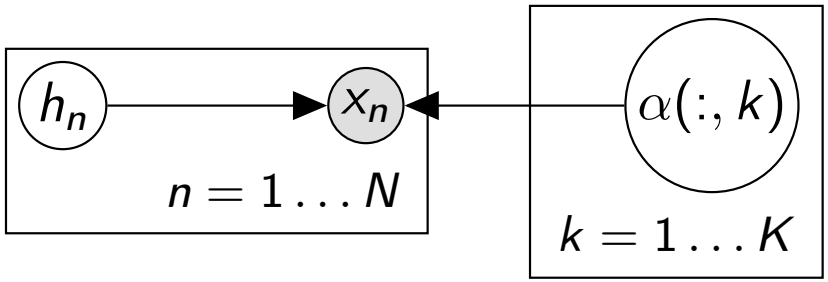


Spectral Learning for Mixture of Dirichlet Distributions



$$h_n \sim \text{Discrete}(p(h_n))$$
$$x_n | h_n, \alpha \sim \text{Dirichlet}(\alpha(1, h_n), \ldots, \alpha(K, h_n))$$

Spectral Learning for Mixture of Dirichlet Distributions



$$h_n \sim \text{Discrete}(p(h_n))$$
$$x_n | h_n, \alpha \sim \text{Dirichlet}(\alpha(1, h_n), \ldots, \alpha(K, h_n))$$

- Note that $\mathbb{E}[s_{n,i} | h_n = k] = \alpha(i,k)/\alpha_0$.
- So, we can estimate α, up to a scaling factor α_0 via spectral learning.

Spectral Learning for Mixture of Markov models

Input: Sequences $\mathbf{x}_{1:N}$
Output: Clustering assignments $\widehat{h}_{1:N}$

Spectral Learning for Mixture of Markov models

Input: Sequences $\mathbf{x}_{1:N}$
Output: Clustering assignments $\widehat{h}_{1:N}$
1. Extract the sufficient statistics s_n from $x_n, \forall n \in \{1, \ldots, N\}$

Spectral Learning for Mixture of Markov models

Input: Sequences $\mathbf{x}_{1:N}$
Output: Clustering assignments $\widehat{h}_{1:N}$
1. Extract the sufficient statistics s_n from $x_n, \forall n \in \{1, \ldots, N\}$
2. Compute empirical moment estimates for $\mathbb{E}[s \otimes s]$ and $\mathbb{E}[s \otimes s \otimes s]$.

Compute U and V such that $\mathbb{E}[s \otimes s] = U \Sigma V^T$.

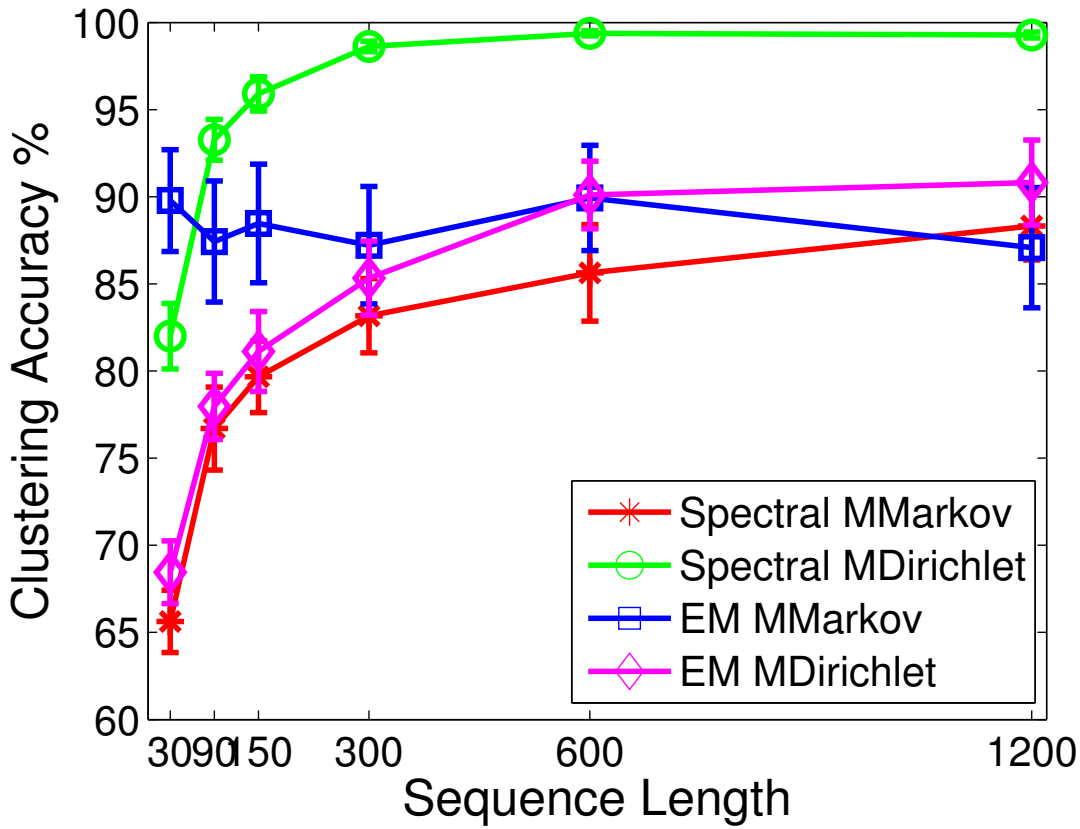
Spectral Learning for Mixture of Markov models

Input: Sequences $\mathbf{x}_{1:N}$
Output: Clustering assignments $\widehat{h}_{1:N}$
1. Extract the sufficient statistics s_n from $x_n, \forall n \in \{1, \ldots, N\}$
2. Compute empirical moment estimates for $\mathbb{E}[s \otimes s]$ and $\mathbb{E}[s \otimes s \otimes s]$. Compute U and V such that $\mathbb{E}[s \otimes s] = U \Sigma V^T$.
3. Estimate α by doing eigen-decomposition of B_i.

Spectral Learning for Mixture of Markov models

Input: Sequences $\mathbf{x}_{1:N}$
Output: Clustering assignments $\widehat{h}_{1:N}$
1. Extract the sufficient statistics s_n from $x_n, \forall n \in \{1, \ldots, N\}$
2. Compute empirical moment estimates for $\mathbb{E}[s \otimes s]$ and $\mathbb{E}[s \otimes s \otimes s]$. Compute U and V such that $\mathbb{E}[s \otimes s] = U \Sigma V^T$.
3. Estimate α by doing eigen-decomposition of B_i.
4. $\forall n \in \{1, \ldots N\}, \widehat{h}_n = \arg\max_k \text{Dirichlet}(s_n, \alpha(:, k))$.

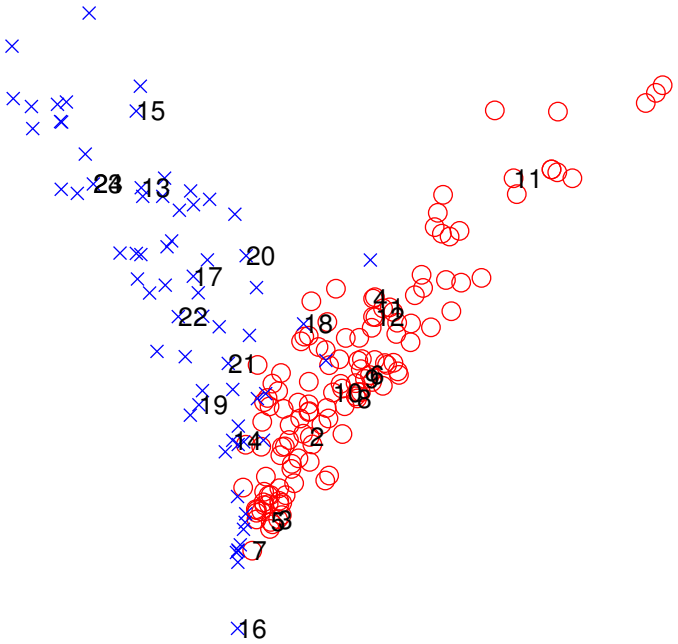
Results - Synthetic Data

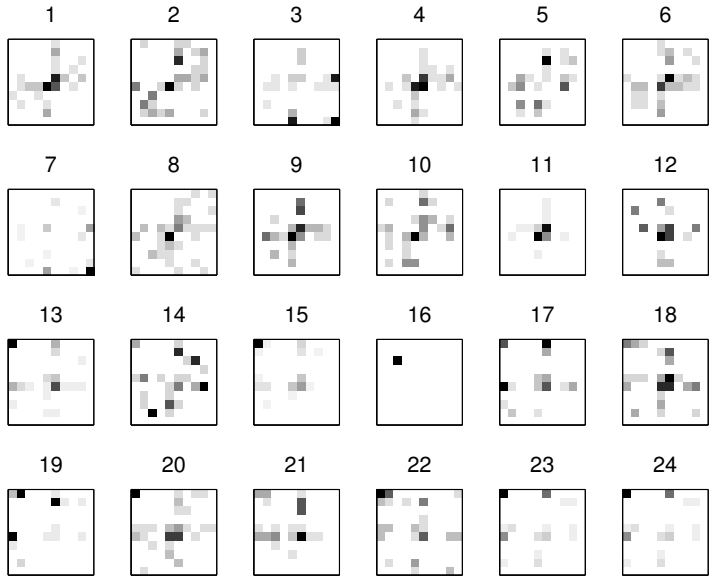


- On 100 synthetic datasets, each consisting of 60 sequences, average clustering accuracies for varying sequence lengths.

Results - Network Traffic Data

- Exploratory data analysis on network traffic data consisting of Skype and Bit-torrent





Results - Network Traffic Data

- Effect of clustering in training

Algorithm	Classification Accuracy
No Clustering	63.41%
Mixture of Dirichlet (Spectral)	84.85%
Mixture of Dirichlet (EM)	79.39%
Mixture of Markov (EM)	83.51%

Results - Finding number of Clusters

- We can also estimate the number of clusters:

$$\mathbb{E}[s_n \otimes s_n] = \alpha \text{diag}(p(h))\alpha^T = \sum_{k=1}^{K} p(h=k)\alpha(:,k)\alpha(:,k)^T$$

- By looking at the jump σ_{k+1}/σ_k of $\mathbb{E}[s_n \otimes s_n]$, we can estimate K.

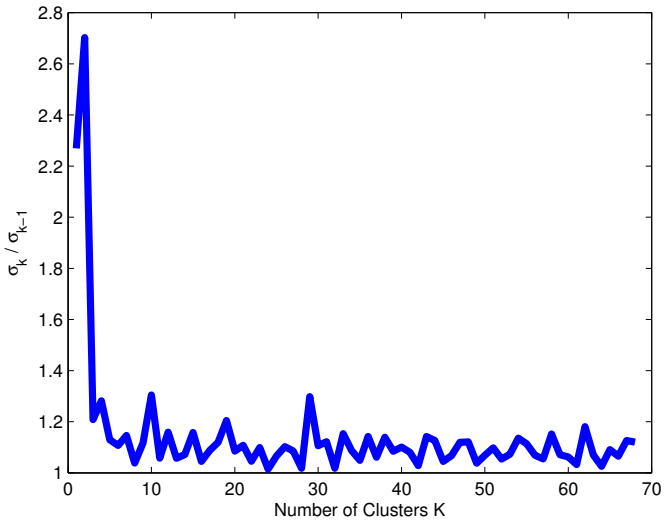


Figure: Network Traffic Data.

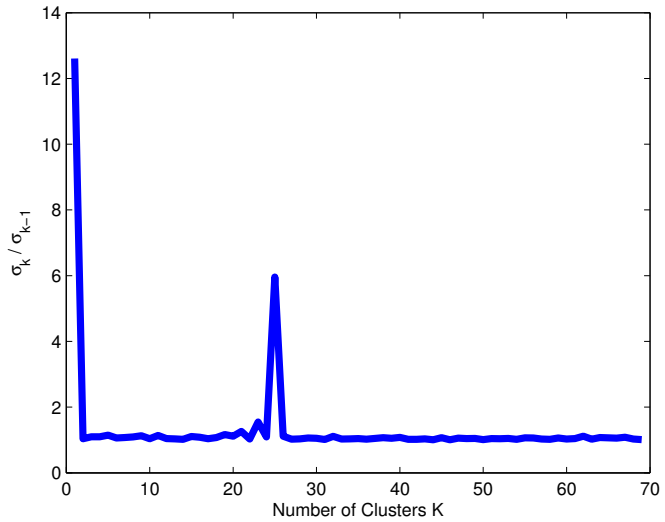
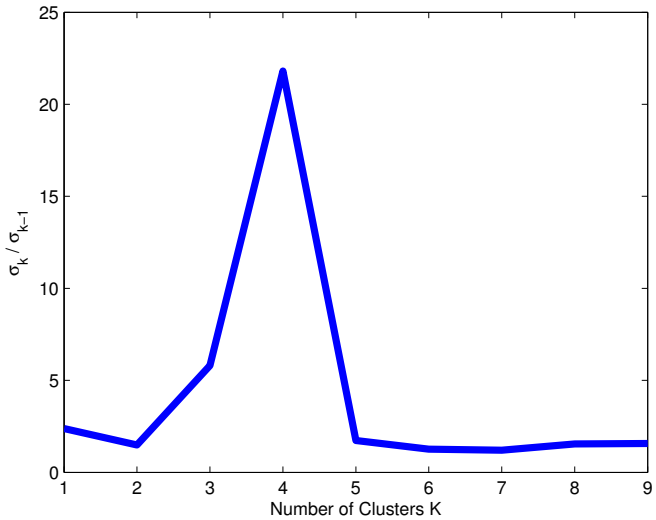
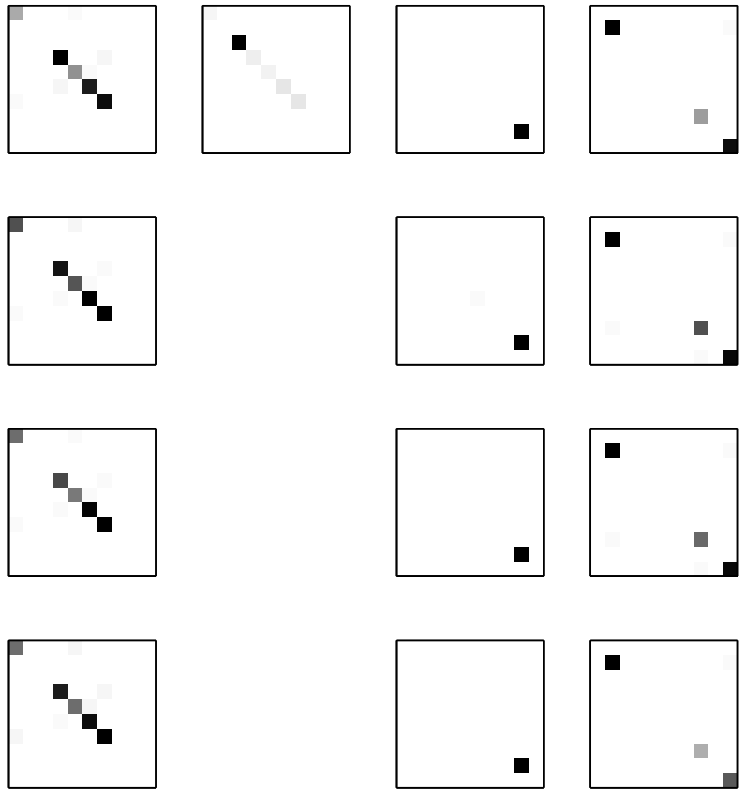


Figure: Synthetic Data.

Results - Finding number of Clusters

- Finding number of clusters on motion capture data.



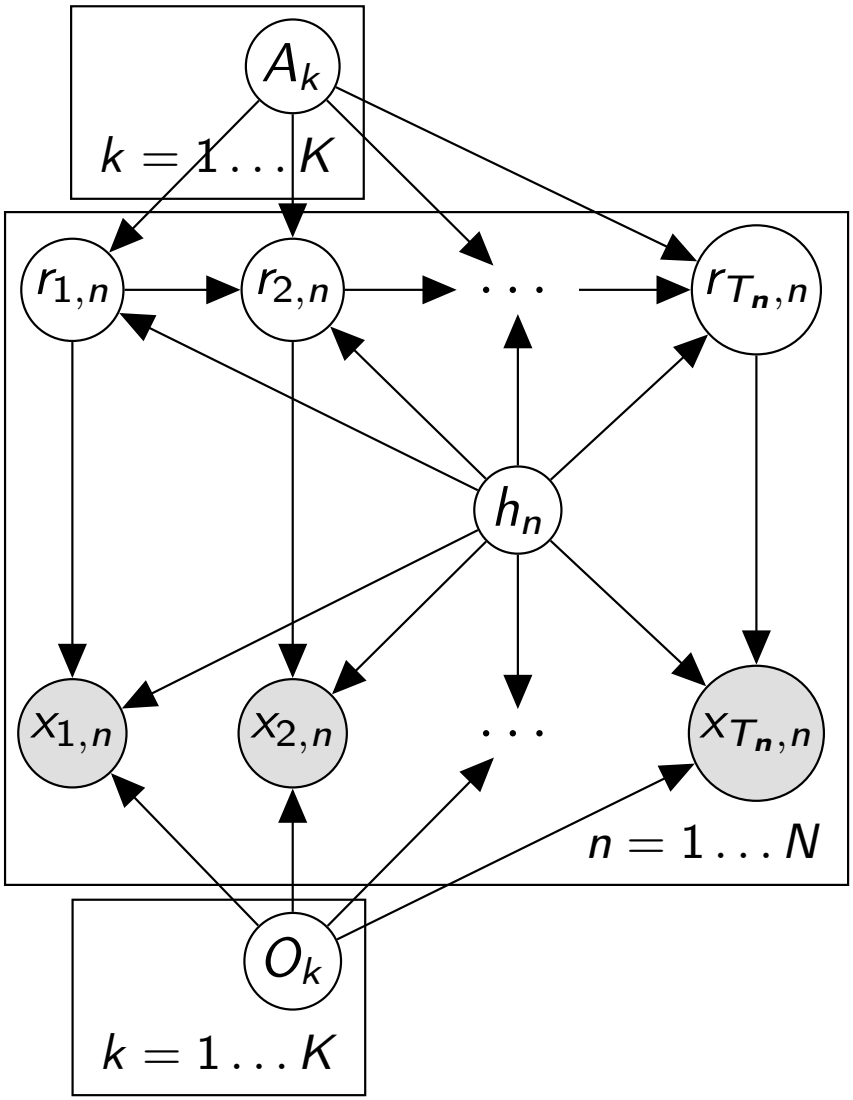


Conclusions - Spectral Learning for Mixture of Markov models

- Spectral learning of mixture of Dirichlet disributions yield a simple and effective algorithm.
- On synthetic data, it outperforms its rivals.
- It is also possible to automatically determine the number clusters.

Outline

Mixture of HMMs



$$
\begin{aligned}
h_n &\sim \text{Discrete}(p(h_n)) \\
r_{t,n}|r_{t-1,n}, h_n &\sim \text{Discrete}(A(:, r_{t-1,n}, h_n)) \\
x_{t,n}|r_{t,n}, h_n &\sim p(x_{t,n}|\ O(:, r_{t,n}, h_n))
\end{aligned}
$$

Spectral Learning based algorithm for Mixture of HMMs

- An EM algorithm for mixture of HMMs is expensive. Requires forward-backward message passing for each sequence in each EM iteration.
- We propose to replace the parameter estimation step with spectral learning.

Cheaper Alternative Algorithm for HMM Mixture learning

Randomly initialize $r_{1:N}$
At each iteration,
for $k = 1 \rightarrow K$ **do**
  $\theta_k \leftarrow \text{EMHMM}(\{x_n | \forall n, h_n = k\})$
end for
for $n = 1 \rightarrow N$ **do**
  $h_n = \arg\max_k p(x_n|\theta_k)$
end for

Cheaper Alternative Algorithm for HMM Mixture learning

Randomly initialize $r_{1:N}$
At each iteration,
for $k = 1 \rightarrow K$ **do**
  $\theta_k \leftarrow \text{SpectralHMM}(\{x_n | \forall n, h_n = k\})$
end for
for $n = 1 \rightarrow N$ **do**
  $h_n = \arg\max_k p(x_n|\theta_k)$
end for

Cheaper Alternative Algorithm for HMM Mixture learning

Randomly initialize $r_{1:N}$
At each iteration,
for $k = 1 \rightarrow K$ **do**
  $\theta_k \leftarrow \text{SpectralHMM}(\{x_n | \forall n, h_n = k\})$
end for
for $n = 1 \rightarrow N$ **do**
  $h_n = \arg\max_k p(x_n|\theta_k)$
end for

With spectral learning, required computations for parameter estimation step become K low-rank SVDs and K eigen-decompositions, which is substantially cheaper compared to $N \times K$ forward-backward.

Results on mocap data

Results obtained on Motion capture data. We input the temporal difference of 3D coordinates of 32 markers on human body. In this case, we used 20 sequences. We restarted the algorithm 10 times with random $r_{1:N}$, and recorded the results:

	Average Success(%)	Max Success(%)	Iteration time* (s)	Average convergence
EM1	70	100	7.5	3.2
EM2	73	100	12	2.9
Spectral	76	100	3.1	3.8

*PC: 3.33 GHz dual core CPU, 4 GB RAM, Software: MATLAB

Learning Infinite mixture of HMMs

- In Finite mixtures, we fix the number of clusters K a-priori.
- In infinite mixtures, K is automatically learned.

Learning Infinite mixture of HMMs

- In Finite mixtures, we fix the number of clusters K a-priori.
- In infinite mixtures, K is automatically learned.
- Learning an infinite mixture would involve an intractable integral over HMM parameters. We approximate these integrals using spectral learning.

Learning Infinite mixture of HMMs

Learning an infinite Mixture of HMMs would require an integral over HMM parameters $\theta = (O, A)$. (Collapsed Gibbs sampler)

Learning Infinite mixture of HMMs

Learning an infinite Mixture of HMMs would require an integral over HMM parameters $\theta = (O, A)$. (Collapsed Gibbs sampler)

- For existing clusters:

$$p(h_n = k, k \leq K | h_{1:N}^{-n}, \mathbf{x}_{1:N})$$
$$= \frac{N_k^{-n}}{N + \alpha - 1} \int p(x_n|\theta) p(\theta | \{x_l : l \neq n, h_l = k\}) d\theta$$

Learning Infinite mixture of HMMs

Learning an infinite Mixture of HMMs would require an integral over HMM parameters $\theta = (O, A)$. (Collapsed Gibbs sampler)

- For existing clusters:

$$p(h_n = k, k \leq K | h_{1:N}^{-n}, \mathbf{x}_{1:N})$$
$$= \frac{N_k^{-n}}{N + \alpha - 1} \int p(x_n|\theta) p(\theta | \{x_l : l \neq n, h_l = k\}) d\theta$$

- To open a new cluster:

$$p(h_n = k + 1 | h_{1:N}^{-n}, \mathbf{x}_{1:N})$$
$$= \frac{\alpha}{N + \alpha - 1} \int p(x_n|\theta) p(\theta) d\theta$$

Learning Infinite Mixture of HMMs

- One can use Gibbs sampling with auxiliary variable method of Neal (2000). However, this would require sampling from the prior to approximate the marginal likelihood. Large number of samples, slow.
- We suggest to use spectral learning for learning IM-HMMs.

IMM - Spectral learning

- Assuming that the sequences are long enough, we make the following approximation;

$$\int p(x_n|\theta)p(\theta|\{x_l : l \neq n, h_l = k\})d\theta \approx p(x_n|\theta_k^{spectral})$$

IMM - Spectral learning

- Assuming that the sequences are long enough, we make the following approximation;

$$\int p(x_n|\theta)p(\theta|\{x_l : l \neq n, h_l = k\})d\theta \approx p(x_n|\theta_k^{spectral})$$

- We compute the marginal likelihood of each sequence offline, and then use them in each iteration.

IMM - Spectral learning

- Assuming that the sequences are long enough, we make the following approximation;

$$\int p(x_n|\theta)p(\theta|\{x_l : l \neq n, h_l = k\})d\theta \approx p(x_n|\theta_k^{spectral})$$

- We compute the marginal likelihood of each sequence offline, and then use them in each iteration.
- By modifying the spectral algorithm for finite mixtures, we obtain a learning algorithm for infinite mixtures of HMMs:

Cheaper Alternative Algorithm for HMM Mixture learning

Randomly initialize $h_{1:N}$
At each iteration,
for $k = 1 \rightarrow K$ **do**
  $\theta_k \leftarrow \text{SpectralHMM}(\{x_n | \forall n, h_n = k\})$
end for
for $n = 1 \rightarrow N$ **do**
  $h_n = \arg\max_k p(x_n|\theta_k)$
end for

Cheaper Alternative Algorithm for HMM Mixture learning

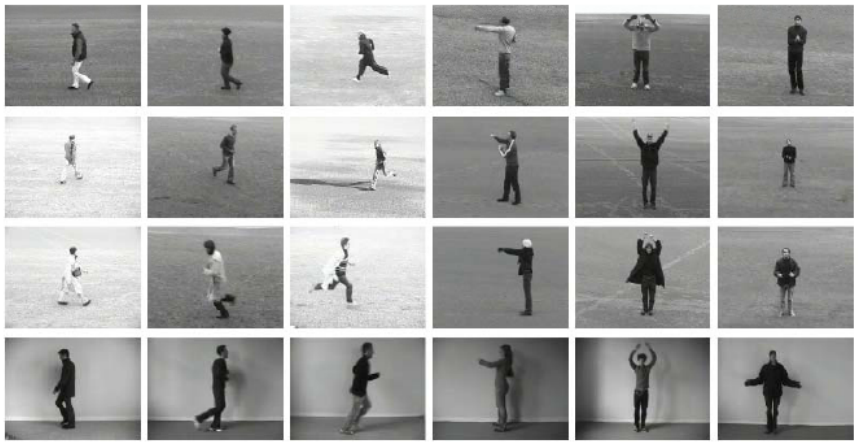
Randomly initialize $h_{1:N}$
At each iteration,
for $k = 1 \rightarrow K$ **do**
  $\theta_k \leftarrow \text{SpectralHMM}(\{x_n | \forall n, h_n = k\})$
end for
for $n = 1 \rightarrow N$ **do**
  $h_n = \arg\max_k [p(x_n|\theta_k) \quad p(x_n)]$
  if $h_n > K$ **then**
    $K = K + 1$
  end if
end for

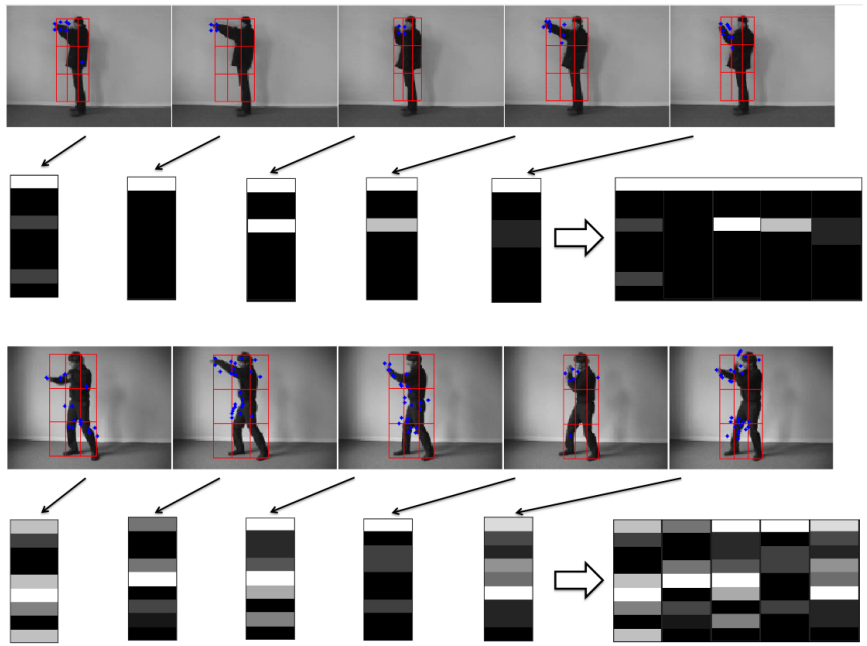
Cheaper Alternative Algorithm for HMM Mixture learning

Randomly initialize $h_{1:N}$
At each iteration,
for $k = 1 \rightarrow K$ **do**
  $\theta_k \leftarrow \text{SpectralHMM}(\{x_n | \forall n, h_n = k\})$
end for
for $n = 1 \rightarrow N$ **do**
  $h_n = \arg\max_k [p(x_n|\theta_k) \quad p(x_n)]$
  if $h_n > K$ **then**
    $K = K + 1$
  end if
end for

- Marginal likelihoods are computed before starting the algorithm $p(x_n)$
- Note that this algorithm is the analogue of the infinite version of k-means, Dirichlet Process Means algorithm of Kulis and Jordan.

Results - Human Action Classification





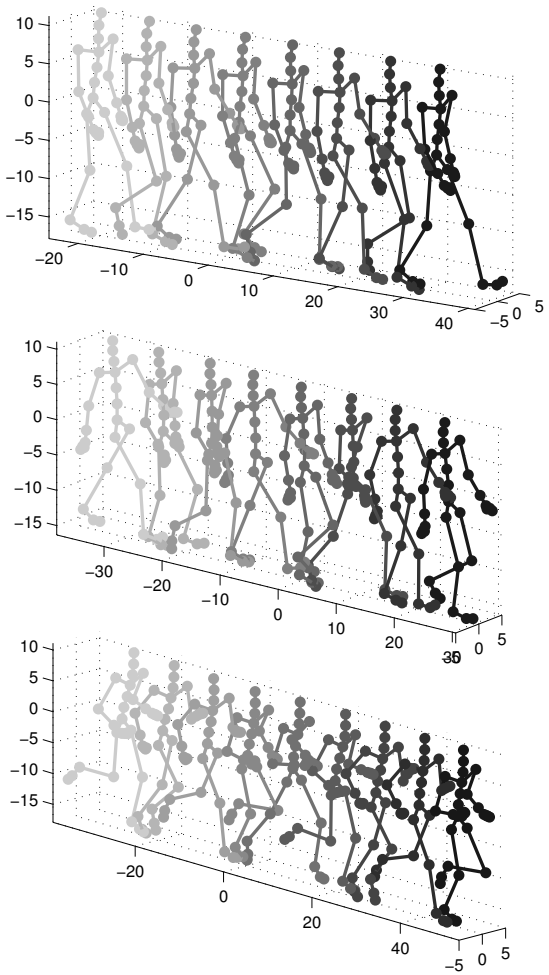
- We use KTH human action database. 6 classes (boxing, hand waving, hand clapping, walking, jogging, running), 100 sequences: 64 sequences for training 36 for testing.
- We do clustering in training phase to increase the clustering accuracy, as data instances in same class tend to show variability.

Results - Human Action Classification

Algorithm	Classification Accuracy
No Clustering	70.1%
Spectral	74.1%
Gibbs Sampling	74.0%

Results - MoCap Data

A dataset 44 sequences with two different walking characteristics and running. Algorithm automatically determines $K = 3$. In 9 seconds algorithm converged.



Conclusions - Spectral Learning for Mixture of HMMs

- We have proposed a simple, fast and effective algorithm for learning mixture of HMMs.
- Algorithm is competitive with more conventional Gibbs sampling method.

Outline

Discriminative vs Generative Models

- Given the training set $(\mathbf{x}_n, h_n)_{n=1}^N$

Training a Generative Model

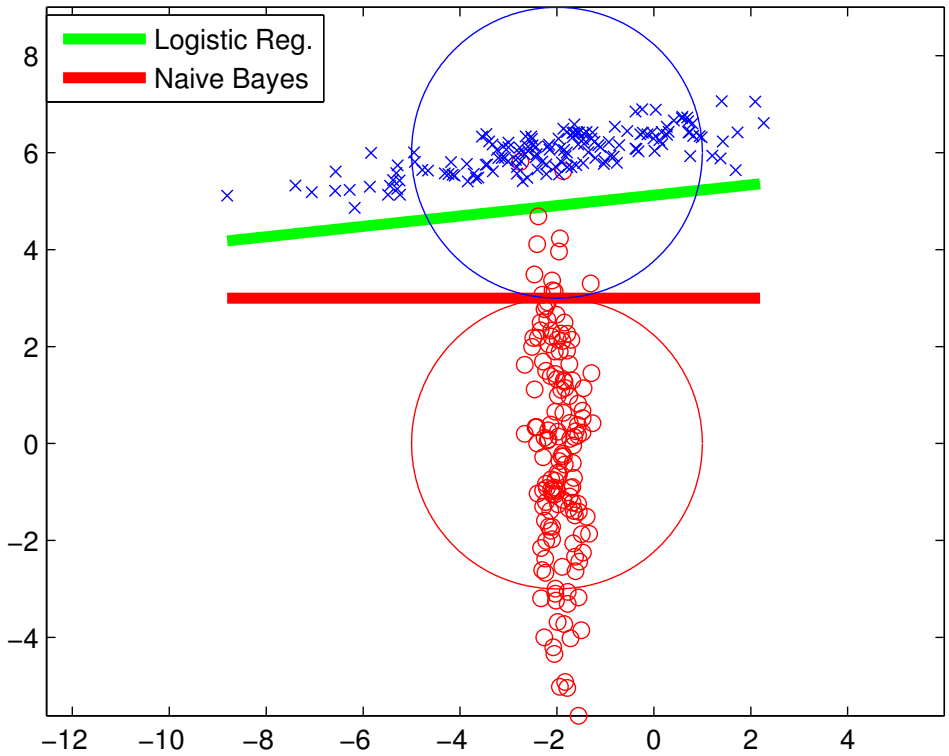
$$\theta^* = \arg\max_\theta \prod_n p(\mathbf{x}_n|\ \theta, h_n)$$

Training a Discriminative Model

$$\theta^* = \arg\max_\theta \prod_n p(h_n|\mathbf{x}_n, \theta)$$

Discriminative vs Generative Models

- Logistic Regression vs Naive Bayes Classifiers (with Isotropic Gaussian observations)



- Generative Models model the data distribution $p(x|h)$.
- Discriminative Models model the class labels $p(h|x)$.

Discriminative and Generative Time Series Models

- HMM, Markov Model and their mixtures are all generative models.

Discriminative and Generative Time Series Models

- HMM, Markov Model and their mixtures are all generative models.
- Deriving a discriminative model corresponding to a LVM amounts to converting the model into an undirected graph. (Defining an energy function)

Discriminative and Generative Time Series Models

- HMM, Markov Model and their mixtures are all generative models.
- Deriving a discriminative model corresponding to a LVM amounts to converting the model into an undirected graph. (Defining an energy function)
- For Discriminative Markov model:

$$\psi(\mathbf{x}_n, h_n; \theta_k) = \sum_{t=1}^{T_n} \sum_{k=1}^{K} \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} [h_n = k][l_1 = x_t][l_2 = x_{t-1}]\theta_{k,l_1,l_2}$$

Discriminative and Generative Time Series Models

- HMM, Markov Model and their mixtures are all generative models.
- Deriving a discriminative model corresponding to a LVM amounts to converting the model into an undirected graph. (Defining an energy function)
- For Discriminative Markov model:

$$\psi(\mathbf{x}_n, h_n; \theta_k) = \sum_{t=1}^{T_n} \sum_{k=1}^{K} \sum_{l_1=1}^{L} \sum_{l_2=1}^{L} [h_n = k][l_1 = x_t][l_2 = x_{t-1}]\theta_{k,l_1,l_2}$$

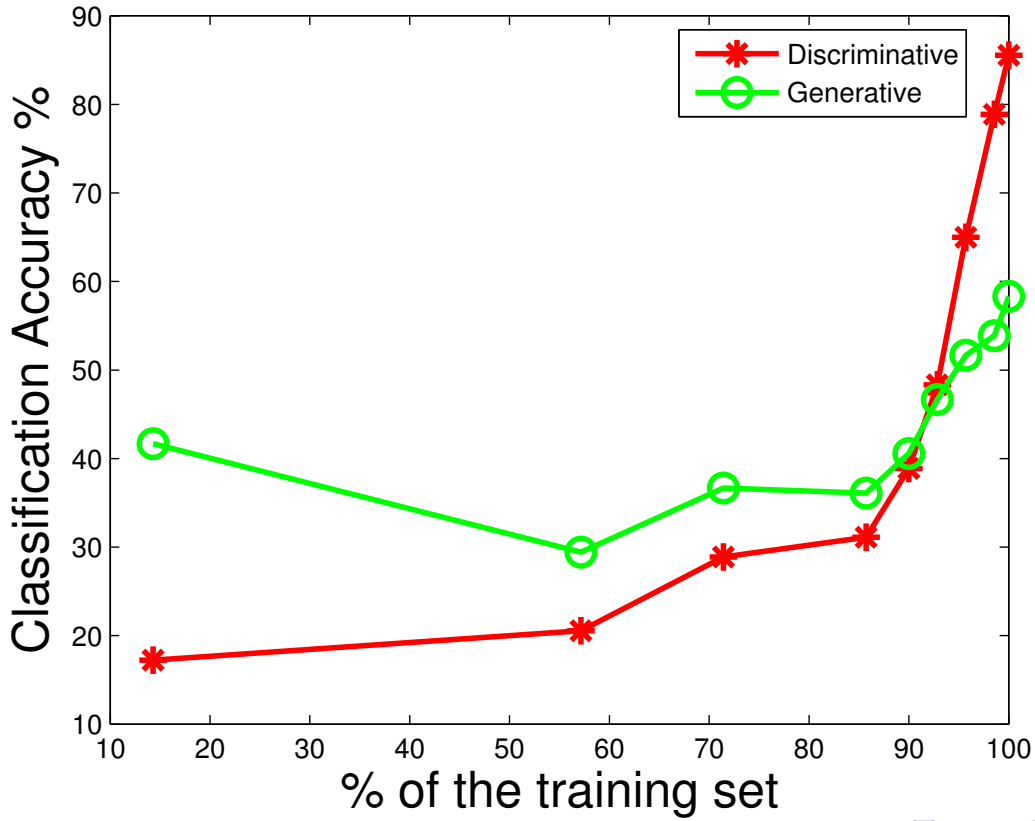
- The training is:

$$\theta^*_{1:K} = \arg\max_{\theta_{1:K}} \sum_{n=1}^{N} \log \frac{\exp(\psi(\mathbf{x}_n, h_n; \theta_k))}{\sum_{h_n=1}^{K} \exp(\psi(\mathbf{x}_n, h_n; \theta_k))}$$

- Note that we treat data x as an input.

Result

- We investigate amount of training data vs classification accuracy for generative and discriminative Markov model
- We use the synthetic control chart dataset from UCI machine learning repository. 6 classes 100 sequences for each class. 70 sequence for training 30 sequence for test.



Conclusions and Future Work

- We have proposed two novel algorithms for time series clustering.
- We have investigated generative and discriminative classification of time series.
- As future work, we plan on investigating the hierarchical Bayesian viewpoint for deriving spectral learning algorithms for more complicated time series models.
- A complete spectral learning algorithm for mixture of HMMs based on constrained optimization.