Why Mixtures of Markov Chains

Generative model of sequences from different sources

Extends markov chains in a simple but powerful way

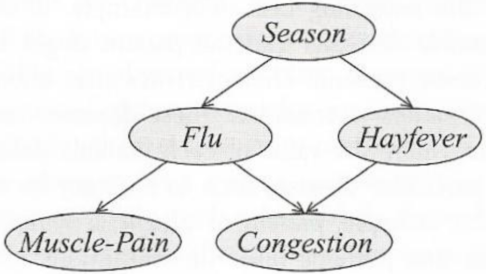
Intuitive way of thinking about clustering of sequences

Obvious choice from a probabilistic graphical model (PGM) approach

Not described in full generality in literature (to the best of my knowledge)

Graphical models make it easier to examine dependencies and influences between random variables

Graph Representation



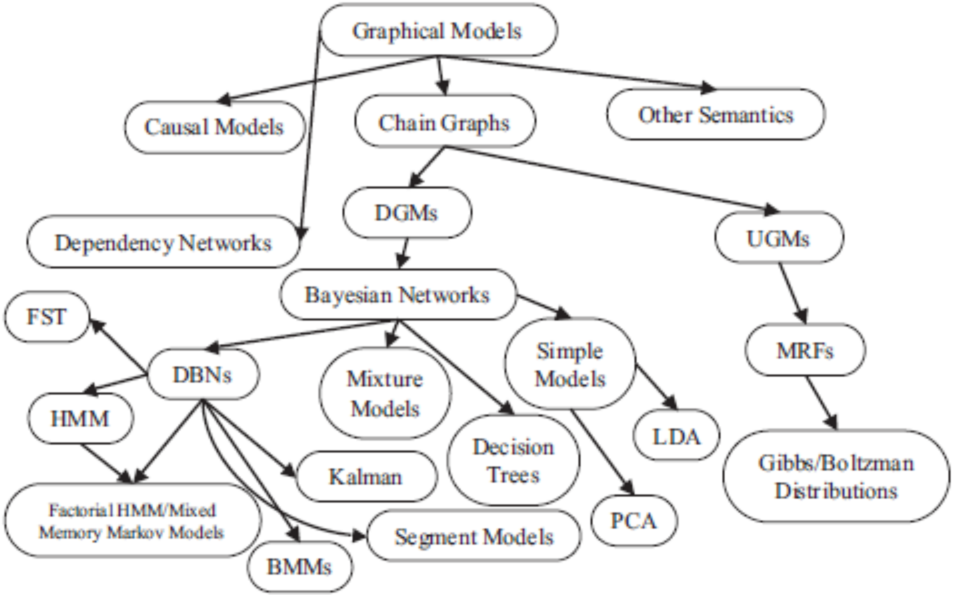
Independencies

$$(F \perp H \mid S)$$
$$(C \perp S \mid F, H)$$
$$(M \perp H, C \mid F)$$
$$(M \perp C \mid F)$$

Factorization

$$P(S, F, H, C, M) = P(S)P(F \mid S)$$
$$P(H \mid S)P(C \mid F, H)P(M \mid F)$$

Large family including Directed, Undirected and Chain Graphs



Relevant for today's talk are only the directed ones aka Bayesian Networks

Directed Graphical Models: Arrows and Boxes

boxes (nodes)= variables

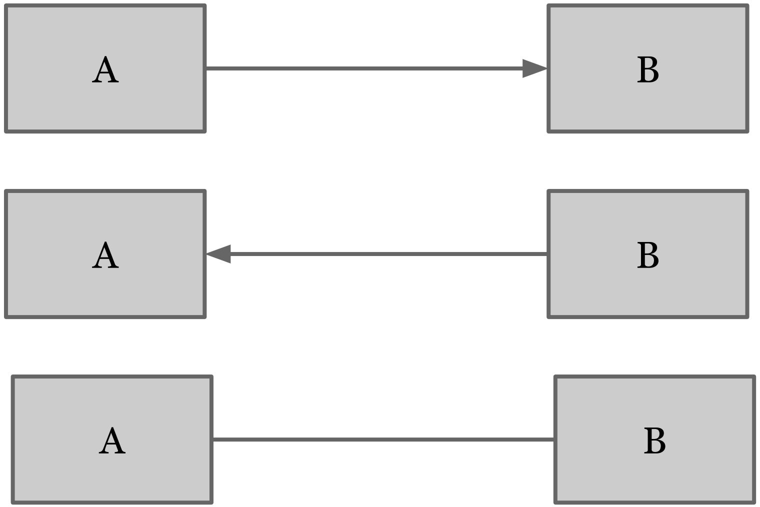
arrows (edges)= factorisation (break-down) of joint probability distribution

Different ways of factorising joint

$$p(a)\,p(b|a)$$

$$p(b)\,p(a|b)$$

$$\frac{\psi(a,b)}{\sum \psi(a,b)}$$



A less general interpretation, but easier to understand

boxes = variables

arrows = causal influences

PGM's can be classified as Generative or Discriminative

Generative

Full joint (of inputs and outputs) probability distribution - P(X,Y)

Better for exploration and unsupervised learning

Mixtures of Gaussians, Hidden Markov Models, Factor Analysis

Discriminative

Only conditional (of outputs on inputs) probability distribution - P(Y|X)

Better for prediction and supervised learning

Maximum Entropy (aka Logistic Regression), Conditional Random Fields

Generative models for sequences (almost) all based on Markov Chains

Markov Chains make a simplifying assumption

$$s_1\ s_2\ \dots\ s_T = s_{1:T}$$

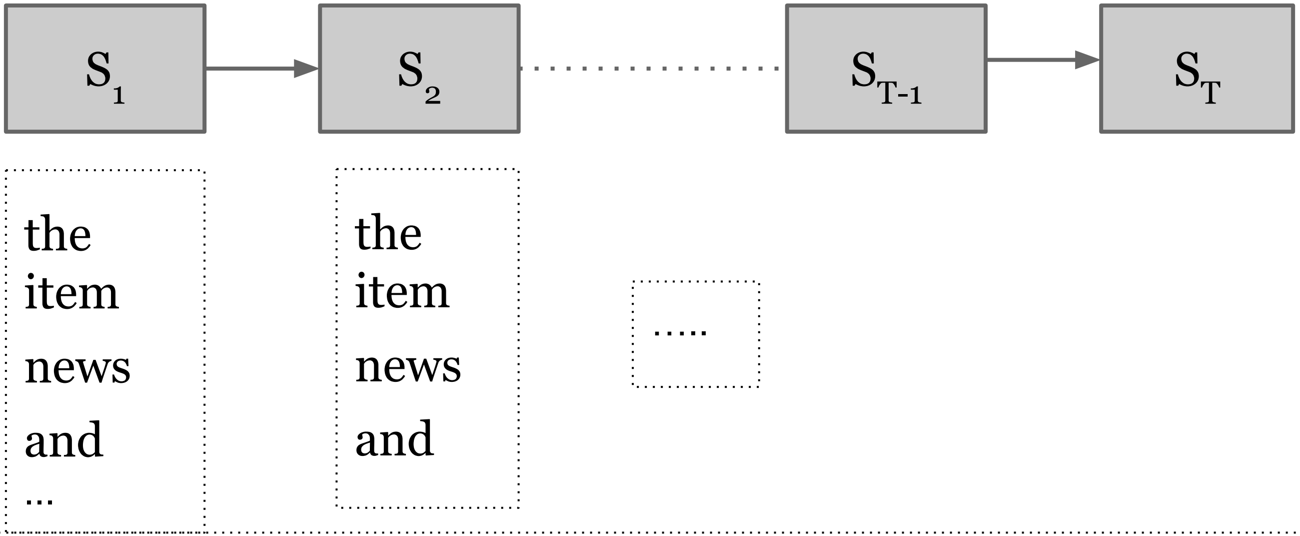
$$p(s_{1:T}) = p(s_1)\ p(s_2|s_1)\dots p(s_T|s_{T-1})$$

probability of a state in the the chain at a given time step is independent of rest of the chain, given recent history

(shown is 1st order chain)

Markov Chains are Dynamical Bayesian Networks, ie, unrolled over time

each variable represents a symbol at a given time step, all variables have the same domain



Markov Chains are Dynamical Bayesian Networks, ie, unrolled over time

each variable represents a symbol at a given time step, all variables have the same domain

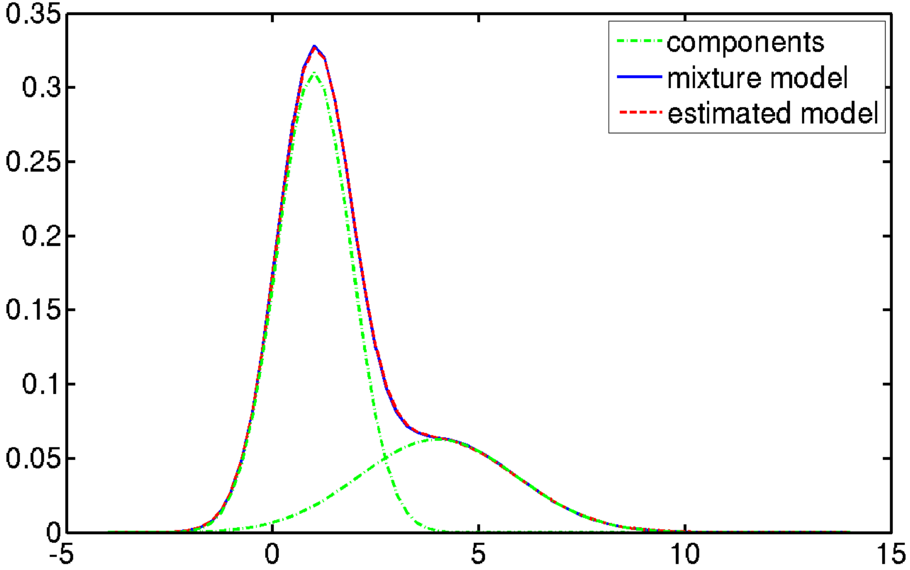
$$S_1 \rightarrow S_2 \cdots\cdots S_{T-1} \rightarrow S_T$$

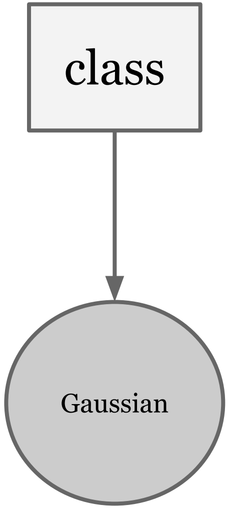
$p(s_t|s_{t-1})$ = transition matrix

(technically also need $p(s_0)$ but can be folded into transition matrix conditioning on start symbol which has probability one)

Mixtures are Bayesian Networks with a hidden variable representing the mixture component / class

allow you to increase the complexity of your model while still keeping the original of probability distribution



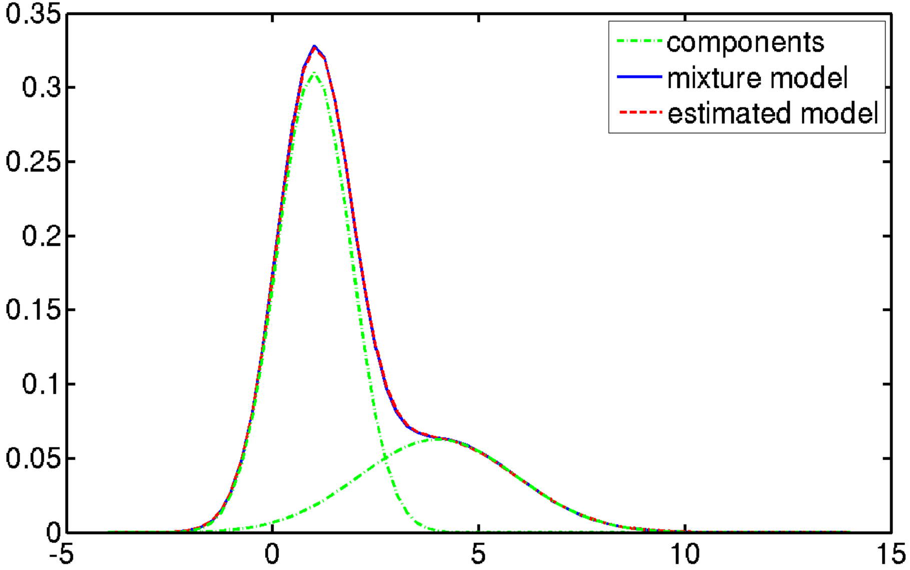


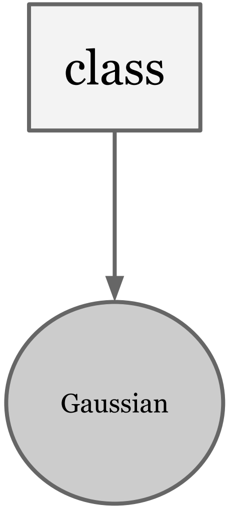
p(class)

p(point | class) ~ N(m,s)

Mixtures are Bayesian Networks with a hidden variable representing the mixture component / class

makes bests sense when data point actually come from different populations



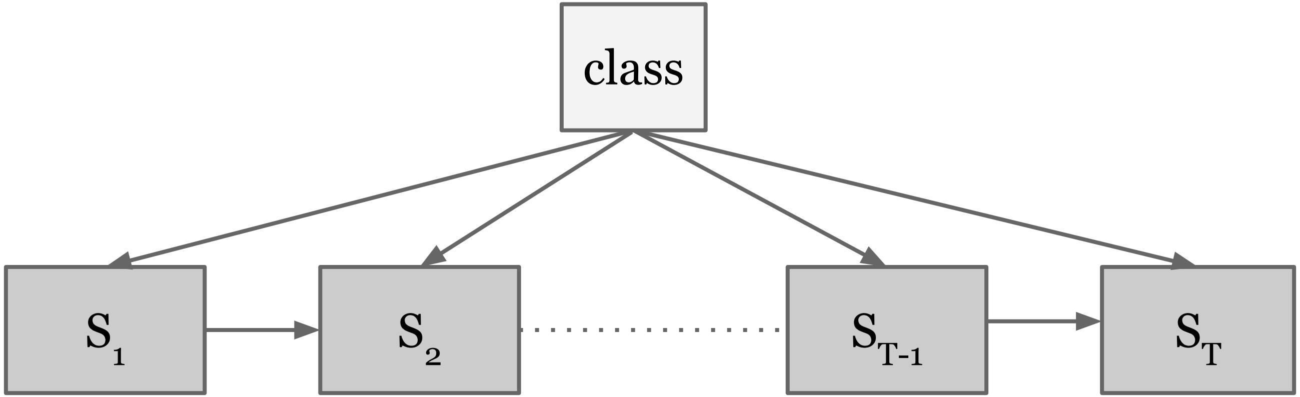


joint = p(class,point)

Mixtures of Markov Chains
are both Mixtures and Markov Chains

assumes sequences in data have different origins/characteristics

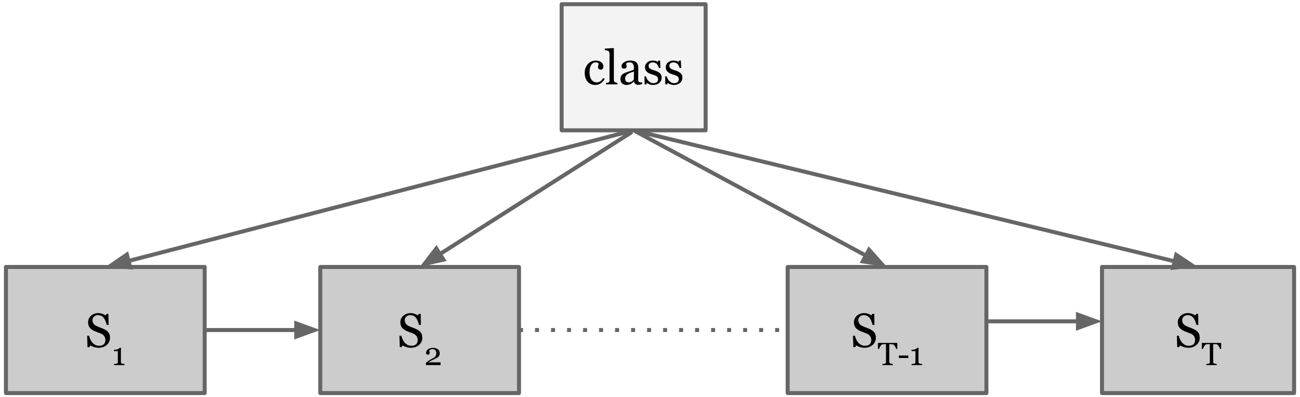
$$p(c, s_{1:T}) = p(c)\, p(s_1|c)\, p(s_2|s_1, c)\ldots p(s_T|s_{T-1}, c)$$



Mixtures of Markov Chains
are both Mixtures and Markov Chains

$p(c)$ = class (prior) probability distribution

$p(s_t|s_{t-1}, c)$ = class-conditional transition matrix



Questions you might want to ask

Q: Heres a sequence $s_{1:T}$, which class(es) does it belong to? (fuzzy clustering)

A: compute posterior probability distribution over class variable

$$p(c|s_{1:T}) = p(c , s_{1:T})/p(s_{1:T})$$
$$p(c|s_{1:T}) = p(c)\, p(s_{1:T}|c)/p(s_{1:T})$$
$$p(c|s_{1:T}) = p(c)\, p(s_{1:T}|c) \,/\, [\, p(c)\, p(s_{1:T}|c) + p(\neg c)\, p(s_{1:T}|\neg c) \,]$$

sum-product inference algorithm O(nk)

Questions you might want to ask

Q: What's the next most likely symbol given a partial sequence, e.g. $s_1\ s_2$ _ ?

A: Compute MAP/Most probable explanation

$$s_3 = \max \arg p(s_1\ s_2\ \_)$$

Q: What's the most likely sequence of length L, e.g., _ _ _?

A: Same

Viterbi inference, special case of max-product algorithm, $O(L|S|^2)$

Questions you might want to ask

Q: Is this new sequence $s_{1:T}$ typical?

A: Compute likelihood of point and compare with reference (most likely sequence in dataset , for instance)

$$p(\text{sequence}|M) \text{ vs } p(\text{reference}|M)$$

$$p(\text{sequence}|M) = p(s_{1:T}) = [\ p(c)\ p(s_{1:T}|c) + p(\neg c)\ p(s_{1:T}|\neg c)\ ]$$

sum-product inference algorithm O(nk)

Questions you might want to ask

Q: How good a model of my data is it?

A: Generate data and inspect

Forward Sampling (N):

1. Generate a class from p(c)
2. Generate first symbol from class-conditional transition matrix $p(s_t|s_{t-1}, c)$
3. Generate the next symbol conditioned on the previous one
4. Stop if current symbol is the end symbol and go back to 1 until N

Parameter Estimation

class (prior) probabilities

$p(c)$

class-conditional transition matrices

$p(s_t|s_{t-1}, c)$

Parameter Estimation (Maximum Likelihood)

Two methods

Constrained non-linear optimisation

Expectation Maximisation (EM) O(kn(h))

EM for discrete PGMs is actually simple in practice

Visible Variables

A : {a,¬a} B : {b,¬b}

Data

a , b

a , ¬b

¬a, b

....

Estimates

p(a) ~ #(a)/N

p(b|a) ~ #(a,b)/#(a,*)

....

EM for discrete PGMs is actually simple in practice

Hidden Variables: Posterior as count

$A : \{a, \neg a\}$ B : {b,¬b}

Data

?, b

? , ¬b

?, b

?, b

....

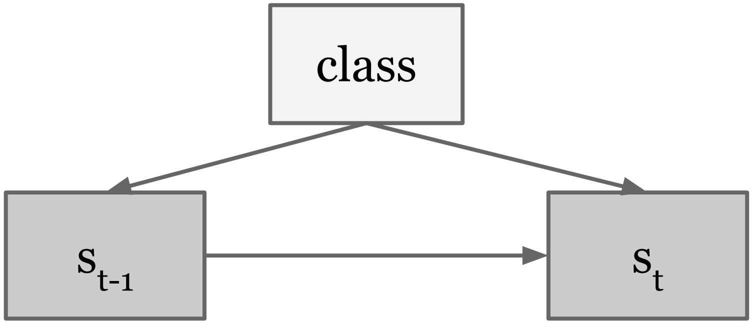
Estimates

$$p(a) \approx \frac{\sum p(a|B)}{N}$$

$$p(b|a) \approx \frac{\sum p(a|b)}{\sum p(a|B)}$$

EM for Mixtures of markov chains

Hidden Variables: Posterior trick



Data

?, $s_{1:T}^{1}$

?, $s_{1:T}^{2}$

?, $s_{1:T}^{3}$

....

Estimates

$$p(class) \approx \frac{\sum p(class|s_{1:T})}{N}$$

$$p(s'|s, class) \approx \frac{\sum p(class|s_{1:T}\ if\ ss' \in s_{1:T})}{\sum p(class|s_{1:T}\ if\ s \in s_{1:T})}$$

Simple EM loop

Until convergence

Expectation

for each data point

for each class

compute posterior

add to class and transition counts

Maximisation

normalise counts

let new class and transition probabilities be the normalised counts

EM> Practical considerations > stopping criterion

Stop when *fit* to *data* stops increasing *significantly*

fit defined as log-likelihood, ie, probability of whole data set

data defined as training data or development data, latter better for generalisation on new data points

significantly is problem dependent, if using development dataset log-likelihood will actually decrease

EM > Practical considerations > local maxima

EM only guarantees improvement to nearest maximum

Random restarts trick

run EM r times starting with different initial parameters, keep best estimated model

brings complexity to O(kn(h)r)

EM > Practical considerations > Sparsity and Smoothing

specially when dealing with language, sparsity is a big problem if unseen data is going to be fed into the model, as most transitions between token will not have been seen during training

Lots of heuristic smoothing algorithms

all involve transferring some of the probability mass of seen transitions to unseen ones

Can be thought of as hacky Bayesian prior on transition counts

EM > Practical considerations > initialisation

Necessary on large datasets with lots of unique tokens

Typically use fast clustering algorithms such as K-means

Can potentially do most of the work

EM > Practical considerations > model complexity

how many classes?

what order markov chain (1st, 2nd..)?

Relationship to other models

Conditional Markov Chains, the supervised generative version

Maximum entropy models, the supervised discriminative version

Mixtures of Hidden Markov Models, with added corruption

Mixtures of Multinomials, assumes time independence

Conditional Multinomials/Naive Bayes, supervised version of Mixtures of Multinomials

Thank you!

jose.llarena@gmail.com

github.com/josellarena