STA 4273H: Statistical Machine Learning

Russ Salakhutdinov

Department of Statistics
rsalakhu@utstat.toronto.edu
http://www.utstat.utoronto.ca/~rsalakhu/
Sidney Smith Hall, Room 6002

Lecture 11
Project Deadline

• Project Deadline is Dec 15, 2011. This is a hard deadline!

• I will not be able to give any extensions, as I have to turn in the marks by Dec 21.

• Send the projects to me via e-mail to: rsalakhu@utstat.toronto.edu
Sequential Data

• So far we focused on problems that assumed that the data points were independent and identically distributed (i.i.d. assumption).

• Express the likelihood function as a product over all data points of the probability distribution evaluated at each data point.

• Poor assumption when working with sequential data.

• For many applications, e.g. financial forecasting, we want to predict the next value in a time series, given past values.

• Intuitively, the recent observations are likely to be more informative in predicting the future.

• Markov models: future predictions are independent of all but the most recent observations.
Example of a Spectrogram

- Example of a spectrogram of a spoken word ‘Bayes theorem’:

- Successive observations are highly correlated.
Markov Models

• The simplest model is the first-order Markov chain:

\[ p(x_1, \ldots, x_N) = p(x_1) \prod_{n=2}^{N} p(x_n|x_{n-1}). \]

• From the d-separation property, the conditionals are given by:

\[ p(x_n|x_1, \ldots, x_{n-1}) = p(x_n|x_{n-1}). \]

• For many applications, these conditional distributions that define the model will be constrained to be equal.
• This corresponds to the assumption of a stationary time series.
• The model is known as homogenous Markov chain.
Second-Order Markov Models

- We can also consider a second-order Markov chain:

- The joint distribution for a sequence of $N$ observations under this model is:

$$ p(x_1, \ldots, x_N) = p(x_1)p(x_2|x_1) \prod_{n=3}^{N} p(x_n|x_{n-1}, x_{n-2}). $$

- We can similarly consider extensions to an $M^{th}$ order Markov chain.
- Increased flexibility $\rightarrow$ Exponential growth in the number of parameters.

- Markov models need big orders to remember past “events”.

Learning Markov Models

• The ML parameter estimates for a simple Markov model are easy. Consider a $K^{th}$ order model:

$$p(x_1, \ldots, x_N) = \prod_{n=k+1}^{N} p(x_n|x_{n-1}, x_{n-2}, \ldots, x_{n-k}).$$

• Each window of $k + 1$ outputs is a training case for the model.

$$p(x_n|x_{n-1}, x_{n-2}, \ldots, x_{n-k}).$$

• Example: for discrete outputs (symbols) and a 2nd-order Markov model we can use the multinomial model:

$$p(x_n = m|x_{n-1} = a, x_{n-2} = b) = \alpha_{mab}.$$

• The maximum likelihood values for $\alpha$ are:

$$\alpha_{mab}^* = \frac{\text{num}[n, \text{s.t. } x_n = m, x_{n-1} = a, x_{n-2} = b]}{\text{num}[n, \text{s.t. } x_{n-1} = a, x_{n-2} = b]}.$$
State Space Models

- How about the model that is not limited by the Markov assumption to any order.

- Solution: Introduce additional latent variables!

- Graphical structure known as the State Space Model.

- For each observation $x_n$, we have a latent variable $z_n$. Assume that latent variables form a Markov chain.

- If the latent variables are discrete $\rightarrow$ Hidden Markov Models (HMMs). Observed variables can be discrete or continuous.

- If the latent and observed variables are Gaussian $\rightarrow$ Linear Dynamical System.
State Space Models

• The joint distribution is given by:

\[
p(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = p(\mathbf{z}_1) \prod_{n=2}^{N} p(\mathbf{z}_n | \mathbf{z}_{n-1}) \prod_{n=1}^{N} p(\mathbf{x}_n | \mathbf{z}_n).
\]

• There is always a path connecting two observed variables \(x_n, x_m\) via latent variables.

• The predictive distribution:

\[
p(\mathbf{x}_{n+1} | \mathbf{x}_1, \ldots, \mathbf{x}_N)
\]

does not exhibit any conditional independence properties! And so prediction depends on all previous observations.

• Even though hidden state sequence is first-order Markov, the output process is not Markov of any order!
Hidden Markov Model

- First order Markov chain generates hidden state sequence (known as transition probabilities):

  \[ p(z_n = k | z_{n-1} = j) = A_{jk}, \quad p(z_1 = k) = \pi_k. \]

- A set of output probability distributions (one per state) converts state path into sequence of observable symbols/vectors (known as emission probabilities):

  \[ p(x_n | z_n, \phi). \]

Gaussian, if \( x \) is continuous.
Conditional probability table if \( x \) is discrete.

\[
p(x_1, \ldots, x_N, z_1, \ldots, z_N) = p(z_1) \prod_{n=2}^{N} p(z_n | z_{n-1}) \prod_{n=1}^{N} p(x_n | z_n).\]
Links to Other Models

• You can view HMM as: A Markov chain with stochastic measurements.

• Or a mixture model with states coupled across time.

We will adopt this view, as we worked with mixture model before.
Transition Probabilities

- It will be convenient to use 1-of-K encoding for the latent variables.
- The matrix of transition probabilities takes form:
  \[ p(z_{nk} = 1 | z_{n-1,j} = 1) = A_{jk}, \quad \sum_k A_{jk} = 1. \]
- The conditionals can be written as:
  \[
p(z_n|z_{n-1}, A) = \prod_{k=1}^{K} \prod_{j=1}^{K} A_{jk}^{z_{n-1,j}, z_{nk}}, \quad p(z_1|\pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}.
  \]
- We will focus on homogenous models: all of the conditional distributions over latent variables share the same parameters \( A \).
- Standard mixture model for i.i.d. data: special case in which all parameters \( A_{jk} \) are the same for all \( j \).
- Or the conditional distribution \( p(z_n|z_{n-1}) \) is independent of \( z_{n-1} \).
Emission Probabilities

• The emission probabilities take form:

\[ p(x_n | z_n, \phi) = \prod_{k=1}^{K} p(x_n | \phi_k)^{z_{nk}}. \]

• For example, for a continuous \( x \), we have

\[ p(x_n | z_n, \phi) = \prod_{k=1}^{K} \mathcal{N}(x_n | \mu_k, \Sigma_k)^{z_{nk}}. \]

• For the discrete, multinomial observed variable \( x \), using 1-of-K encoding, the conditional distribution takes form:

\[ p(x_n | z_n, \phi) = \prod_{i=1}^{D} \prod_{k=1}^{K} \mu_{ik}^{x_{ni}z_{nk}}. \]
HMM Model Equations

• The joint distribution over the observed and latent variables is given by:

\[ p(X, Z|\theta) = p(z_1|\pi) \prod_{n=2}^{N} p(z_n|z_{n-1}, A) \prod_{n=1}^{N} p(x_n|z_n, \phi), \]

where \( \theta = \{\pi, A, \phi\} \) are the model parameters.

• Data are not i.i.d. Everything is coupled across time.

• Three problems: computing probabilities of observed sequences, inference of hidden state sequences, learning of parameters.
HMM as a Mixture Through Time

- Sampling from a 3-state HMM with a 2-d Gaussian emission model.

- The transition matrix is fixed: $A_{kk} = 0.9$ and $A_{jk} = 0.05$. 
Applications of HMMs

• Speech recognition.
• Language modeling
• Motion video analysis/tracking.
• Protein sequence and genetic sequence alignment and analysis.
• Financial time series prediction.
Maximum Likelihood for the HMM

• We observe a dataset $X = \{x_1, \ldots, x_N\}$.
• The goal is to determine model parameters $\theta = \{\pi, A, \phi\}$.
• The probability of observed sequence takes form:

$$p(X|\theta) = \sum_Z p(X, Z|\theta).$$

$$p(\text{observed sequence}) = \sum_{\text{all paths}} p(\text{observed outputs, state paths}).$$

• In contrast to mixture models, the joint distribution $p(X, Z | \theta)$ does not factorize over $n$.

• It looks hard: $N$ variables, each of which has $K$ states. Hence $N^K$ total paths.

• Remember inference problem on a simple chain.
Probability of an Observed Sequence

• The joint distribution factorizes:

\[
p(X|\theta) = \sum_Z p(X, Z|\theta) = \sum_{z_1, \ldots, z_n} p(z_1, x_1) \prod_{n=2}^{N} p(z_n|z_{n-1})p(x_n|z_n)
\]

\[
= \sum_{z_1} p(z_1)p(x_1|z_1) \sum_{z_2} p(z_2|z_1)p(x_2|z_2) \ldots \sum_{z_N} p(z_N|z_{N-1})p(x_N|z_N).
\]

• **Dynamic Programming**: By moving the summations inside, we can save a lot of work.
EM algorithm

• We cannot perform direct maximization (no closed form solution):

\[ p(X|\theta) = \sum_Z p(X, Z|\theta). \]

• **EM algorithm:** we will derive efficient algorithm for maximizing the likelihood function in HMMs (and later for linear state-space models).

• **E-step:** Compute the posterior distribution over latent variables:

\[ p(Z|X, \theta^{old}). \]

• **M-step:** Maximize the expected complete data log-likelihood:

\[ Q(\theta, \theta^{old}) = \sum_Z p(Z|X, \theta^{old}) \log p(X, Z|\theta). \]

• If we knew the true state path, then ML parameter estimation would be trivial.

• We will first look at the E-step: Computing the true posterior distribution over the state paths.
Inference of Hidden States

- We want to estimate the hidden states given observations. To start with, let us estimate a single hidden state:

\[
\gamma(z_n) = p(z_n | X) = \frac{p(X | z_n) p(z_n)}{p(X)}.
\]

- Using conditional independence property, we obtain:

\[
p(z_n | X) = \frac{p(x_1, \ldots, x_n | z_n) p(x_{n+1}, \ldots, x_N | z_n) p(z_n)}{p(X)}
\]

\[= \frac{p(x_1, \ldots, x_n, z_n) p(x_{n+1}, \ldots, x_N | z_n)}{p(X)} = \frac{\alpha(z_n) \beta(z_n)}{p(X)}.
\]
Inference of Hidden States

• Hence:

\[
\gamma(z_n) = \frac{p(x_1, \ldots, x_n, z_n)p(x_{n+1}, \ldots, x_N|z_n)}{p(X)} = \frac{\alpha(z_n)\beta(z_n)}{p(X)}.
\]

\[\alpha(z_n) \equiv p(x_1, \ldots, x_n, z_n)\]

\[\beta(z_n) \equiv p(x_{n+1}, \ldots, x_N|z_n).\]

The joint probability of observing all of the data up to time n and \(z_n\).

The conditional probability of all future data from time \(n+1\) to \(N\).

• Each \(\alpha(z_n)\) and \(\beta(z_n)\) represent a set of K numbers, one for each of the possible settings of the 1-of-K binary vector \(z_n\).

• We will derive efficient recursive algorithm, known as the alpha-beta recursion, or forward-backward algorithm.

• Remember the sum-product message passing algorithm for tree-structured graphical models.
The Forward ($\alpha$) Recursion

• The forward recursion:

$$\alpha(z_n) = p(x_1, \ldots, x_n, z_n)$$

$$= p(x_n|z_n)p(x_1, \ldots, x_{n-1}, z_n)$$

$$= p(x_n|z_n) \sum_{z_{n-1}} p(x_1, \ldots, x_{n-1}, z_{n-1}, z_n)$$

$$= p(x_n|z_n) \sum_{z_{n-1}} p(x_1, \ldots, x_{n-1}, z_{n-1})p(z_n|z_{n-1})$$

$$= p(x_n|z_n) \sum_{z_{n-1}} \alpha(z_{n-1})p(z_n|z_{n-1})$$

Computational cost scales like $O(K^2)$.

• Observe:

$$p(X) = \sum_{z_N} \alpha(z_N).$$

• This enables us to easily (cheaply) compute the desired likelihood.
The Forward ($\alpha$) Recursion

• The forward recursion:

Exponentially many paths.

At each node, sum up the values of all incoming paths.

• This is exactly dynamic programming.
The Forward ($\alpha$) Recursion

- Illustration of the forward recursion

Here $\alpha(z_{n,1})$ is obtained by:

- Taking the elements $\alpha(z_{n-1,j})$
- Summing the up with weights $A_{j1}$, corresponding to $p(z_n | z_{n-1})$
- Multiplying by the data contribution $p(x_n | z_{n1})$

$$
\alpha(z_n) = p(x_n | z_n) \sum_{z_{n-1}} \alpha(z_{n-1}) p(z_n | z_{n-1})
$$

- The initial condition is given by:

$$
\alpha(z_1) = p(x_1 | z_1) p(z_1) = \prod_{k=1}^{K} \left[ \pi_k p(x_1 | \phi_k) \right]^{z_{1,k}}.
$$
The Backward ($\beta$) Recursion

- There is also a simple recursion for $\beta(z_n)$:

$$
\beta(z_n) = p(x_{n+1}, \ldots, x_N | z_n)
= \sum_{z_{n+1}} p(x_{n+1}, \ldots, x_N, z_{n+1} | z_n)
= \sum_{z_{n+1}} p(x_{n+1}, \ldots, x_N | z_{n+1}, z_n) p(z_{n+1} | z_n)
= \sum_{z_{n+1}} p(x_{n+2}, \ldots, x_N | z_{n+1}) p(x_{n+1} | z_{n+1}) p(z_{n+1} | z_n)
= \sum_{z_{n+1}} \beta(z_{n+1}) p(x_{n+1} | z_{n+1}) p(z_{n+1} | z_n)
$$
The Backward ($\beta$) Recursion

- Illustration of the backward recursion

$$
\beta(z_{n+1}) \quad \beta(z_{n+1,1}) \\
\downarrow \quad \downarrow \\
k = 1 \quad A_{11} \quad p(x_n|z_{n+1,1})
$$

$$
\beta(z_{n+1,2}) \\
\downarrow \\
k = 2 \quad A_{12} \quad p(x_n|z_{n+1,2})
$$

$$
\beta(z_{n+1,3}) \\
\downarrow \\
k = 3 \quad n + 1 \quad p(x_n|z_{n+1,3})
$$

$$
\beta(z_n) = \sum_{z_{n+1}} \beta(z_{n+1}) p(x_{n+1}|z_{n+1}) p(z_{n+1}|z_n)
$$

- Initial condition:

$$
p(z_N|X) = \frac{\alpha(z_N)\beta(z_N)}{p(X)} = \frac{p(X,z_N)\beta(z_N)}{p(X)}.
$$

- Hence:

$$
\beta(z_N) = 1.
$$
The Backward ($\beta$) Recursion

- $\alpha(z_{nk})$ gives total inflow of probability to node (n,k).
- $\beta(z_{nk})$ gives total outflow of probability.

- In fact, we can do one forward pass to compute all the $\alpha(z_n)$ and one backward pass to compute all the $\beta(z_n)$ and then compute any $\gamma(z_n)$ we want. Total cost is $O(K^2N)$. 

Computing Likelihood

• Note that

\[ \sum_{z_n} \gamma(z_n) = \sum_{z_n} p(z_n|X) = 1. \]

• We can compute the likelihood at any time using \( \alpha - \beta \) recursion:

\[ p(X|\theta) = \sum_{z_n} \alpha(z_n)\beta(z_n). \]

• In the forward calculation we proposed originally, we did this at the final time step \( n = N \).

\[ p(X|\theta) = \sum_{z_N} \alpha(z_N). \]

Because \( \beta(z_n)=1 \).

• This is a good way to check your code!
Two-Frame Inference

• We will also need the cross-time statistics for adjacent time steps:

\[
\xi(z_{n-1}, z_n) = \frac{p(z_{n-1}, z_n | X)}{p(X)}
= \frac{p(X | z_{n-1}, z_n) p(z_{n-1}, z_n)}{p(X)}
= \frac{p(x_1, \ldots, x_{n-1} | z_{n-1}) p(x_n | z_n) p(x_{n+1}, \ldots, x_N | z_n) p(z_n | z_{n-1}) p(z_{n-1})}{p(X)}
= \frac{\alpha(z_{n-1}) p(x_n | z_n) p(z_n | z_{n-1}) \beta(z_n)}{p(X)}.
\]

• This is a $K \times K$ matrix with elements $\xi(i,j)$ representing the expected number of transitions from state $i$ to state $j$ that begin at time $n-1$, given all the observations.

• It can be computed with the same $\alpha$ and $\beta$ recursions.
EM algorithm

- **Intuition**: if only we knew the true state path then ML parameter estimation would be trivial.

- E-step: Compute the posterior distribution over the state path using $\alpha - \beta$ recursion (dynamic programming):

$$p(Z|X, \theta^{old}).$$

- M-step: Maximize the expected complete data log-likelihood (parameter re-estimation):

$$Q(\theta, \theta^{old}) = \sum_Z p(Z|X, \theta^{old}) \log p(X, Z|\theta).$$

- We then iterate. This is also known as a **Baum-Welch algorithm** (special case of EM).

- In general, finding the ML parameters is NP hard, so initial conditions matter a lot and convergence is hard to tell.
Complete Data Log-likelihood

• Complete data log-likelihood takes form:

$$
\log p(X, Z|\theta) = \log \left[ p(z_1|\pi) \prod_{n=2}^{N} p(z_n|z_{n-1}, A) \prod_{n=1}^{N} p(x_n|z_n, \phi) \right]
$$

$$
= \log \left[ \prod_{k=1}^{K} \pi_k^{z_{1k}} \prod_{n=2}^{N} \prod_{k=1}^{K} \prod_{j=1}^{K} A_{jk}^{z_{n-1,j}z_{nk}} \prod_{n=1}^{N} \prod_{k=1}^{K} p(x_n|z_n)^{z_{nk}} \right]
$$

$$
= \sum_{k=1}^{K} z_{1k} \log \pi_k + \sum_{n=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} z_{n-1,j}z_{nk} \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \log p(x_n|z_n).
$$

• Statistics we need from the E-step are:

$$
\gamma(z_n) = p(z_n|X).
$$

$$
\xi(z_{n-1}, z_n) = p(z_{n-1}, z_n|X).
$$

- transition model
- observation model
Expected Complete Data Log-likelihood

- The complete data log-likelihood takes form:

\[ Q(\theta, \theta^{old}) = \sum_{Z} p(Z|X, \theta^{old}) \log p(X, Z|\theta). \]

\[ = \sum_{k=1}^{K} \gamma(z_{1k}) \log \pi_k + \sum_{n=2}^{N} \sum_{k=1}^{K} \sum_{j=1}^{K} \xi(z_{n-1, j}, z_{nk}) \log A_{jk} + \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_{nk} \log p(x_n|z_n). \]

- Hence in the E-step we evaluate:

\[ \gamma(z_n) = p(z_n|X). \]

\[ \xi(z_{n-1}, z_n) = p(z_{n-1}, z_n|X). \]

- In the M-step we optimize Q with respect to parameters: \( \theta = \{\pi, A, \phi\} \).
Parameter Estimation

• Initial state distribution: expected number of times in state k at time 1:

\[ \pi_k^{new} = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}. \]

• Expected # of transitions from state j to k which begin at time n-1:

\[ \xi(z_{n-1,j}, z_{n,k}) = p(z_{n-1,j}, z_{n,k} | X), \]

so the estimated transition probabilities are:

\[ A_{jk}^{new} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{i=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}. \]

• The EM algorithm must be initialized by choosing starting values for \( \pi \) and \( A \).

• Note that any elements of \( \pi \) or \( A \) that initially are set to zero will remain zero in subsequent EM updates.
Parameter Estimation: Emission Model

- For the case of **discrete multinomial observed variables**, the observation model takes form:
  \[
  p(x_n | z_n, \phi) = \prod_{i=1}^{D} \prod_{k=1}^{K} \mu_{ik}^{x_{ni}z_{nk}}.
  \]

- And the corresponding M-step update:
  \[
  \mu_{ik}^{new} = \frac{\sum_{n=1}^{N} \gamma(z_{nk})x_{ni}}{\sum_{n=1}^{N} \gamma(z_{nk})}.
  \]

- For the case of the **Gaussian emission model**:
  \[
  p(x_n | z_n, \phi) = \prod_{k=1}^{K} \mathcal{N}(x_n | \mu_k, \Sigma_k)^{z_{nk}}.
  \]

- And the corresponding M-step updates:
  \[
  \mu_k^{new} = \frac{1}{N_k} \sum_{n} \gamma(z_{nk})x_n, \quad N_k = \sum_{n} \gamma(z_{nk}),
  \]
  \[
  \Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(y_{nk})(x_n - \mu_k)(x_n - \mu_k)^T
  \]
Viterbi Decoding

• The numbers $\gamma(z_n)$ above gave the probability distribution over all states at any time.

• By choosing the state $\gamma^*(z_n)$ with the largest probability at each time, we can make an "average" state path. This is the path with the maximum expected number of correct states.

• To find the single best path, we do Viterbi decoding which is Bellman’s dynamic programming algorithm applied to this problem.

• The recursions look the same, except with max instead of $\sum$.

• Same dynamic programming trick: instead of summing, we keep the term with the highest value at each node.

• There is also a modified EM (Baum-Welch) training based on the Viterbi decoding. Like K-means instead of mixtures of Gaussians.

• Remember the max-sum algorithm for tree structured graphical models.
Viterbi Decoding

• A fragment of the HMM lattice showing two possible paths:

  \[
  k = 1 \quad \square \quad \square \quad \square \quad \square \quad \ldots
  \]
  \[
  k = 2 \quad \square \quad \square \quad \square \quad \square \quad \ldots
  \]
  \[
  k = 3 \quad \square \quad \square \quad \square \quad \square \quad \ldots
  \]

  \[n - 2 \quad n - 1 \quad n \quad n + 1\]

• Viterbi decoding efficiently determines the most probable path from the exponentially many possibilities.

• The probability of each path is given by the product of the elements of the transition matrix \(A_{jk}\), along with the emission probabilities associated with each node in the path.
Using HMMs for Recognition

- We can use HMMs for recognition by:
  - training one HMM for each class (requires labeled training data)
  - evaluating probability of an unknown sequence under each HMM
  - classifying unknown sequence by choosing an HMM with highest likelihood

- This requires the solution of two problems:
  - Given model, evaluate probability of a sequence. (We can do this exactly and efficiently.)
  - Given some training sequences, estimate model parameters. (We can find the local maximum using EM.)
Autoregressive HMMs

- One limitation of the standard HMM is that it is poor at capturing long-range correlations between observations, as these have to be mediated via the first order Markov chain of hidden states.

- Autoregressive HMM: The distribution over $x_n$ depends on a subset of previous observations.

- The number of additional links must be limited to avoid an excessive number of free parameters.

- The graphical model framework motivates a number of different models based on HMMs.
Input-Output HMMs

- Both the emission probabilities and the transition probabilities depend on the values of a sequence of observations $u_1, \ldots, u_N$.

- Model parameters can be efficiently fit using EM, in which the E-step involves forward-backward recursion.
Factorial HMMs

- Example of Factorial HMM comprising of two Markov chains of latent variables:

- Motivation: In order to represent 10 bits of information at a given time step, a standard HMM would need $K=2^{10}=1024$ states.

- Factorial HMMs would use 10 binary chains.

- Much more powerful model.

- The key disadvantage: Exact inference is intractable.

- Observing the $x$ variables introduces dependencies between latent chains.

- Hence E-step for this model does not correspond to running forward-backward along the M latent chain independently.
Factorial HMMs

- The conditional independence property: \( z_{n+1} \perp z_{n-1} \mid z_n \) does not hold for the individual latent chains.

- There is no efficient exact E-step for this model.

- One solution would be to use MCMC techniques to obtain approximate sample from the posterior.

- Another alternative is to resort to variational inference.

- The variational distribution can be described by M separate Markov chains corresponding to the latent chains in the original model (structured mean-field approximation).
Regularizing HMMs

- There are two problems:
  - for high dimensional outputs, lots of parameters in the emission model
  - with many states, transition matrix has many (squared) elements

- First problem: full covariance matrices in high dimensions or discrete symbol models with many symbols have lots of parameters. To estimate these accurately requires a lot of training data.

- We can use mixtures of diagonal covariance Gaussians.

- For discrete data, we can use mixtures of base rates.

- We can also tie parameters across states.
Regularizing Transition Matrices

• One way to regularize large transition matrices is to constrain them to be sparse: instead of being allowed to transition to any other state, each state has only a few possible successor states.

• A very effective way to constrain the transitions is to order the states in the HMM and allow transitions only to states that come later in the ordering.

• Such models are known as “linear HMMs”, “chain HMMs” or “left- to-right HMMs”. Transition matrix is upper- diagonal (usually only has a few bands).
Linear Dynamical Systems

• In HMMs, latent variables are discrete but with arbitrary emission probability distributions.

We now consider a linear-Gaussian state-space model, so that latent variables and observed variables are multivariate Gaussian distributions.

• An HMM can be viewed as an extension of the mixture models to allow for sequential correlations in the data.

• Similarly, the linear dynamical system (LDS) can be viewed as a generalization of the continuous latent variable models, such as probabilistic PCA.
Linear Dynamical Systems

- The model is represented by a tree-structured directed graph, so inference can be solved efficiently using the sum-product algorithm.

- The forward recursions, analogous to the $\alpha$-messages of HMMs are known as the Kalman filter equations.

- The backward recursions, analogous to the $\beta$-messages, are known as the Kalman smoother equations.

- The Kalman filter is used in many real-time tracking applications.

- Because the LDS is a linear-Gaussian model, the joint distribution over all variables, as well as marginals and conditionals, will be Gaussian.

- This leads to tractable inference and learning.
The Model

- We can write the transition and emission distributions in the general form:

\[
p(z_n | z_{n-1}) = \mathcal{N}(z_n | Az_{n-1}, \Gamma),
\]

\[
p(x_n | z_n) = \mathcal{N}(x_n | Cz_n, \Sigma),
\]

\[
p(z_1) = \mathcal{N}(z_1 | \mu_0, V_0).
\]

- These can be expressed in terms of noisy linear equations:

\[
z_n = Az_{n-1} + w_n, \quad w \sim \mathcal{N}(w | 0, \Gamma),
\]

\[
x_n = Cz_n + v_n, \quad v \sim \mathcal{N}(v | 0, \Sigma),
\]

\[
z_1 = \mu_0 + u, \quad u \sim \mathcal{N}(u | 0, V_0).
\]

- Model parameters \( \theta = \{A, \Gamma, C, \Sigma, \mu_0, V_0\} \) can be learned using EM algorithm (similar to standard HMM case).
Inference in LDS

- Consider forward equations. The initial message is Gaussian, and since each of the factors is Gaussian, all subsequent messages will also be Gaussians.

\[ z_n = A z_{n-1} + w_n, \quad w \sim \mathcal{N}(w|0, \Gamma), \]
\[ x_n = C z_n + v_n, \quad v \sim \mathcal{N}(v|0, \Sigma), \]
\[ z_1 = \mu_0 + u, \quad u \sim \mathcal{N}(u|0, \Sigma_0). \]

- Similar to HMMs, let us define the normalized version of \( \alpha(z_n) \):

\[
\hat{\alpha}(z_n) = p(z_n | x_1, \ldots, x_n) = \frac{\alpha(z_n)}{p(x_1, \ldots, x_n)} = \mathcal{N}(z_n | \mu_n, V_n).
\]

Remember: for HMMs

\[
\alpha(z_n) = p(x_n | z_n) \sum_{z_{n-1}} \alpha(z_{n-1}) p(z_n | z_{n-1})
\]

- Using forward recursion, we get:

\[
c_n \hat{\alpha}(z_n) = p(x_n | z_n) \int \hat{\alpha}(z_{n-1}) p(z_n | z_{n-1}) dz_{n-1}
\]
\[
= \mathcal{N}(x_n | C z_n, \Sigma) \int \mathcal{N}(z_{n-1} | \mu_{n-1}, V_{n-1}) \mathcal{N}(z_n | A z_{n-1}, \Gamma) dz_{n-1}.
\]
Inference in LDS

• Hence we obtain:

\[ c_n \hat{\alpha}(z_n) = p(x_n|z_n) \int \hat{\alpha}(z_{n-1}) p(z_n|z_{n-1}) dz_{n-1} \]

\[ = \mathcal{N}(x_n|Cz_n, \Sigma) \int \mathcal{N}(z_{n-1}|\mu_{n-1}, V_{n-1}) \mathcal{N}(z_n|Az_{n-1}, \Gamma) dz_{n-1}. \]

in which case \( \alpha(z_n) \) is Gaussian:

\[ c_n \hat{\alpha}(z_n) = \mathcal{N}(z_n|\mu_n, V_n). \]

\[
\begin{align*}
\mu_n &= A\mu_{n-1} + K_n(x_n - CA\mu_{n-1}) \\
V_n &= (I - K_n C)P_{n-1} \\
P_{n-1} &= AV_{n-1}A^T + \Gamma.
\end{align*}
\]

and we have also defined the Kalman gain matrix:

\[ K_n = P_{n-1}C^T \left(CP_{n-1}C^T + \Sigma\right)^{-1}. \]
Kalman Filter

• Let us examine the evolution of the mean:

\[ z_n = A z_{n-1} + w_n, \quad w \sim \mathcal{N}(w|0, \Gamma), \]
\[ x_n = C z_n + v_n, \quad v \sim \mathcal{N}(v|0, \Sigma), \]
\[ z_1 = \mu_0 + u, \quad u \sim \mathcal{N}(u|0, V_0). \]

Prediction of the mean over \( z_n \).

Predicted observation for \( x_n \).

\[ \mu_n = A \mu_{n-1} + K_n (x_n - CA \mu_{n-1}). \]

Predicted mean plus the correction term controlled by the Kalman gain matrix.

Error between the predicted observation \( x_n \) and the actual observation \( x_n \).

• We can view the Kalman filter as a process of making subsequent predictions and then correcting these predictions in the light of the new observations.
Kalman Filter

- Example:

\[
p(z_{n-1} | x_1, \ldots, x_{n-1})
\]
blue curve

\[
p(z_n | x_1, \ldots, x_{n-1})
\]
red curve

\[
p(z_n | x_1, \ldots, x_n)
\]
blue curve

incorporate transition model

incorporate new observation
(density of the new point is given by the green curve)

- The new observation has shifted and narrowed the distribution compared to (see red curve) \( p(z_n | x_1, \ldots, x_{n-1}) \).
Tracking Example

- LDS that is being used to track a moving object in 2-D space:

  - **Blue points** indicate the true position of the object.
  - **Green points** denote the noisy measurements.
  - **Red crosses** indicate the means of the inferred posterior distribution of the positions inferred by the Kalman filter.
Particle Filters

- For dynamical systems that are non-Gaussian (e.g. emission densities are non-Gaussian), we can use sampling methods to find a tractable solution to the inference problem.

- Consider a class of distributions represented by the graphical model:

- Suppose we observed $X_n = \{x_1, \ldots, x_n\}$, and we wish to approximate:

\[
\mathbb{E}[f(z_n)] = \int f(z_n)p(z_n|X_n)dz_n
\]

\[
= \int f(z_n)p(z_n|x_n, X_{n-1})dz_n
\]

\[
= \frac{\int f(z_n)p(x_n|z_n)p(z_n|X_{n-1})dz_n}{\int p(x_n|z_n)p(z_n|X_{n-1})dz_n} \approx \sum_{l=1}^{L} w_n^{(l)} f(z_n^{(l)}),
\]
Particle Filters

- Hence

\[ \mathbb{E}[f(z_n)] = \int f(z_n)p(z_n|X_n)dz_n \approx \sum_{l=1}^{L} w_n^{(l)} f(z_n^{(l)}), \quad z_n^{(l)} \sim p(z_n|X_{n-1}), \]

with importance weights:

\[ w_n^{(l)} = \frac{p(x_n|z_n^{(l)})}{\sum_{m=1}^{L} p(x_n|z_n^{(m)})}. \]

- Hence the posterior \( p(z_n|X_n) \) is represented by the set of \( L \) samples together with the corresponding importance weights.

- We would like to define a sequential algorithm.

- Suppose that a set of samples and weights have been obtained at time step \( n \).

- We wish to find the set of new samples and weights at time step \( n+1 \).
Particle Filters

• From our previous result, let $f(z_n) = p(z_{n+1}|z_n)$,

$$
p(z_{n+1}|X_n) = \int p(z_{n+1}|z_n)p(z_n|X_n)dz_n
\approx \sum_{l=1}^{L} w_n^{(l)} p(z_{n+1}|z_n^{(l)}), \quad z_n^{(l)} \sim p(z_n|X_{n-1}),
$$

$$
w_n^{(l)} = \frac{p(x_n|z_n^{(l)})}{\sum_{m=1}^{L} p(x_n|z_n^{(m)})}.
$$

• Summary of the particle filter algorithm:

- At time $n$, we have a sample representation of the posterior distribution $p(z_n | X_n)$ expressed as $L$ samples with corresponding weights.

- We next draw $L$ samples from the mixture distribution (above).

- For each sample, we then use the new observation to re-evaluate the weights:

$$
w_{n+1}^{(l)} = \frac{p(x_{n+1}|z_{n+1}^{(l)})}{\sum_{m=1}^{L} p(x_{n+1}|z_{n+1}^{(m)})}.
$$
• At time $n$, the posterior $p(z_n \mid X_n)$ is represented as a mixture distribution.
• We draw a set of $L$ samples from this distribution (incorporating the transition model).
• The new weights evaluated by incorporating the new observation $x_{n+1}$. 