CS281: Advanced ML **September 11, 2017**

Lecture 10: Exact Inference: Time Series

Lecturer: Sasha Rush *Scribes: Max Hopkins, Sebastian Wagner-Carena, Mien Wang, Jamila Pegues*

10.1 Prelude

10.1.1 Notation

Recall from Lecture 9 our notation for the joint probability distributions of *Undirected Graphical Models* (UGMs). In particular, we have

$$p(x_1, \ldots, x_T) = exp\{\sum_c \theta_c(x_c) - A(\theta)\}$$

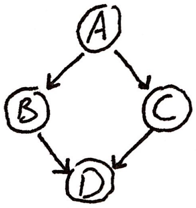
Here, $\theta_c(x_c)$ is the score associated with some clique c, and x_c is some value assignment on the clique. This notation is great because it corresponds to exponential families! However, there are a few other notations you may see around:

$$\begin{aligned} p(x_1, \ldots, x_T) &\propto \prod_c exp(\theta_c(x_c)) \\ &= \prod_c \psi_c(x_c)) \end{aligned}$$

This latter notation is used by Murphy. $\psi_c(x_c)$ are the potentials, and they are simply $\exp(\theta_c(x_c)$. The score functions θ_c are then known as the log potentials.

10.1.2 Moralization

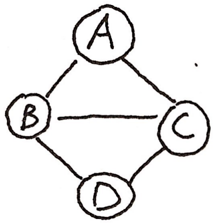
Recall from last lecture the process of conversion between a *Directed Graphical Model* (DGM) and UGM, known as moralization. Here we consider two diagrams:



The first has joint probability distribution

$$P(A, B, C, D) = P(A)P(B|A)P(C|A)P(D|B, C)$$

After moralization, we have



We see that the UGM has two cliques, the $c_0 = \{A, B, C\}$, and $c_1 = \{B, C, D\}$. Using the notation above, this gives us

$$P(A, B, C, D) = \psi_{c_0}(A, B, C)\psi_{c_1}(B, C, D)$$

This form gives a particularly nice parallelism where it is clear that the first three terms

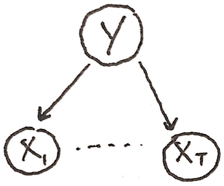
$$P(A)P(B|A)P(C|A) = \psi_{c_0}(A, B, C)$$

and

$$P(D|B, C) = \psi_{c_1}(B, C, D)$$

10.1.3 Conditional Independence vs. Parametrization

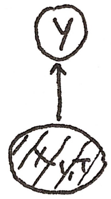
It is important to keep in mind that DGMs only specify conditional independence. The same DGM's could correspond to completely different parametrizations of the random variables. For example, consider Naive Bayes:



In Lecture 5, we discussed all kinds of Naive Bayes (NB), including Bernoulli, MVN, Categorical, and more. All of these follow the diagram above, which only specifies the conditional probability distribution

$$p(x|y) = p(y)\prod_i p(x_i|y)$$

Now consider a conditional model:

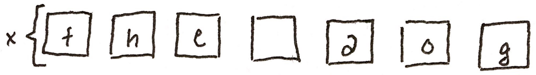


Here we are only interested in $p(y|x_1, \ldots, x_T)$. Similarly to NB, the parametrizations of this model can take many forms. For instance, one could use logistic or linear regression, most GLM's, Neural Networks, or even convolutional Neural Networks. In fact this is the model on which Alphago functions, where features x are the tile placements on the board, and y is the output move!
The main point here is that we can stick arbitrary parametrizations into graphical models. These diagrams just specify the conditional dependence–not the distributions. Further, as long as we specify the graphical model structure, we will be able to tell how hard inference will be.

10.2 Time Series

In this lecture, we only consider an informal definition of a time series. This structure is marked by collecting data over time, and predicting an output for each data input. We are going to consider a special case, labeling a time series. Here our we will have $x_{1:T}$ as our input sequence (features), and $y_{1:T}$ as our output labels. We will begin by providing examples of labeling time series:

Example 1 (OCR)**.** Here we have blocks of pixels, discrete vectors, each which depict a letter. These blocks are our x_i, and each corresponding y_i the discrete symbol the x_i represents

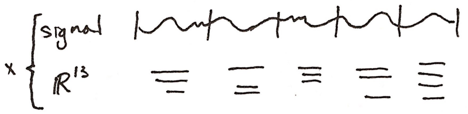


y = {t,h,e, ,d,o,g}

Example 2 (NLP)**.** Here we are given discete words as input variables x, and we wish to predict discrete y, their parts of speech

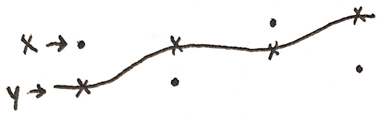
x: the dog ate a carrot
y: DT NOUN VERB DT NOUN

Example 3 (Speech Recognition)**.** Recall from previous lectures that our signal may be divided up into time steps and translated to continues vectors in $\mathbb{R}^{13}$, these our are continuous input variables x. Our output for each vector is the phoneme corresponding to the sound.

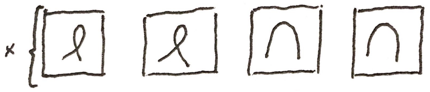


y = {d,d,d,o,o}

Example 4 (Tracking)**.** Here we are tracking the position of some object with presumed Gaussian noise. In this case both our input and output variables are continuous. The inputs are our position vectors, and the output is the predicted correction for noise.

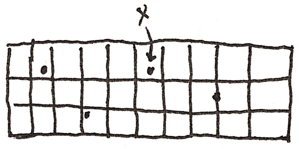


Example 5 (Education)**.** This example is based upon the presentation at the beginning of the class. Our inputs are given by a kinect tracker and are continuous body positions at snapshots in time. The discrete output y is whether the body position corresponds to attentive or bored.



y = {attentive, attentive, bored, bored }

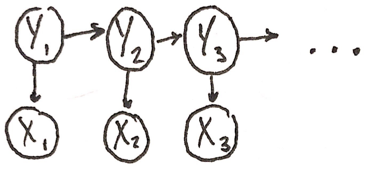
Example 6 (Touch-typing)**.** We consider typing on an iphone, where inputs are the continuous position of keystrokes on the phone. The outputs y are given by discrete letters.



y = {s,t,o,p}

10.3 Markov Models

10.3.1 Hidden Markov Model



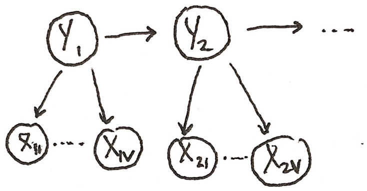
We discussed several *Markov Models* in previous lectures. We begin with the *Hidden Markov Model* (HMM), which actually predates DGMs. HMMs assume discrete y, though DGMs with the same structure may have continuous y. This has a fully joint parametrization $p(y_{1:T}, x_{1:T})$, where in most cases $p(y_t|y_{t-1})$ is categorical. We can choose the distribution of $p(x_i|y_i)$ to fit our given circumstances:

1. $p(x_i|y_i)$ is categorical [e.g. parts of speech]

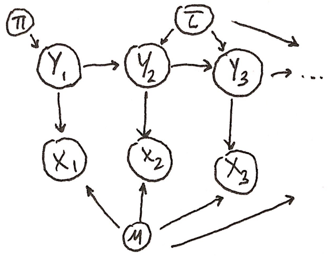
2. $p(x_i|y_i)$ is MVN [typing, speech (this uses a mixed gaussian)]

3. $p(x_{i1}, \ldots, x_{iv}|y_i) = \prod_v p(x_{iv}|y_i)$ [OCR]

The last example here has an embedded Naive Bayes model for each feature x.



HMMs such as the above are often parametrized in the following manner:

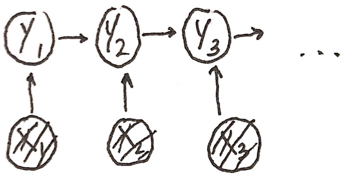


10.3.2 State Space Model

While continuous y has the same graphical model, it has completely different usage and was developed completely separately. This model is called the *State Space Model* (SSM). This model is often used when we have a continuous signal disturbed by gaussian noise–this becomes multivariate and we can compute inference nicely.

10.3.3 Maximum Entropy Markov Model

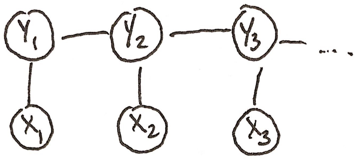
Yet another Markov Model is the *Maxent Markov Model* (MEMM). This model assumes we have observed all features, and flips the direction of the conditioning in the vertical direction.



Thus we are interested in $p(y_1, \ldots, y_T | x_1, \ldots, x_T)$, and $p(y_t | x_t, y_{t-1})$ is a GLM such as logistic or softmax regression.
MEMM comes with the distinct advantage that we no longer have to assume our features are independent. This is particularly useful in, say, tagging parts of speech where can pick an arbitrary feature basis without worrying about independence. However, the model comes with the downside that there is no closed form, so we must use SGD, i.e. we isolate each x_i and predict y_i with logistic regression.
Picking $p(y_t | x_t, y_{t-1})$ here as an arbitrary neural network is called a Neural Network Markov Model or NN-Markov Model.

10.4 Conditional Random Field Markov Model

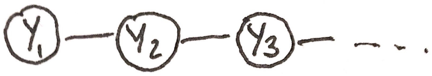
While CRF is still a Markov Model, it gets its own subsection due to its importance. The CRF is simply the UGM of all the above Markov Models:



Recalling Subsection 10.1.1, we write

$$p(y_{1:T} | x_{1:T} = exp\{\sum_t (\theta_t^h(y_t, y_{t-1}) + \theta_t^o(y_t, x_t)) - A(\theta)\}$$

Here h and o refer to the labeled arrows in the diagram. This is the general form of any Markov Model. Note that because we have observed the features $x_{1:T}$, we may rewrite the $T_t^o(y_t, x_t)$ terms as $\theta_t^o(y_t; x_t)$. If we are trying to do inference, we can think of the above after conditioning as simply



This is a simple *Markov Chain* UGM. In fact after converting to UGM and conditioning on our features, all Markov models become the above!
Now while we can compute the closed form of the MLE on HMMs, CRF is a bit tricker. However, it is a member of the exponential family model, and we know the MLE for these families in general.

$$\mathbb{E}(\phi(x)) = \frac{\sum \phi(x_d)}{N}$$

Here, $\phi(y)$ are the sufficient statistics–but what do these look like? In fact we went over this in a previous lecture, it is simply a vector of indicator functions for every clique assignment (see Lecture 9). For inference in particular, we care about

$$\mathbb{E}(\mathbf{1}(x_c = v))$$

Then the clique marginals are given by

$$\frac{\sum\limits_{x'=x'_c=v} p(x')}{\sum\limits_{x''} p(x'')}$$

However this may be computationally intractable, as the x'' we sum over is the entire universe! However, this can be computed efficiently for some models such as chain models

10.5 Bonus: Lecture by Bertrand Schnieder

Bertrand Schnieder gave a snazzy guest lecture on his research, and advertised that the datasets from his research would be ideal as a base for CS 281 final projects. We take a moment to review his lecture below. Schnieder was very interested in studying patterns across concepts of collaborative learning. For example, joint attention refers to when a group of individuals are focused on the same physical object, and is an important part of language development. Signs of joint attention can be seen in the physical movement of the individuals involved, such as their eye movements, gestures, and body postures.

Schnieder discussed studies that he and others carried out to explore joint attention. For example, one study gave forty-two pairs of two participants a task to solve, within a total of forty-five minutes of time. Over that time period, Schnieder et al. used high-frequency and multi-modal sensors to track different aspects of movement, including: video, audio, eye movement (tracked at 60Hz), physiological data (like heart rate; tracked at 1-30Hz), and even body posing and coordination (using a Kinect; tracked at 30Hz). Schnieder pointed out that these sorts of studies provide massive amounts of data. During the lecture, he highlighted that there are a number of cool CS 281 projects that could arise from datasets like this one. Some projects he proposed are:

1. Unsupervised machine learning: modeling collaborative learning processes with probabilistic graphical models.

2. Supervised machine learning: training models to make predictions. For instance, training a model to predict the joint visual attention of two people based on their gestures, head orientation, and speech. Another example is training a model to predict physiological activity based on features like pupil size and body posture. With supervised machine learning, however, Schnieder noted that overfitting can be a dangerous pitfall.

3. Design your own!

Schnieder emphasized that he is open to ideas, and that he highly encourages anyone interested in working with this data in some way or form to email him at bertrand_schneider@gse.harvard.edu.

10.6 Practice Problem: EM for a GMM HMM

Consider a HMM model where the observations are described by a mixture of Gaussians:

$$p(\boldsymbol{x}_t | z_t = z_j, \boldsymbol{\theta}) = \sum_k w_{jk} \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{\mu}_{jk}, \Sigma_{jk})$$

Where w_{jk} is the k^{th} mixing weight at time t in an unobserved state $z_t = j$. So, each of the J possible z states describe an observation with a unique GMM of K gaussians/mixing weights. If the latent states are unobservable, using direct MLE or MAP to infer parameters is intractable, and EM becomes an attractive solution.

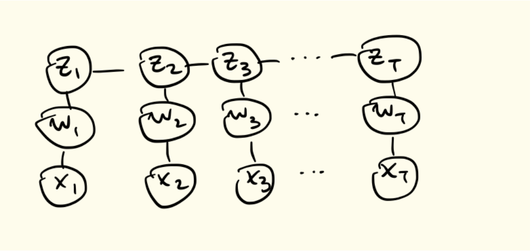
1. Draw the graphical model

2. Write the E step

3. Write the M step

Solutions

1. Graphical Model:



This model contains two layers of latent variables: the time states, $\boldsymbol{z}$, and the Gaussian mixing weight associated with each state, $\boldsymbol{w}$. Both contribute to the likelihood of the observations, $\boldsymbol{x}$

As you might have guessed, the key to this problem is a combination of EM for GMMs as described in Murphy 11.4, and EM for HMMs as described in Murphy 17.5

Describe this HMM using the following notation:

Transition Matrix, A:

$$A_{i,j} = p(z_t = j | z_{t-1} = i)$$

Observation Probability, b:

$$b_{z_t} = p(\boldsymbol{x}_t | z_t = j, \boldsymbol{\theta}) = \sum_k w_{jk} \mathcal{N}(\boldsymbol{x}_t | \mu_{jk}, \Sigma_{jk})$$

Initial Probability, π_y:

$$\pi_y = p(z_1 = y)$$

2. Expectation Step:
The complete data log likelihood for this model can be represented as:

$$l_c(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(\boldsymbol{x}_i, \boldsymbol{z}_i | \boldsymbol{\theta})$$

Where N is the number of samples of $\boldsymbol{x}$ of length T (e.g. N samples of T windows in audio processing) Without observing $\boldsymbol{w}$ or $\boldsymbol{z}$, maximizing the likelihood directly is intractable. Instead, we maximize the expected complete data log likelihood:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) = \mathbb{E}_{\boldsymbol{z}} \left[l_c(\boldsymbol{\theta}) | D, \boldsymbol{\theta}^{t-1} \right] = \sum_{\boldsymbol{z}} l_c(\boldsymbol{\theta}) p(\boldsymbol{z} | D, \boldsymbol{\theta}^{t-1})$$

Without loss of generality, consider only maximizing Q for one of the N time-sequences, Q_i, a chain of T nodes.

$$Q_i(\boldsymbol{\theta}, \boldsymbol{\theta}^{t-1}) = \sum_{\boldsymbol{z}_i} \log \Big(p(\boldsymbol{x}_i, \boldsymbol{z}_i | \boldsymbol{\theta}) \Big) p(\boldsymbol{z}_i | \boldsymbol{x}_i, \boldsymbol{\theta}^{t-1})$$

Where,

$$p(\boldsymbol{x}_i, \boldsymbol{z}_i | \boldsymbol{\theta}) = \pi_y b_{z_1} \prod_{t=2}^{T} A_{z_{t-1}, z_t} b_{z_t}$$

Separating out terms, Q_i becomes:

$$\begin{aligned} Q_i(\boldsymbol{\theta},\boldsymbol{\theta}^{t-1}) = & \sum_{\boldsymbol{z}_i} \log\left(\pi_y\right) p(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{\theta}^{t-1}) + \\ & \sum_{\boldsymbol{z}_i}\left(\sum_{t=2}^{T}\log(A_{z_{t-1},z_t})\right)p(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{\theta}^{t-1}) + \\ & \sum_{\boldsymbol{z}_i}\left(\sum_{t=1}^{T}\log b_{z_t}\right)p(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{\theta}^{t-1}) \end{aligned}$$

3. Maximization Step (see Murphy 17.5):
The strategy of optimizing the parameters of the first two terms the same as it is for any Markov chain: Simplifying the above summation, the first term becomes:

$$\sum_{y\in Z}\log(\pi_y)p(z_1 = y|\boldsymbol{x_i},\boldsymbol{\theta}^{t-1})$$

This can be optimized across N samples by choosing

$$\pi_y = \frac{1}{N}\sum_{i=0}^{N-1} p(z_{i1} = y|\boldsymbol{x_i},\boldsymbol{\theta}^{t-1})$$

The second term becomes:

$$\sum_{j\in Z}\sum_{s\in Z}\sum_{t=2}^{T}\log(A_{ij})p(z_{i,t-1} = j, z_{i,t} = s|\boldsymbol{x_i},\boldsymbol{\theta}^{t-1})$$

This can be maximized across N samples by choosing

$$A_{ij} = \frac{\sum_{i=0}^{N-1}\sum_{t=2}^{T} p(z_{i,t-1} = i, z_{i,t} = j|\boldsymbol{x_i},\boldsymbol{\theta}^{t-1})}{\sum_{i=0}^{N-1}\sum_{t=2}^{T} p(z_{i,t-1} = i|\boldsymbol{x_i},\boldsymbol{\theta}^{t-1})}$$

The final term becomes:

$$\sum_{j=1}^{|Z|}\sum_{k=1}^{|W|}\sum_{t=1}^{T}\log\left(w_{jk}\mathcal{N}(\boldsymbol{x}_{i,t}|\mu_{jk},\Sigma_{jk})\right)p(z_t = j|\boldsymbol{x}_i,\boldsymbol{\theta}^{t-1}))$$

Using Murphy's notation for EM with GMM's (11.4), the responsibility of a given cluster k for a given data point i, r_{ik} is the posterior probability of $\boldsymbol{x}_i$ belonging to cluster k:

$$r_{ik} = p(w_i = w_k|\boldsymbol{x}_i,\boldsymbol{\theta}^{t-1})$$

In this model, the responsibility must incorporate the joint probability of a cluster weight, $w_{jk,t}$ *and* the latent z_t

$$r_{jk,t} = p(z_t = z_j, w_t = w_{jk}|\boldsymbol{x}_i,\boldsymbol{\theta}^{t-1})$$

The rest of the optimization procedure is analogous to Murphy's description of EM in GMM's (11.4)
Let $L = |Z| \times |W|$. The L mixing weights can be optimized across N samples by choosing

$$w_{jk} = \frac{\sum_{i=1}^{N-1}\sum_{t=1}^{T} r_{ijk,t}}{\sum_{i=1}^{N-1}\sum_{t=1}^{T}\sum_{k=1}^{|W|} r_{ijk,t}}$$

The L MVN means can be optimized across N samples by choosing:

$$\boldsymbol{\mu}_{jk} = \frac{\sum_{i=1}^{N-1} \sum_{t=1}^{T} r_{ijk,t} \boldsymbol{x}_{i,t}}{\sum_{i=1}^{N-1} \sum_{t=1}^{T} r_{ijk,t}}$$

The L MVN covariance matrices can be optimized across N samples by choosing:

$$\Sigma_{jk} = \frac{\sum_{i=1}^{N-1} \sum_{t=1}^{T} r_{ijk,t} (\boldsymbol{x}_{i,t} - \boldsymbol{\mu}_{jk})(\boldsymbol{x}_{i,t} - \boldsymbol{\mu}_{jk})^T}{\sum_{i=1}^{N-1} \sum_{t=1}^{T} r_{ijk,t}}$$