CSC 412/2506: Probabilistic Learning and Reasoning

Week 6: HMMs and Variational Inference I

Denys Linkov

University of Toronto

Overview: part 1

- Hidden Markov Models
- Forward / Backward Algorithm
- Viterbi Algorithm

Sequential data

We generally assume data was i.i.d, however this may be a poor assumption:

- Sequential data is common in time-series modelling (e.g. stock prices, speech, video analysis) or ordered (e.g. textual data, gene sequences).
- Recall the general joint factorization via the chain rule

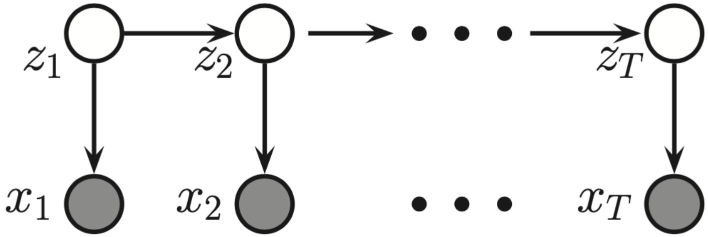
$$p(x_{1:T}) = \prod_{t=1}^{T} p(x_t|x_{t-1}, ..., x_1) \text{ where } p(x_1|x_0) = p(x_1).$$

- But this quickly becomes intractable for high-dimensional data -each factor requires exponentially many parameters to specify as a function of T.
- So we **made** the simplifying assumption that our data can be modeled as a **first-order Markov chain**

$$p(x_t|x_{1:t-1}) = p(x_t|x_{t-1})$$

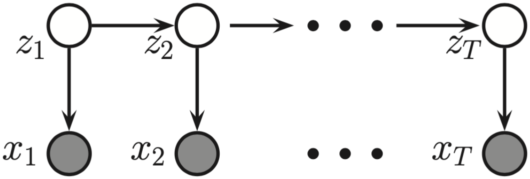
Sequential data

- In certain cases, Markov chain assumption is also restrictive.
- The state of our variables is fully observed. Hence, we introduce Hidden Markov Models



Hidden Markov Models (HMMs)

- HMMs hide the temporal dependence by keeping it in the unobserved state.
- No assumptions on the temporal dependence of observations is made.
- For each observation x_t, we associate a corresponding unobserved hidden/latent variable z_t



- The joint distribution of the model becomes

$$p(x_{1:T}, z_{1:T}) = p(z_1) \prod_{t=2}^{T} p(z_t | z_{t-1}) \prod_{t=1}^{T} p(x_t | z_t)$$

Hidden Markov Models (HMMs)

Unlike simple Markov chains, the observations are not limited by a Markov assumption of any order. Assuming we have a homogeneous model, we only have to know three sets of distributions

1. **Initial distribution**: $\pi(i) = p(z_1 = i)$. The probability of the first hidden variable being in state i (often denoted π)
2. **Transition distribution**: $\Psi(i, j) = p(z_{t+1} = j | z_t = i) \quad i \in \{1, ..., k\}$. The probability of moving from hidden state i to hidden state j.
3. **Emission probability**: $\psi_t(i) = p(x_t | z_t = i)$. The probability of an observed random variable x given the state of the hidden variable that "emitted" it.

HMMs: Objectives

We consider the following objectives:

1. Compute the probability of a latent sequence given an observation sequence.
   That is, we want to be able to compute $p(z_{1:t}|x_{1:t})$. This is achieved with the **Forward-Backward algorithm**.
2. Infer the most likely sequence of hidden states.
   That is, we want to be able to compute

$$z^\star = \underset{z_{1:T}}{\operatorname{argmax}}\, p(z_{1:T}|x_{1:T}).$$

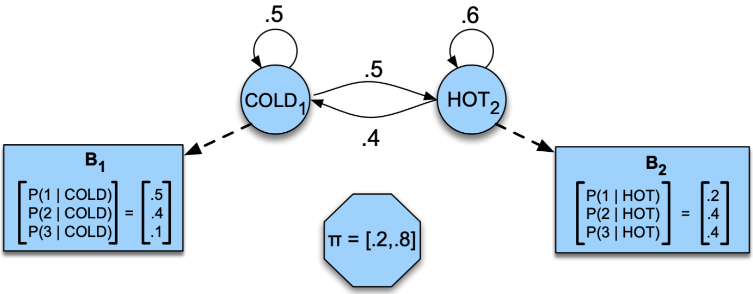
   This is achieved using the **Viterbi algorithm**.
3. You have some HMM, let's do prediction i.e $p(x_{1:t}|z_{1:t})$. Use a forward pass algorithm using our hidden states.

A future anthropologist studies ice-cream eating

- You are a climatologist in the year 2799 studying the history of global warming. You cannot find any records of the weather in Baltimore, Maryland, for the summer of 2020, but you do find Jason Eisner's diary, which lists how many ice creams Jason ate every day that summer. Our goal is to use these observations to estimate the temperature every day. We'll simplify this weather task by assuming there are only two kinds of days: cold (C) and hot (H). (Eisner, 2002)
- Given a sequence of observations x_t of ice cream eaten, find the weather events z_t for each day.
- All graphics from (SLP, 2024)

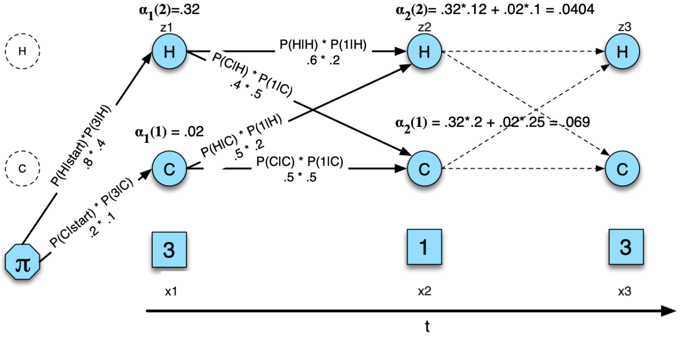
The probabilities

- Let's define our variables
- $x \in \{1, 2, 3\}$ is the number of ice cream's eaten
- $z \in \{hot, cold\}$ is the weather



Most likely outcomes

- Assume we have some model that Jason kept for us. We could compute the joint probabilities $p(x_1, ..., x_t, z_1, ...z_t)$



Forward algorithm

- The goal is to recursively compute the filtered marginals,

$$\alpha_t(j) = p(z_t = j | x_{1:t})$$

  in a HMM,
- Assuming that we know the initial $p(z_1)$, transition $p(z_t|z_{t-1})$, and emission $p(x_t|z_t)$ probabilities $\forall t \in [1, T]$.
- This is a step in the **forward-backward algorithm**, which we'll reference later.

Forward algorithm

The algorithm has two steps:

- First one is the **prediction step**, in which we compute the one-step-ahead predictive density; this acts as the new prior for time t:

$$\begin{aligned} p(z_t = j | x_{1:t-1}) &= \sum_i p(z_t = j | z_{t-1} = i) p(z_{t-1} = i | x_{1:t-1}) \\ &= \sum_i \Psi(i, j) \alpha_{t-1}(i) \end{aligned}$$

- Next one is the **update step**,

$$\begin{aligned} \alpha_t(j) = p(z_t = j | x_{1:t}) &= p(z_t = j | x_{1:t-1}, x_t) \\ &= \frac{p(z_t = j, x_{1:t-1}, x_t)}{p(x_{1:t-1}, x_t)} \\ &= \frac{p(x_t, z_t = j, x_{1:t-1})}{p(x_{1:t-1}, x_t)} \end{aligned}$$

Keep Going with our derivation

- Update step continued

$$\begin{aligned}
\alpha_t(j) &= \frac{p(x_t, z_t = j, x_{1:t-1})}{p(x_{1:t-1}, x_t)} \\
&= \frac{p(x_t|z_t = j, x_{1:t-1})p(z_t = j|x_{1:t-1})}{p(x_{1:t-1}, x_t)} \\
&\propto p(x_t|z_t = j, x_{1:t-1})p(z_t = j|x_{1:t-1}) \\
&\propto p(x_t|z_t = j)p(z_t = j|x_{1:t-1}) \text{ MC assumption} \\
&= \psi_t(j)p(z_t = j|x_{1:t-1})
\end{aligned}$$

- Where the normalizing constant is

$$Z_t = p(x_t|x_{1:t-1}) = \sum_j p(z_t = j|x_{1:t-1})p(x_t|z_t = j)$$

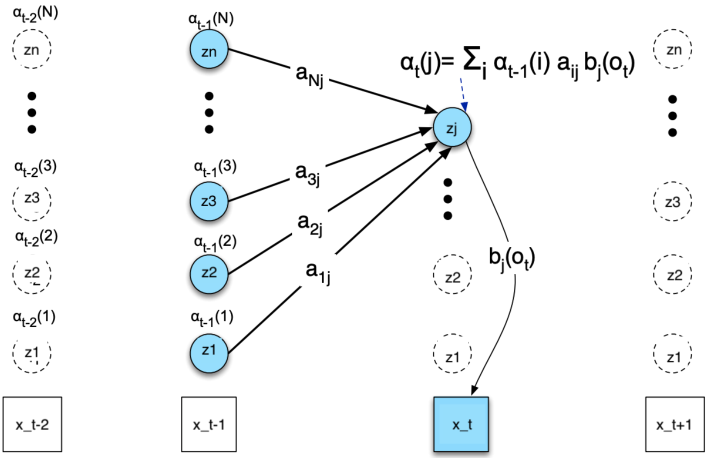
- This process is called the predict-update cycle.
- Using matrix notation, we can write the update in the following simple form:

$$\alpha_t \propto \psi_t \odot (\Psi^T \alpha_{t-1})$$

  where

- $\psi_t(j) = p(x_t|z_t = j)$ is the local evidence at time t,
- $\Psi(i,j) = p(z_t = j|z_{t-1} = i)$ is the transition matrix,
- and $\odot$ is the Hadamard (entrywise) product.

Visualizing Forward Algorithm



**Will update the variables on the slide q=z, o=x

Forward-Backward algorithm

- The Forward-backward algorithm is used to efficiently estimate the latent sequence given an observation sequence under a HMM.
- That is, we want to compute

$$p(z_t|x_{1:T}) \quad \forall_t \in [1, T]$$

assuming that we know the initial $p(z_1)$, transition $p(z_t|z_{t-1})$, and emission $p(x_t|z_t)$ probabilities $\forall t \in [1, T]$.

Forward-Backward algorithm

This task of hidden state inference breaks down into the following:

- **Filtering**: compute posterior over current hidden state, $p(z_t|x_{1:t})$.
- **Prediction**: compute posterior over future hidden state, $p(z_{t+k}|x_{1:t})$.
- **Smoothing**: compute posterior over past hidden state, $p(z_k|x_{1:t}) \quad 1 < k < t$.

The probability of interest, $p(z_t|x_{1:T})$ is computed using a forward and backward recursion

- **Forward Recursion**: $p(z_t|x_{1:t})$
- **Backward Recursion**: $p(x_{1+t:T}|z_t)$

Forward-Backward algorithm

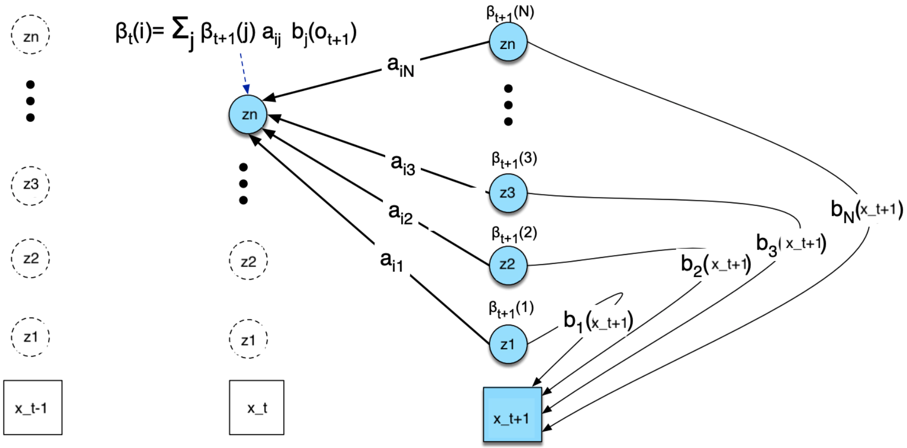
We can break the chain into two parts, the past and the future, by conditioning on z_t:

- We have

$$\begin{aligned}\gamma_t = p(z_t|x_{1:T}) &\propto p(z_t, x_{1:T}) \\ &= p(z_t, x_{1:t})p(x_{t+1:T}|z_t, x_{1:t}) \\ &= p(z_t, x_{1:t})p(x_{t+1:T}|z_t) \\ &\propto \text{(Forward Recursion)(Backward Recursion)}\end{aligned}$$

- The third line is arrived at by noting the conditional independence $x_{t+1:T} \perp x_{1:t}|z_t$.
- We know how to perform forward recursion from the previous part.

Visualizing Backward



Backward recursion

In the backward pass,

$$\begin{aligned}
\beta_t(i) &= p(x_{t+1:T}|z_t = i) \\
&= \sum_j p(z_{t+1} = j, x_{t+1:T}|z_t = i) \\
&= \sum_j p(x_{t+2:T}|z_{t+1} = j, z_t = i, x_{t+1})p(x_{t+1}|z_{t+1} = j, z_t = i)p(z_{t+1} = j|z_t = i) \\
&= \sum_j p(x_{t+2:T}|z_{t+1} = j)p(x_{t+1}|z_{t+1} = j)p(z_{t+1} = j|z_t = i) \\
&= \sum_j \beta_{t+1}(j)\psi_{t+1}(j)\Psi(i, j)
\end{aligned}$$

- Notice that our backward recursion contains our emission, $\psi_{t+1} = p(x_{t+1}|z_{t+1})$ and transition, $\Psi = p(z_{t+1}|z_t)$ probabilities.

Backward recursion

- In vector notation
$$\beta_t = \Psi(\psi_{t+1} \odot \beta_{t+1})$$
where $\beta_T(i) = 1$.
- Once we have the forward and the backward steps complete, we can compute
$$\gamma_t(i) \propto \alpha_t(i)\beta_t(i).$$
which is called the **forward-backward algorithm**.
- Recall
$$\begin{aligned}\gamma_t = p(z_t|x_{1:T}) \propto& p(z_t, x_{1:t})p(x_{t+1:T}|z_t) \\ \propto& (\text{Forward Recursion})(\text{Backward Recursion})\end{aligned}$$

How do we find the most likely path of hidden states?

We can use a graph traversal algorithm! Example from wikipedia
Let's assume we found another one of Jason's journals, about how he's feeling

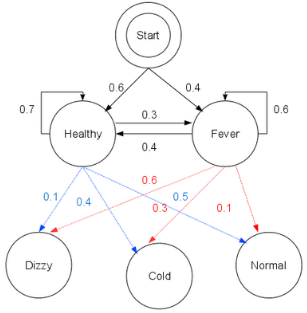


Figure: Define our transition probabilities

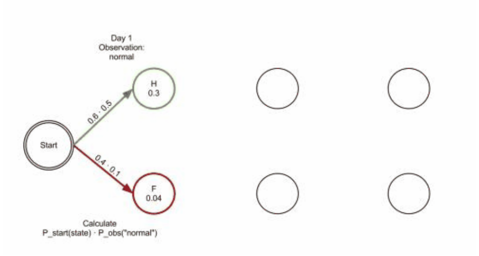
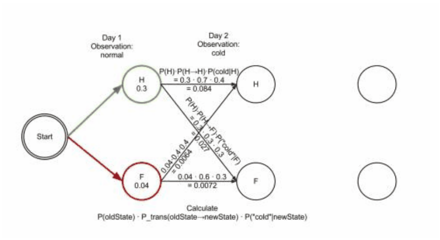


Figure: Day 1

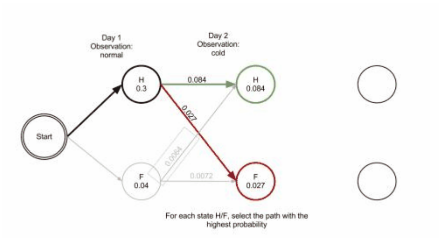
How do we find the most likely path of hidden states?

We can use a graph traversal algorithm! Example from wikipedia



Day 1 Observation: normal
Day 2 Observation: cold
Start
H 0.3
F 0.04
H
F
P(H)·P(H→H)·P(cold|H) = 0.3 · 0.7 · 0.4 = 0.084
P(H)·P(H→F)·P("cold"|F) = 0.3 · 0.3 · 0.3 = 0.027
0.04·0.4·0.4 = 0.0064
0.04 · 0.6 · 0.3 = 0.0072
Calculate P(oldState) · P_trans(oldState→newState) · P("cold"|newState)

Figure: Day 2



Day 1 Observation: normal
Day 2 Observation: cold
Start
H 0.3
F 0.04
H 0.084
F 0.027
0.084
0.027
0.0064
0.0072
For each state H/F, select the path with the highest probability

Figure: Day 2 confirm

How do we find the most likely path of hidden states?

We can use a graph traversal algorithm! Example from wikipedia

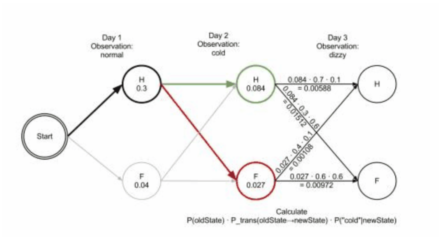


Figure: Day 3

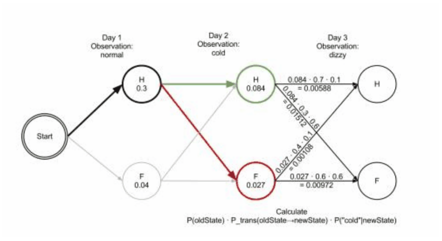


Figure: Day 3 confirm

Formalize this idea - Viterbi algorithm

- The Viterbi algorithm (Viterbi 1967) is used to compute the most probable sequence.

$$\hat{z} = \arg\max_{z_{1:T}} \; p(z_{1:T}|x_{1:T})$$

- Since this is MAP inference, we might think of replacing sum-operators with max-operators, just like we did in sum-product and max-product.
- But this, in general, will lead to incorrect results.
- In Viterbi algorithm, the forward pass does use max- product, but the backwards pass uses a traceback procedure to recover the most probable path.

Viterbi algorithm

- Let's define
$$\delta_t(j) = \max_{z_1,..,z_{t-1}} p(z_{1:t-1}, z_t = j | x_{1:t})$$
which is the probability of ending up in state j at time t, by taking the most probable path.
- We notice that

$$\begin{aligned}\delta_t(j) &= \max_{z_1,..,z_{t-1}} p(z_{1:t-1}, z_t = j | x_{1:t}) \\ &\propto \max_{z_1,..,z_{t-1}} p(z_{1:t-2}, z_{t-1} = i | x_{1:t-1}) p(z_t = j | z_{t-1} = i) p(x_t | z_t = j) \\ &= \max_i \delta_{t-1}(i) \Psi(i,j) \psi_t(j)\end{aligned}$$

- Let's keep track of the most likely previous state,
$$\theta_t(j) = \arg\max_i \delta_{t-1}(i) \Psi(i,j) \psi_t(j).$$

Summary of Viterbi Algorithm

- Dynamic programming algorithm to find the most likely state sequence for a given emission sequence in a Hidden Markov Model.
- Time complexity: $O(n \cdot k^2)$, where n is the length of the emission sequence and k is the number of states. Much more efficient than brute-force $O(k^n)$.
- Applications:
  - Speech recognition
  - Part-of-speech tagging
  - Gene finding
  - and many more...

Summary: HMMs

- HMMs hide the temporal dependence by keeping it in the unobserved state.
- No assumptions on the temporal dependence of observations is made.
- Forward-backward algorithm can be used to find “beliefs”
- Viterbi algorithm can be used to do MAP.
- Next: Variational inference.

Overview: part 2

- Variational Inference
- M-projection
- I-projection
- Naive mean-field approach

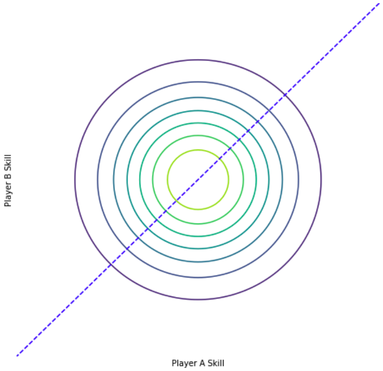
Posterior Inference for Latent Variable Models

We've worked with a few latent variable models, such as the generative image model and the trueskill model.
These models have a factorization $p(x, z) = p(z)p(x|z)$ where

- x are the observations or data,
- z are the unobserved (latent) variables
- $p(z)$ is usually called the **prior**
- $p(x|z)$ is usually called the **likelihood**
- The conditional distribution of the unobserved variables given the observed variables (aka the **posterior**) is

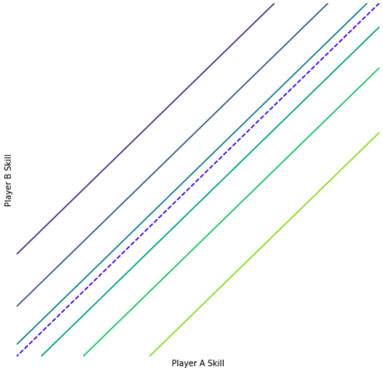
$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x, z)dz}$$

Prior:



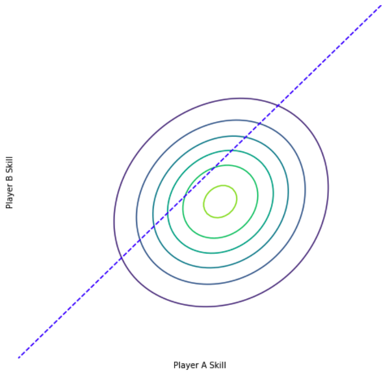
Says we're very uncertain about both player's skill.

Likelihood:



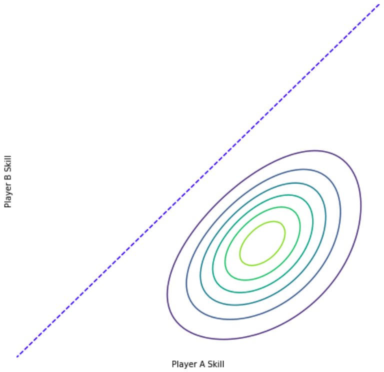
This is the part of the model that gives meaning to the latent variables.

Posterior:



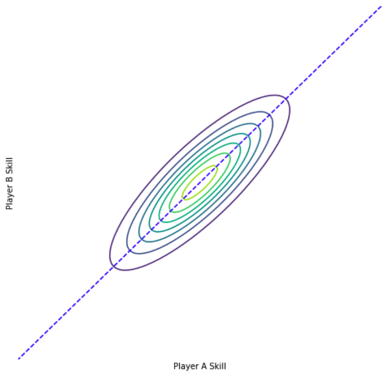
The posterior isn't Gaussian anymore.

Posterior after A beats B 10 times:



Now the posterior is certain that A is better than B.

Posterior after both beat each other 10 times:



Now the posterior is certain that neither player is much better than the other, but is uncertain how good they both are in an absolute sense.

What is hard to compute about the posterior?

- The integral $p(x) = \int p(x,z)dz$ is intractable whenever z is high dimensional. This makes evaluating or sampling from the normalized posterior $p(z|x)$ for a given x and z also intractable.
- Here is a list of operations that are expensive:
  - Computing a posterior probability: $p(z|x) = \frac{p(z)p(x|z)}{p(x)}$
  - Computing the evidence / marginal likelihood $p(x) = \int p(z,x)dz$
    - Useful for choosing between models, or fitting model parameters.
  - Computing marginals of $p(z_1|x) = \int p(z_1, z_2, \dots z_D|x)dz_2, dz_3, \dots dz_D$
    - E.g. finding the posterior over a single tennis player's skill given all games.
  - Sampling $z \sim p(z|x)$
    - Useful for summarizing which hypotheses are likely given the data, making predictions, and decisions.

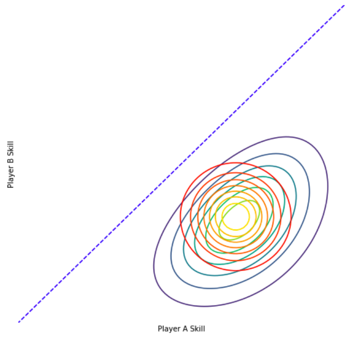
Variational methods

- Variational inference is closely related to the calculus of variations, developed in the 1700s by Euler, Lagrange.
- Variational inference is an approximate inference method where we seek a tractable (e.g., factorized) approximation to the target intractable distribution.

Variational methods

To be more formal, variational inference works as follows:

- Choose a tractable distribution $q(z) \in Q$ from a feasible set Q. This distribution will be used to approximate $p(z|x)$.
    - For example, $q(z) = \mathcal{N}(z|\mu, \Sigma)$. The idea is that we'll try choose a Q that makes $q(z)$ a good approximation of the true posterior $p(z|x)$.
- Encode some notion of "difference" between $p(z|x)$ and q that can be effciently estimated. Usually we will use the KL divergence.
- Minimize this difference. Usually we will use an iterative optimization method.



- Whatever feasible set we choose for Q, it's usually not the case that there is any $q \in Q$ that exactly matches the true posterior.
- But computing the true posterior is intractable, so we have to take a shortcut somewhere.

How to measure closeness: KL divergence

We will measure the difference between q and p using the **Kullback-Leibler divergence**

$$\begin{aligned} KL(q(z)||p(z|x)) &= \int q(z) \log \frac{q(z)}{p(z|x)} dz \\ &= \mathop{\mathbb{E}}_{z\sim q} \log \frac{q(z)}{p(z|x)} \end{aligned}$$

Properties of the KL Divergence

- $KL(q||p) \geq 0$
- $KL(q||p) = 0 \Leftrightarrow q = p$
- $KL(q||p) \neq KL(p||q)$
- KL divergence is not a metric, since it's not symmetric

Which direction of KL to use? $KL(q||p)$ vs $KL(p||q)$

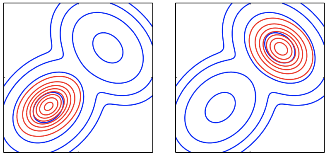
- We could minimize $KL(q||p)$ or $KL(p||q)$
- Which one to choose?
- As always, we will go with the tractable one.

Information (I-)Projection:

I-projection: $q^* = \arg\min_{q \in Q} KL(q||p) = \mathbb{E}_{x \sim q(x)} \log \frac{q(x)}{p(x)}$:

- $p \approx q \implies KL(q||p)$ small
- I-projection underestimates support, and does not yield the correct moments.
- $KL(q||p)$ penalizes q having mass where p has none.

$p(x)$ is mixture of two 2D Gaussians and Q is the set of all 2D Gaussian distributions (with arbitrary covariance matrices)



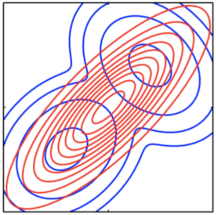
p=Blue, q^*=Red (two equivalently good solutions!)

Moment (M-)projection

M-projection: $q^* = \arg\min_{q \in Q} KL(p||q) = \mathbb{E}_{x \sim p(x)} \log \frac{p(x)}{q(x)}$:

- $p \approx q \implies KL(p||q)$ small
- $KL(p||q)$ penalizes q missing mass where p has some.
- M-projection yields a distribution $q(x)$ with the correct mean and covariance.

$p(x)$ is mixture of two 2D Gaussians and Q is the set of all 2D Gaussian distributions (with arbitrary covariance matrices)



p=Blue, q^*=Red

Maximum entropy interpretation

- A related quantity is the **entropy**:
$$H(p) = -\mathbb{E}_{x\sim p(x)} \log p(x)$$
measuring the uncertainty in the distribution p.
- Consider the optimization problem
$$\begin{aligned} &\text{maximize } H(p) \\ &\text{subject to } \mathbb{E}_{x\sim p(x)}[f_i(x)] = t_i \quad \text{for } i = 1, .., k. \end{aligned}$$
- **Theorem**: Exponential family of distributions maximize the entropy $H(p)$ over all distributions satisfying
$$\mathbb{E}_{x\sim p(x)}[f_i(x)] = t_i \quad \text{for } i = 1, .., k.$$
- In M-projection, if Q is set of exponential families, then the expected sufficient statistics wrt $q^*(x)$ is the same as that wrt $p(x)$.
- M-projection require expectation wrt p, hence intractable.
- Most variational inference algorithms make use of the I-projection.

Mean-field approach

- Say we have an arbitrary MRF:

$$p(x|\theta) = \exp\left\{ \sum_{c\in\mathcal{C}} \phi_c(x_c) - \log Z(\theta) \right\}$$

- We find an approximate distribution $q(x) \in Q$ by performing I-projection to $p(x)$.

$$q^* = \arg\min_{q\in Q} KL(q||p) = \mathbb{E}_{x\sim q(x)} \log \frac{q(x)}{p(x|\theta)}$$

$$\arg\min_{q\in Q} KL(q||p) = \mathbb{E}_{x\sim q(x)} \Big[\log q(x) - \sum_{c\in\mathcal{C}} \phi_c(x_c) + \log Z(\theta) \Big]$$

$$= \arg\max_{q\in Q} \sum_{c\in\mathcal{C}} \mathbb{E}_q[\phi_c(x_c)] + H(q)$$

- For tractability, we need a nice set Q. If $p \in Q$, then $q^* = p$. But this almost never happens.

Naive Mean-Field

- One way to proceed is the mean-field approach where we assume:

$$q(x) = \prod_{i \in V} q_i(x_i)$$

  the set Q is composed of those distributions that factor out.

- Using this in the maximization problem, we can simplify things

$$q^* = \arg\max_{q \in Q} \sum_{c \in \mathcal{C}} \sum_{x_c} q(x_c)\phi_c(x_c) + H(q)$$

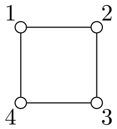
- We notice $q(x_c) = \prod_{i \in c} q_i(x_i)$ and also

$$\begin{aligned}
H(q) =& \mathbb{E}_q[-\log q(x)] = -\sum_x q(x) \log q(x) \\
=& -\sum_x q(x) \Big[\sum_i \log q_i(x_i) \Big] \\
=& -\sum_i \sum_x \Big[q_i(x_i) \log q_i(x_i) \Big] \frac{q(x)}{q_i(x_i)} \\
=& -\sum_i \sum_{x_i} \Big[q_i(x_i) \log q_i(x_i) \Big] \sum_{x \backslash x_i} \frac{q(x)}{q_i(x_i)} \\
=& -\sum_i \sum_{x_i} \Big[q_i(x_i) \log q_i(x_i) \Big] \\
=& \sum_i H(q_i)
\end{aligned}$$

Recall our pairwise MRF

We saw this pairwise 4 cycle in lecture 3

$$p(x_1, x_2, x_3, x_4) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\psi_{3,4}(x_3, x_4)\psi_{1,4}(x_1, x_4)$$



We can write it more generically using our formula for cliques

$$p(\boldsymbol{x}) \;\propto\; \prod_{C \in \mathcal{C}} \psi_C(x_C)$$

Example: Pairwise MRF

- Thus the final optimization problem reduces to

$$q^* = \arg\max_q \sum_{c \in \mathcal{C}} \sum_{x_c} \phi_c(x_c) \prod_{i \in c} q_i(x_i) + \sum_i H(q_i)$$

$$\text{subject to: } q_i(x_i) \geq 0 \text{ and } \sum_{x_i} q_i(x_i) = 1.$$

- Let's further simplify the setting and assume that we have a pairwise MRF. Then the optimization problem becomes

$$q^* = \arg\max_q \sum_{(i,j) \in \mathcal{E}} \sum_{x_i, x_j} \phi_{ij}(x_i, x_j) q_i(x_i) q_j(x_j) - \sum_i \sum_{x_i} q_i(x_i) \log(q_i(x_i))$$

$$\text{subject to: } q_i(x_i) \geq 0 \text{ and } \sum_{x_i} q_i(x_i) = 1.$$

Coordinate maximization

This problem is hard as it has many local maxima! But we can still try to optimize using block coordinate ascent.

- Initialize $\{q_i(x_i)\}_{i \in V}$ uniformly
- Iterate over $i \in V$
  - Greedily maximize the objective over $q_i(x_i)$
  - This is equivalent to: $q_i(x_i) \propto \exp\left\{ \sum_{j \in N(i)} \sum_{x_j} q_j(x_j) \phi_{ij}(x_i, x_j) \right\}$
    - which follows from: write the Lagrangian, take the derivative, set to zero, and solve
  - Repeat until convergence.

This is guaranteed to converge but can converge to local optima.

Summary

- Approximate the complex (intractable) distribution with a simpler (tractable) distribution
- I-projection & M-projection measure the distance to true posterior
- Mean field approximation is a way to simplify the set of distributions
- More variational inference after midterm (which is in 2 weeks).