Pattern Recognition and Machine Learning

EQ2341 VT25

Antoine Honoré
(honore@kth.se)

Plan

Lecture	Topic	Time	Slide
1	HMM + EM	1h30	3
2	EM (continued) + Baum-Welch	1h	33
3	Lagrange multipliers + Baum-Welch (Q&A)	1h	55
4	Bayesian Learning + Variational inference	1h30	64
5	Viterbi + EM (Q&A)	1h	87
6	Transformers	1h30	95
7	VAEs	1h30	107
8	Overall recap	1h30	121
		10h30	

Lecture 1 HMMs

What is a hidden Markov model ?

A parametric statistical model

- For doing what?
  - Modeling an observed data sequence with the assumption that it is related to another unobserved (latent) data sequence.
- With parameter set λ, $\underline{\mathbf{x}} = [\mathbf{x}_1, \dots, \mathbf{x}_T]$ the observed data, $\underline{\mathbf{s}} = [\mathbf{s}_1, \dots, \mathbf{s}_T]$ the un-observed data, the joint model is written:

$$p(\underline{\mathbf{x}}, \underline{\mathbf{s}} | \lambda) \tag{1}$$

- The model does not tell us how to write the likelihood $p(\underline{\mathbf{x}}|\lambda)$ of a data sequence $\underline{\mathbf{x}}$, we have to use the joint distribution:

$$p(\underline{\mathbf{x}}|\lambda) = \int p(\underline{\mathbf{x}}, \underline{\mathbf{s}}|\lambda) d\underline{\mathbf{s}} \tag{2}$$

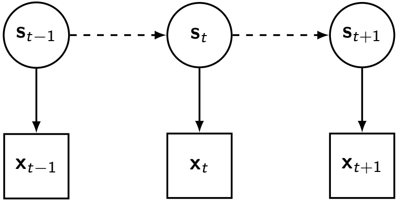
What is a hidden Markov model ?

A parametric statistical model

A HMM makes the following assumption about the relation between the data and the latent:

1. the observed sample at time t depends *only* on the latent variable at time t.
2. the latent variable at time t depends *only* on the latent variable at time $t-1$.

Questions: How do you draw this ? Write the distribution $p(\underline{\mathbf{x}}, \underline{\mathbf{s}})$ according to your drawing



$$p(\underline{\mathbf{x}}, \underline{\mathbf{s}}) = p(\mathbf{s}_1)p(\mathbf{x}_1|\mathbf{s}_1)\prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{s}_t)p(\mathbf{s}_t|\mathbf{s}_{t-1})$$

What is a hidden Markov model ?

A parametric statistical model

$$p(\underline{\mathbf{x}}, \underline{\mathbf{s}}|\lambda) = p(\mathbf{s}_1|\lambda)p(\mathbf{x}_1|\mathbf{s}_1, \lambda)\prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{s}_t, \lambda)p(\mathbf{s}_t|\mathbf{s}_{t-1}, \lambda) \tag{3}$$

- The parameters of the HMM describe how the latent variable is related to the observed data. We choose the parameter space, and learn the best values in that space.
- Here, we treat only HMM with a finite latent space, i.e. the variables $\mathbf{S}_t$ can take $N \in \mathbb{N}^*$ values
  - for $t = 1$ we write: $\mathbf{q} = [p(\mathbf{S}_1 = i)]_{i \in [N]}$
  - for $t > 1$ the kernel $p(\mathbf{S}_t|\mathbf{S}_{t-1})$ can be written as a matrix: $A = [p(\mathbf{S}_t = j|\mathbf{S}_{t-1} = i)]_{i,j \in [N]}$
  - $\forall t$ we write B the parameters of $p(\mathbf{x}_t|\mathbf{s}_t)$
- HMMs defined over sequences of finite length (what we have in practice) are called finite duration. I won't spend time explaining the details of this, refer to 5.3 in the book.

What is a hidden Markov model ?

A parametric statistical model

$$p(\underline{\mathbf{x}}, \underline{\mathbf{s}}|\lambda) = p(\mathbf{s}_1|\lambda)p(\mathbf{x}_1|\mathbf{s}_1, \lambda)\prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{s}_t, \lambda)p(\mathbf{s}_t|\mathbf{s}_{t-1}, \lambda)$$

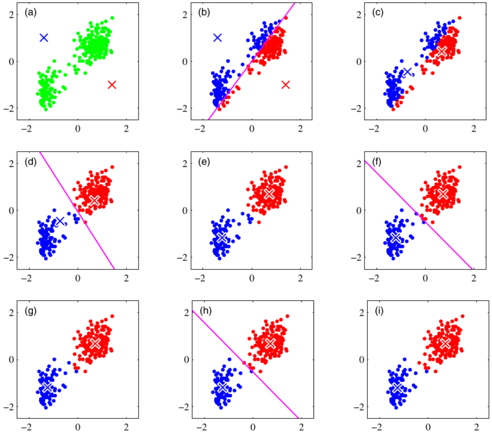
Question: How do you write $p(\mathbf{S}_2)$? What about, $\forall t \in [T], \quad \mathbf{p}_t = p(\mathbf{S}_t)$?

- $\forall j \in [N] \quad p(\mathbf{S}_2 = j) = \sum_{i=1}^{N} p(\mathbf{S}_2 = j|\mathbf{S}_1 = i)p(\mathbf{S}_1 = i)$, so $p(\mathbf{S}_2) = A^T\mathbf{q}$
- $\forall t \in [T] \quad \mathbf{p}_t = A^T\mathbf{p}_{t-1}$

EM: Latent variable models

EM: Latent variable models

Let's forget about time series for a minute, assume data in 2d.



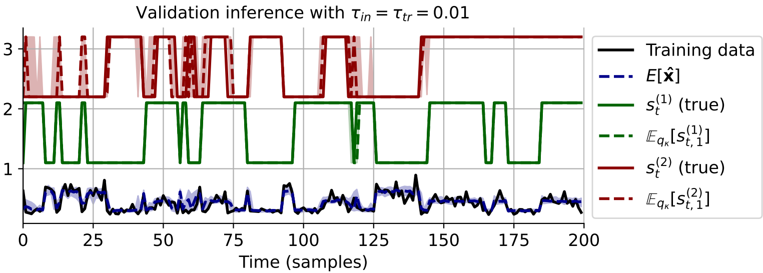
- Do you recognize this algorithm ?
  - it's K-means, a clustering method
- Is this a probabilistic method ?
  - No it's a hard cluster assignment method, BUT there exists an equivalent probabilistic K-means
- Are there latent variables ?
  - yes, the cluster means !
- What are the parameters ?
  - The cluster means too

Bishop 2006 Figure 9.1

EM: Latent variable models

Why choose a latent variable model ?

- Latent variable models are useful when there is a reason to assume an underlying structure in the data (In our case a discrete structure)



Validation inference with $\tau_{in} = \tau_{tr} = 0.01$

EM: Learning Latent variable models

Likelihood maximization ?

- once the structure is specified (the parameter space is chosen), we need to learn the parameters that maximize the likelihood of the data under that model:

$$\lambda^* = \arg\max_{\lambda} p(\underline{\mathbf{x}}|\lambda) = \arg\max_{\lambda} \ln p(\underline{\mathbf{x}}|\lambda) \tag{4}$$

- In other words, find the model with the set of parameters that is the most likely to have generated the data.
- Solving λ^* directly is *intractable*,
  - We would have to integrate/sum over the N^T possible combinations of state sequences:
- Therefore we resort to an iterative scheme to find a locally optimal λ

EM: Learning Latent variable models

Intractable you say ? Let's look at a GMM example

- Given a training set $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$, and a GMM $\mathbf{x}^{(i)} \sim p(\mathbf{x}|\lambda)$, $\lambda = \{(\mu_j, \sigma_j^2, w_j)\}_{j=1}^{K}$
- The log-likelihood is (assuming iid)

$$
\begin{aligned}
L(\lambda) = \sum_{i=1}^{N} \ln p(\mathbf{x}^{(i)}|\lambda) &= \sum_{i=1}^{N} \ln \left[\sum_{j=1}^{K} p(\mathbf{x}^{(i)}, \mathbf{S}^{(i)} = j|\lambda) \right] \\
&= \sum_{i=1}^{N} \ln \left[\sum_{j=1}^{K} p(\mathbf{S}^{(i)} = j) \cdot p(\mathbf{x}^{(i)}|\mathbf{S}^{(i)} = j) \right] \\
&= \sum_{i=1}^{N} \ln \left[\sum_{j=1}^{K} w_j \cdot \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(\mathbf{x}^{(i)} - \mu_j)^2}{2\sigma_j^2} \right) \right]
\end{aligned}
$$

- Solve for λ by setting partial derivatives to 0? No closed-form solution.

EM: Learning Latent variable models

Intractable you say ? Let's look at a Gaussian Mixture model example

- Good news: The optimization is easy if we assume that we know the variable $\mathbf{S}^{(i)}$, $\Longrightarrow$ it becomes a matter of estimating the parameter of single Gaussians.
- Thus we resort to an iterative scheme:
  1. first find the *expected* latent variables
  2. then *maximize* the expected likelihood wrt the parameters
- EM: *Expectation* + *Maximization*
  1. Assume some parameter λ', get the distribution of the latent variable assuming these parameters: $p(\mathbf{S}|\mathbf{X}, \lambda')$.
  2. Maximize $Q(\lambda, \lambda') = E_{p(\mathbf{S}|\mathbf{X}, \lambda')}[\ln p(\mathbf{X}, \mathbf{S}|\lambda)]$ wrt λ.
- Price to pay :
  - In general this iterative scheme does not converge to a global optima, but only to a local optima

EM: Learning Latent variable models

Let's spend some time on the Q function

- In EM what we are really doing is optimizing the Q function, instead of optimizing the log-likelihood directly.
- how do we ensure

$$Q(\lambda, \lambda') > Q(\lambda', \lambda') \implies \ln p(\underline{\mathbf{x}}|\lambda) > \ln p(\underline{\mathbf{x}}|\lambda') \tag{5}$$

- Showing

$$\ln p(\underline{\mathbf{x}}|\lambda) - \ln p(\underline{\mathbf{x}}|\lambda') \geq Q(\lambda, \lambda') - Q(\lambda', \lambda') \tag{6}$$

  is sufficient, why ?
- because then $Q(\lambda, \lambda') - Q(\lambda', \lambda') > 0 \implies \ln p(\mathbf{x}|\lambda) - \ln p(\mathbf{x}|\lambda') > 0$

EM: Learning Latent variable models

Let's spend some time on the Q function

$$
\begin{aligned}
\ln p[\underline{\mathbf{x}} \mid \lambda] - \ln p[\underline{\mathbf{x}} \mid \lambda'] &= \ln \frac{p[\underline{\mathbf{x}} \mid \lambda]}{p[\underline{\mathbf{x}} \mid \lambda']} = \ln \sum_{(i_1 \dots i_T)} \frac{p[\underline{\mathbf{S}} = (i_1 \dots i_T), \underline{\mathbf{x}} \mid \lambda]}{p[\underline{\mathbf{x}} \mid \lambda']} \\
&= \ln \sum_{(i_1 \dots i_T)} \underbrace{\frac{p[\underline{\mathbf{S}} = (i_1 \dots i_T) \mid \underline{\mathbf{x}}, \lambda']}{p[\underline{\mathbf{S}} = (i_1 \dots i_T) \mid \underline{\mathbf{x}}, \lambda']}}_{1} \cdot \frac{p[\underline{\mathbf{S}} = (i_1 \dots i_T), \underline{\mathbf{x}} \mid \lambda]}{p[\underline{\mathbf{x}} \mid \lambda']} \\
&= \ln \sum_{(i_1 \dots i_T)} p[\underline{\mathbf{S}} = (i_1 \dots i_T) \mid \underline{\mathbf{x}}, \lambda'] \frac{p[\underline{\mathbf{S}} = (i_1 \dots i_T), \underline{\mathbf{x}} \mid \lambda]}{p[\underline{\mathbf{S}} = (i_1 \dots i_T), \underline{\mathbf{x}} \mid \lambda']} \\
&= \ln E\left[\frac{p[\underline{\mathbf{S}}, \underline{\mathbf{x}} \mid \lambda]}{p[\underline{\mathbf{S}}, \underline{\mathbf{x}} \mid \lambda']} \middle| \underline{\mathbf{x}}, \lambda' \right]
\end{aligned}
$$

EM: Learning Latent variable models

Let's spend some time on the Q function

$$\begin{aligned}
\ln p[\underline{\mathbf{x}} \mid \lambda] - \ln p[\underline{\mathbf{x}} \mid \lambda'] = \cdots &= \ln E\left[\left.\frac{p[\underline{\mathbf{S}}, \underline{\mathbf{x}} \mid \lambda]}{p[\underline{\mathbf{S}}, \underline{\mathbf{x}} \mid \lambda']}\right| \underline{\mathbf{x}}, \lambda'\right] \\
&\text{(Jensen inequality)} \\
&\geq E\left[\ln \left.\frac{p[\underline{\mathbf{S}}, \underline{\mathbf{x}} \mid \lambda]}{p[\underline{\mathbf{S}}, \underline{\mathbf{x}} \mid \lambda']}\right| \underline{\mathbf{x}}, \lambda'\right] \\
&= E\left[\ln p[\underline{\mathbf{S}}, \underline{\mathbf{x}} \mid \lambda] \mid \underline{\mathbf{x}}, \lambda'\right] - E\left[\ln p[\underline{\mathbf{S}}, \underline{\mathbf{x}} \mid \lambda'] \mid \underline{\mathbf{x}}, \lambda'\right] \\
&= Q(\lambda, \lambda') - Q(\lambda', \lambda')
\end{aligned}$$

Recall Jensen inequality in probability theory

$$\varphi(E[X]) \leq E[\varphi(X)] \quad \text{for a convex function } \varphi$$

EM

Summary

- A HMM is a timeseries parametric statistical model with latent variables
- Assumptions are made on the latent variable model & the relationship between the observed and the latent variable
- Learning the parameters from data is not tractable the usual way, i.e. finding global optimum with derivatives, is not feasible
- We prove another scheme to learn parameters

EM

Example with timeseries

Example with timeseries

You observe data $\underline{\mathbf{x}} = [\mathbf{x}_1, \dots, \mathbf{x}_T]$, and you know this model:

$$\begin{aligned} \mathbf{x}_0 &= 0 \\ \mathbf{x}_t &= \mathbf{S}_t \lambda \mathbf{x}_{t-1} + W_t, \quad \forall t = 1, \dots, T, \end{aligned}$$

where $\lambda \in \mathbb{R}$, $\forall t \in [T], \mathbf{S}_t \sim \mathcal{U}(\{-1, +1\}), W_t \sim \mathcal{N}(0, \sigma^2)$ is white noise.
Questions:

1. Draw the relationships between the random variables.
2. Write $p(\underline{\mathbf{X}}, \underline{\mathbf{S}} | \lambda)$.
3. Write $p(\underline{\mathbf{S}} | \underline{\mathbf{X}}, \lambda)$ for the *expectation* step
4. Write $Q(\lambda, \lambda') = E_{p(\underline{\mathbf{S}}|\underline{\mathbf{X}}, \lambda')}[\ln p(\underline{\mathbf{S}}, \underline{\mathbf{X}} | \lambda)]$ so that you can perform the *maximization* step.

EM

Example with timeseries

Joint distribution

$$p\left(\underline{\mathbf{S}} = (i_1 \dots i_T), \underline{\mathbf{x}} \mid \lambda\right) = \prod_{t=1}^{T} p\left(\mathbf{S}_t = i_t, \mathbf{x}_t \mid \mathbf{x}_{t-1}, \lambda\right)$$

where $\forall t \in [T]$

$$\begin{aligned} p\left(\mathbf{S}_t = i_t, \mathbf{x}_t \mid \mathbf{x}_{t-1}, \lambda\right) &= p\left(\mathbf{S}_t = i_t\right) p\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{S}_t = i_t, \lambda\right) \\ p\left(\mathbf{S}_t = +1, \mathbf{x}_t \mid \mathbf{x}_{t-1}, \lambda\right) &= \frac{1}{2} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\mathbf{x}_t - (+1)\lambda \mathbf{x}_{t-1})^2}{2\sigma^2}} \\ p\left(\mathbf{S}_t = -1, \mathbf{x}_t \mid \mathbf{x}_{t-1}, \lambda\right) &= \frac{1}{2} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\mathbf{x}_t - (-1)\lambda \mathbf{x}_{t-1})^2}{2\sigma^2}} \end{aligned}$$

EM

Example with timeseries

Posterior distribution

$$\gamma_{+,t} = p\left[\mathbf{S}_t = +1 \mid \mathbf{x}_t, \mathbf{x}_{t-1}, \lambda\right] = \frac{p(\mathbf{x}_t|\mathbf{S}_t = +1, \mathbf{x}_{t-1}, \lambda)p(\mathbf{S}_t = +1|\mathbf{x}_{t-1}, \lambda)}{p(\mathbf{x}_t|\mathbf{x}_{t-1}, \lambda)}$$

$$= \frac{e^{-\frac{(\mathbf{x}_t - \lambda \mathbf{x}_{t-1})^2}{2\sigma^2}}}{e^{-\frac{(\mathbf{x}_t - \lambda \mathbf{x}_{t-1})^2}{2\sigma^2}} + e^{-\frac{(\mathbf{x}_t + \lambda \mathbf{x}_{t-1})^2}{2\sigma^2}}}$$

$$= \frac{e^{\frac{\lambda \mathbf{x}_t \mathbf{x}_{t-1}}{\sigma^2}}}{e^{\frac{\lambda \mathbf{x}_t \mathbf{x}_{t-1}}{\sigma^2}} + e^{-\frac{\lambda \mathbf{x}_t \mathbf{x}_{t-1}}{\sigma^2}}}$$

$$\gamma_{-,t} = 1 - \gamma_{+,t}$$

$$= \frac{e^{-\frac{\lambda x_t x_{t-1}}{\sigma^2}}}{e^{\frac{\lambda x_t x_{t-1}}{\sigma^2}} + e^{-\frac{\lambda x_t x_{t-1}}{\sigma^2}}}$$

EM

Example with timeseries

Q function

$$
\begin{aligned}
Q(\lambda, \lambda') &= \sum_{i_1} \cdots \sum_{i_T} P\left[\underline{\mathbf{S}} = (i_1 \dots i_T) \mid \mathbf{x}, \lambda'\right] \ln p\left[\underline{\mathbf{S}} = (i_1 \dots i_T), \mathbf{x} \mid \lambda\right] \\
&= \sum_{i_1} \cdots \sum_{i_T} P\left[\underline{\mathbf{S}} = (i_1 \dots i_T) \mid \mathbf{x}, \lambda'\right] \sum_{t=1}^{T} \ln P\left[\mathbf{S}_t = i_t, \mathbf{x}_t \mid \mathbf{x}_{t-1}, \lambda\right] \\
&= \sum_{t=1}^{T} \sum_{i_t} P\left[\mathbf{S}_t = i_t \mid \mathbf{x}_t, \mathbf{x}_{t-1}, \lambda'\right] \ln P\left[\mathbf{S}_t = i_t, \mathbf{x}_t \mid \mathbf{x}_{t-1}, \lambda\right] \\
&= \sum_{t=1}^{T} \left[\gamma_{+,t} \frac{-(\mathbf{x}_t - \lambda \mathbf{x}_{t-1})^2}{2\sigma^2} + \gamma_{-,t} \frac{-(\mathbf{x}_t + \lambda \mathbf{x}_{t-1})^2}{2\sigma^2} + Cst\right],
\end{aligned}
$$

where Cst does not depend on λ and so will not intervene in the maximization.

EM

Example with timeseries

Q function maximization

$$
\begin{aligned}
0 = \left.\frac{\partial Q(\lambda, \lambda')}{\partial \lambda}\right|_{\lambda^*} &= \sum_{t=1}^{T} \gamma_{+,t} \frac{(\mathbf{x}_t - \lambda^* \mathbf{x}_{t-1})\mathbf{x}_{t-1}}{\sigma^2} - \gamma_{-,t} \frac{(\mathbf{x}_t + \lambda^* \mathbf{x}_{t-1})\mathbf{x}_{t-1}}{\sigma^2} \\
&= \sum_{t=1}^{T} (\gamma_{+,t} - \gamma_{-,t}) \frac{\mathbf{x}_t \mathbf{x}_{t-1}}{\sigma^2} - \underbrace{(\gamma_{+,t} + \gamma_{-,t})}_{1} \frac{\lambda^* \mathbf{x}_{t-1}^2}{\sigma^2} \\
&= \sum_{t=2}^{T} (\gamma_{+,t} - \gamma_{-,t}) \frac{\mathbf{x}_t \mathbf{x}_{t-1}}{\sigma^2} - \frac{\lambda^* \mathbf{x}_{t-1}^2}{\sigma^2}
\end{aligned}
$$

Note that $\gamma_{+,t} - \gamma_{-,t} = \tanh\left(\frac{\lambda' \mathbf{x}_t \mathbf{x}_{t-1}}{\sigma^2}\right)$

$$
\lambda^* = \frac{\sum_{t=2}^{T} \tanh(\frac{\lambda' \mathbf{x}_t \mathbf{x}_{t-1}}{\sigma^2}) \mathbf{x}_t \mathbf{x}_{t-1}}{\sum_{t=2} \mathbf{x}_{t-1}^2}
$$

EM

Example with K-means & GMMs

K-means

- Suppose $N \in \mathbb{N}_*$ data points $\mathbf{x}_n \in \mathbb{R}^d$ for $n = 1, \cdots, N$.
- Suppose we can measure distances in $\mathbb{R}^d$ with a bivariate function d, e.g. $d(\mathbf{x}_n, \mathbf{x}_{n'}) = ||\mathbf{x}_n - \mathbf{x}_{n'}||_2$.
- We want to assign all our data points to one of $K \in \mathbb{N}_*$ clusters, characterized by their means $\boldsymbol{\mu}_k \in \mathbb{R}^d$ for $k = 1, \cdots, K$.
- We use the notation $r_{nk} \in \{0, 1\}$, where $r_{nk} = 1$ and $r_{nk'} = 0$ for $k' \neq k$ if point n is assigned to the k-th cluster.
- The goal of K-means clustering is to
  1. learn the means of the each cluster and
  2. assign every point in the data set to one of the clusters.

EM

Example with K-means & GMMs

Optimization problem

The goal is to find $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$ which minimize

$$J(\{r_{nk}\},\{\boldsymbol{\mu}_k\}) = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}||\mathbf{x}_n - \boldsymbol{\mu}_k||_2^2, \tag{7}$$

Example with K-means & GMMs: Cluster assignment

At iteration i, what are the optimal values for $r_{nk}^{(i)}$ according to the current estimate $\boldsymbol{\mu}_k$?

$$r_{nk}^{(i)} = \begin{cases} 1 & \text{if } k = \arg\min_j ||\mathbf{x}_n - \boldsymbol{\mu}_j^{(i-1)}||_2^2 \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

EM

Example with K-means & GMMs

Optimization problem

How can you optimize $\boldsymbol{\mu}_k^{(i)}$ based on the new estimates for $r_{nk}^{(i)}$?
Derive and set to 0:

$$\left.\frac{\partial J(r_{nk}^{(i)}, \boldsymbol{\mu}_k)}{\partial \boldsymbol{\mu}_k}\right|_{\boldsymbol{\mu}_k^{(i)}} = 0$$

$$\Longrightarrow 2\sum_{n=1}^{N} r_{nk}^{(i)}(\mathbf{x}_n - \boldsymbol{\mu}_k^{(i)}) = 0 \tag{9}$$

$$\boldsymbol{\mu}_k^{(i)} = \frac{\sum_n r_{nk}^{(i)}\mathbf{x}_n}{\sum_n r_{nk}^{(i)}}$$

Probabilistic K-means

Again, denote $\mathbf{X} = \{\mathbf{x}_n\}_{n=1}^{N}$ the set of observed training data.

- Probabilistic interpretation of K-means: defining the clusters in terms of distributions rather than simply by there means.
- We will aim at maximizing the likelihood of the dataset $\mathbf{X}$ wrt to a mixture of Gaussian model:

$$\ln p(\mathbf{X}|\lambda) = \sum_{n=1}^{N} \ln \left[\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right], \tag{10}$$

  where $\lambda = \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
- And assuming initial values for λ.

EM

Example with K-means & GMMs

Derivatives wrt $\boldsymbol{\mu}_k$ must be 0:

$$\left.\frac{\partial \ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_k}\right|_{\boldsymbol{\mu}_k^\star} = 0$$

$$\implies 0 = \sum_{n=1}^{N} \frac{\frac{\partial \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k}}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \text{ where we derived } \ln(.) \text{ wrt } \boldsymbol{\mu}_k$$

Next, we derive the numerator (using equation (86)) from the Matrix Cookbook)

$$\frac{\partial \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k} = \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k),$$

Denoting $\gamma(z_{nk}) = \dfrac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \in \mathbb{R}$, we get: $0 = \displaystyle\sum_{n=1}^{N} \gamma(z_{nk})\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_k^*)$

Finally :

$$\boldsymbol{\mu}_k^* = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\mathbf{x}_n, \text{ with } N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

EM

Example with K-means & GMMs

Derivatives wrt $\boldsymbol{\Sigma}_k$ must be 0:

$$\left.\frac{\partial \ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}_k}\right|_{\boldsymbol{\Sigma}_k^\star} = 0 \quad \Longrightarrow$$

$$\boldsymbol{\Sigma}_k^\star = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T$$

Details for the derivation at
`https://www.cs.ubc.ca/~murphyk/Teaching/CS340-Fall07/reading/gauss.pdf`

EM

Example with K-means & GMMs

The mixture parameter $\boldsymbol{\pi}_k$ has the additional constraint that $\sum_{k=1}^{K} \pi_k = 1$, thus we introduce a Lagrangian multiplier and maximize wrt π_k and β, the quantity:

$$l(\pi_k, \beta) = \ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) - \beta \left(1 - \sum_{k=1}^{K} \pi_k\right)$$

Deriving wrt π_k and setting to 0 gives

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} + \beta$$

Multiplying by π_k summing over k and using the constraint :

$$\beta = -N, \pi_k = \frac{N_k}{N},$$

where $N_k = \sum_n \gamma(z_{nk})$ and $N = \sum_k N_k$.

EM

Example with K-means & GMMs

The EM for Gaussian Mixtures p438 Bishop 2006:

1. Initialize the parameters
2. Evaluate the responsibilities $\gamma(z_{nk})$
3. Re-estimate the parameters using the current responsibilities
4. Check convergence

EM

Example with K-means & GMMs

Reducing GMM to K-means

- How should you define the GMM so that clustering reduces to K-means ?
- Assume $\boldsymbol{\Sigma}_k = \epsilon \mathbf{I}$, this leads to

$$\gamma(z_{nk}) = \frac{\pi_k \exp(-||\mathbf{x}_n - \boldsymbol{\mu}_k||_2^2/2\epsilon)}{\sum_j \pi_j \exp(-||\mathbf{x}_n - \boldsymbol{\mu}_j||_2^2/2\epsilon)}.$$

- If we denote $k^\star$ the cluster mean that is closer to point n,
- then for $k \neq k^\star, \gamma(z_{nk}) \to 0$ and $\gamma(z_{nk^\star}) \to 1$ when $\epsilon \to 0$,
- in turn leading to a hard assignment to cluster $k^\star$ for point n.

EM

Example with K-means & GMMs

Summary

- The procedure of EM is:
  1. Select initial parameters λ'
  2. Write $Q(\lambda, \lambda') = E_{p(\underline{\mathbf{S}}|\underline{\mathbf{X}},\lambda')}[\ln p(\underline{\mathbf{S}}, \underline{\mathbf{X}}|\lambda)]$.
  3. Maximize Q wrt λ.
- We saw EM in practice for a timeseries model.
- We revisited K-means as a particular case of clustering with GMMs.

Lecture 2 Baum-Welch

EM + HMM = Baum-Welch

The Q function, i.e the expectation step

Back to HMMs with N states

$$p(\underline{\mathbf{x}}, \underline{\mathbf{s}}|\lambda) = p(\mathbf{s}_1|\lambda)p(\mathbf{x}_1|\mathbf{s}_1, \lambda) \prod_{t=2}^{T} p(\mathbf{x}_t|\mathbf{s}_t, \lambda)p(\mathbf{s}_t|\mathbf{s}_{t-1}, \lambda),$$

with $\lambda = \{q, A, B\}$.

The Q function

$$Q(\lambda, \lambda') = \sum_{i_1}^{N} \cdots \sum_{i_T}^{N} p(\underline{\mathbf{s}} = (i_1, \ldots, i_T)|\underline{\mathbf{x}}, \lambda') \ln p(\underline{\mathbf{x}}, \underline{\mathbf{s}} = (i_1, \ldots, i_T), \lambda)$$

EM + HMM = Baum-Welch

The Q function, i.e the expectation step

The Q function

$$Q(\lambda, \lambda') = \sum_{i_1}^{N} \cdots \sum_{i_T}^{N} p(\underline{\mathbf{s}} = (i_1, \ldots, i_T) | \underline{\mathbf{x}}, \lambda')[\ln p(\mathbf{s}_1 | \lambda) + \sum_{t=2}^{T} \ln p(\mathbf{s}_t | \mathbf{s}_{t-1}, \lambda) + \sum_{t=1}^{T} \ln p(\mathbf{x}_t | \mathbf{s}_t, \lambda)]$$

We look at the different parameters independently:

$$Q_1(\lambda, \lambda') = \sum_{i_1}^{N} \cdots \sum_{i_T}^{N} p(\underline{\mathbf{s}} = (i_1, \ldots, i_T) | \underline{\mathbf{x}}, \lambda') \ln p(\mathbf{s}_1 | \lambda)$$

$$Q_2(\lambda, \lambda') = \sum_{i_1}^{N} \cdots \sum_{i_T}^{N} p(\underline{\mathbf{s}} = (i_1, \ldots, i_T) | \underline{\mathbf{x}}, \lambda') \left[\sum_{t=2}^{T} p(\mathbf{s}_t | \mathbf{s}_{t-1}, \lambda) \right]$$

$$Q_3(\lambda, \lambda') = \sum_{i_1}^{N} \cdots \sum_{i_T}^{N} p(\underline{\mathbf{s}} = (i_1, \ldots, i_T) | \underline{\mathbf{x}}, \lambda') \left[\sum_{t=1}^{T} \ln p(\mathbf{x}_t | \mathbf{s}_t, \lambda) \right]$$

EM + HMM = Baum-Welch

The Q function, i.e the expectation step

Trick

An important trick in the calculation is to marginalize the posterior with everything expect the time index in the term from the joint distribution. E.g. Q_1:

$$p(\underline{\mathbf{s}} = (i_1, \dots, i_T)|\underline{\mathbf{x}}, \lambda') = p(\mathbf{s}_1 = i_1|\underline{\mathbf{x}}, \lambda') p(\mathbf{s}_2, \dots, \mathbf{s}_T = (i_2, \dots, i_T)|\underline{\mathbf{x}}, \mathbf{s}_1 = i_1, \lambda')$$

$$Q_1(\lambda, \lambda') = \sum_{i_1=1}^{N} \cdots \sum_{i_T=1}^{N} p(\underline{\mathbf{s}} = (i_1, \dots, i_T)|\underline{\mathbf{x}}, \lambda') \ln p(\mathbf{s}_1|\lambda)$$

$$Q_1(\lambda, \lambda') = \sum_{i_1=1}^{N} p(\mathbf{s}_1 = i_1|\underline{\mathbf{x}}, \lambda') \ln p(\mathbf{s}_1|\lambda) \cdot \underbrace{\sum_{i_2=1}^{N} \dots, \sum_{i_T=1}^{N} p(\mathbf{s}_2, \dots, \mathbf{s}_T = (i_2, \dots, i_T)|\underline{\mathbf{x}}, \mathbf{s}_1 = i_1, \lambda')}_{=1}$$

EM + HMM = Baum-Welch

The Q function, i.e the expectation step

Similarly in Q_2

$$Q_2(\lambda, \lambda') = \sum_{i_1=1}^{N} \cdots \sum_{i_T=1}^{N} p(\underline{\mathbf{s}} = (i_1, \ldots, i_T) | \underline{\mathbf{x}}, \lambda') \left[\sum_{t=2}^{T} \ln p(\mathbf{s}_t | \mathbf{s}_{t-1}, \lambda) \right]$$

$$= \sum_{t=2}^{T} \sum_{i_{t-1}}^{N} \sum_{i_t}^{N} p(\mathbf{s}_{t-1} = i_{t-1}, \mathbf{s}_t = i_t | \underline{\mathbf{x}}, \lambda') \ln p(\mathbf{s}_t | \mathbf{s}_{t-1}, \lambda)$$

$$\cdot \underbrace{\left[\sum_{i_{k \neq t-1, t}} p(\cap_{k \neq t-1, t} (\mathbf{s}_k = i_k) | \underline{\mathbf{x}}, \lambda') \right]}_{=1}$$

EM + HMM = Baum-Welch

The Q function, i.e the expectation step

Similarly in Q_3:

$$Q_3(\lambda,\lambda') = \sum_{i_1=1}^{N}\cdots\sum_{i_T=1}^{N} p(\underline{\mathbf{s}} = (i_1,\ldots,i_T)|\underline{\mathbf{x}},\lambda')\left[\sum_{t=1}^{T}\ln p(\mathbf{x}_t|\mathbf{s}_t,\lambda)\right]$$

$$= \sum_{t=1}^{T}\sum_{i_t=1}^{N} p(\mathbf{s}_t = i_t|\underline{\mathbf{x}},\lambda')\ln p(\mathbf{x}_t|\mathbf{s}_t,\lambda)\cdot\underbrace{\left[\sum_{(i_{k\neq t})_k} p(\cap_{k\neq t}(\mathbf{s}_k = i_k)|\underline{\mathbf{x}},\lambda')\right]}_{=1}$$

EM + HMM = Baum-Welch

The Q function, i.e the expectation step

Finally, if $\forall i, t \in [N] \times [T] \quad \gamma_{i,t} = p(\mathbf{s}_t = i|\underline{\mathbf{x}}, \lambda')$, and
$\forall (i,j) \in [N]^2 \quad \xi_{i,j,t} = p(\mathbf{s}_{t-1} = i, \mathbf{s}_t = j|\lambda')$

$$Q_1(\lambda, \lambda') = \sum_{i=1}^{N} \gamma_{i,1} \ln p(\mathbf{s}_1 = i|\lambda)$$

$$Q_2(\lambda, \lambda') = \sum_{t=2}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} \xi_{i,j,t} \ln p(\mathbf{s}_t = j|\mathbf{s}_{t-1} = i, \lambda)$$

$$Q_3(\lambda, \lambda') = \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_{i,t} \ln p(\mathbf{x}_t|\mathbf{s}_t = i, \lambda)$$

EM + HMM = Baum-Welch

The Q function, i.e the expectation step

$\forall i, j, t \in [N] \times [N] \times [T]$, how to calculate $\gamma_{i,t} = p(\mathbf{s}_t = i | \underline{\mathbf{x}}, \lambda')$, and $\xi_{i,j,t} = p(\mathbf{s}_{t-1} = i, \mathbf{s}_t = j | \underline{\mathbf{x}}, \lambda')$?

- See chap 5 !

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_1

$$Q_1(\mathbf{q}, \lambda') = \sum_{i=1}^{N} \gamma_{i,1} \ln q_i$$

Similar to updating the mixture weights in a GMM !

$$\forall i \in [N] \quad q_i^\star = \frac{\gamma_{i,1}}{\sum_{j=1}^{N} \gamma_{j,1}}$$

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_2

$$Q_2(A, \lambda') = \sum_{t=2}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} \xi_{i,j,t} \ln a_{i,j}$$

There are N additional constraints

$$\forall i \in [N] \quad \sum_{j=1}^{N} a_{i,j} = 1$$

We define Lagrange multipliers: $\forall i \in [N] \quad \nu_i$, the criteria becomes:

$$\forall i, j \in [N]^2 \quad l(\nu_i, a_{i,j}, \lambda') = Q_2(A, \lambda') + \nu_i(1 - \sum_{k=1}^{N} a_{i,k})$$

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_2

Solving for $a_{i,j}$.

$$\left.\frac{\partial l(\nu_i, a_{i,j}, \lambda')}{\partial a_{i,j}}\right|_{a^*_{i,j}} = 0 \implies$$

$$\sum_{t=2}^{N} \frac{\xi_{i,j,t}}{a_{i,j}} - \nu_i = 0$$

With the constraint: $\nu_i = \sum_{k=1}^{N} \sum_{t=2}^{T} \xi_{i,k,t}$

Then: $a^{\star}_{i,j} = \frac{1}{\nu_i} \sum_{t=2}^{T} \xi_{i,j,t}$

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_3

$$Q_3(\lambda, \lambda') = \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_{i,t} \ln p(\mathbf{x}_t | \mathbf{s}_t = i, \lambda)$$

We still have not spoken about the emission distributions !

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_3

$$Q_3(\lambda, \lambda') = \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_{i,t} \ln p(\mathbf{x}_t | \mathbf{s}_t = i, \lambda)$$

- $p(\mathbf{x}_t | \mathbf{s}_t = i, \lambda)$ are the emission density functions
- it should be possible to differentiate the density function wrt its parameters
- We are going to assume that $\forall i, t \in [N] \times [T]$ the emission distributions are either
  1. Discrete: $p(\mathbf{x}_t | \mathbf{s}_t = i, \lambda) = [b_{i,1}, \dots, b_{i,M}]$, with $\sum_{m=1}^{M} b_{i,m} = 1$.
  2. GMM: $p(\mathbf{x}_t | \mathbf{s}_t = i, \lambda) = \sum_{m=1}^{M} w_{im} \mathcal{N}(\mathbf{x}_t; \mu_{im}, C_{im})$, with $\sum_{m=1}^{M} w_{i,m} = 1$

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_3

Discrete case: $\mathbf{X}_t \in \{\alpha_1, \dots, \alpha_M\}$. It is useful to define auxiliary random variables when dealing with discrete distributions, i.e. if we observe $\mathbf{X}_t = \mathbf{x}_t = \alpha_m$, we write $\mathbf{Z}_t = m$, then:

$$b_{i,m} = p(\mathbf{X}_t = \alpha_m | \mathbf{S}_t = \mathbf{s}_t, \lambda) = p(\mathbf{Z}_t = m | \mathbf{S}_t = i, \lambda) = \sum_{k=1}^{M} \mathbb{1}(\mathbf{z}_t = k) p(\mathbf{Z}_t = k | \mathbf{S}_t = i, \lambda)$$

$$Q_3(\lambda, \lambda') = \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_{i,t} \ln \sum_{k=1}^{M} \mathbb{1}(\mathbf{z}_t = k) p(\mathbf{Z}_t = k | \mathbf{S}_t = i, \lambda)$$

$$Q_3(\lambda, \lambda') = \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_{i,t} \ln \sum_{k=1}^{M} \mathbb{1}(\mathbf{z}_t = k) b_{i,k}$$

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_3

$$Q_3(\lambda, \lambda') = \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma_{i,t} \ln \sum_{k=1}^{M} \mathbb{1}(\mathbf{z}_t = k) b_{i,k}$$

The method is similar to optimizing Q_1. We take N Lagrange multipliers ν_i.

$$\forall i, m \in [N] \times [T] \; \frac{\partial}{\partial b_{i,m}} \left[Q_3(\lambda, \lambda') + \nu_i (1 - \sum_{k=1}^{M} b_{i,k}) \right] \Bigg|_{b^*_{i,m}} = 0$$

$$\implies 0 = \sum_{t=1}^{T} \frac{\gamma_{i,t}}{b^*_{i,m}} \mathbb{1}(\mathbf{z}_t = m) - \nu_i$$

$$\text{Which gives: } b^*_{i,m} = \frac{1}{\nu_i} \sum_{t=1}^{T} \gamma_{i,t} \mathbb{1}(\mathbf{z}_t = m), \text{ with } \nu_i = \sum_{k=1}^{M} \sum_{t=1}^{T} \gamma_{i,t} \mathbb{1}(\mathbf{z}_t = k)$$

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_3

GMM case: We define a random variable to help solve the optimization problem. We augment the latent variable space with $\mathbf{U}_t \in \{1, \dots, M\}$ which indicates which mixture component is chosen at time t. The joint distribution is now written:

$$p(\underline{\mathbf{x}}, \underline{\mathbf{s}}, \underline{\mathbf{u}}|\lambda) = p(\mathbf{s}_1|\lambda)p(\mathbf{x}_1, \mathbf{u}_1|\mathbf{s}_1, \lambda) \prod_{t=2}^{T} p(\mathbf{x}_t, \mathbf{u}_t|\mathbf{s}_t, \lambda)p(\mathbf{s}_t|\mathbf{s}_{t-1}, \lambda)$$

We use the new latent variable as follows, $\forall i, t, m \in [N] \times [T] \times [M]$:

$$p(\mathbf{x}_t, \mathbf{U}_t = m|\mathbf{S}_t = i) = p(\mathbf{x}_t|\mathbf{S}_t = i, \mathbf{U}_t = m)p(\mathbf{U}_t = m|\mathbf{S}_t = i)$$

where $p(\mathbf{x}_t|\mathbf{S}_t = i, \mathbf{U}_t = m) = \mathcal{N}(\mathbf{x}_t; \mu_{i,m}, C_{i,m})$ with mixture weight $w_{i,m} = p(\mathbf{U}_t = m|\mathbf{S}_t = i)$.

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_3

We also define $\forall m, i, t \in [M] \times [N] \times [T]$

$$\begin{aligned}
\gamma_{i,m,t} &= p(\mathbf{S}_t = i, \mathbf{U}_t = m | \underline{\mathbf{x}}, \lambda') \quad \text{Question? How is this related to } \gamma_{i,t} \text{ ?} \\
&= p(\mathbf{U}_t = m | \underline{\mathbf{x}}, \mathbf{S}_t = i, \lambda') \underbrace{p(\mathbf{S}_t = i | \underline{\mathbf{x}}, \lambda')}_{\gamma_{i,t}} \\
&= \gamma_{i,t} \frac{p(\mathbf{U}_t = m, \mathbf{x}_t | \mathbf{S}_t = i, \underline{\mathbf{x}}_{t' \neq t}, \lambda')}{p(\mathbf{x}_t | \mathbf{S}_t = i, \lambda')} \\
&= \gamma_{i,t} \frac{p(\mathbf{U}_t = m, \mathbf{x}_t | \mathbf{S}_t = i, \lambda')}{p(\mathbf{x}_t | \mathbf{S}_t = i, \lambda')} \\
&= \gamma_{i,t} \frac{w_{im} \mathcal{N}(\mathbf{x}_t ; \mu_{i,m}, C_{i,m})}{\sum_{k=1}^{M} w_{i,k} \mathcal{N}(\mathbf{x}_t ; \mu_{i,k}, C_{i,k})}
\end{aligned}$$

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_3

GMM case: With our new variable $\mathbf{U}_t$, Q is written:

$$Q(\lambda, \lambda') = \sum_{i_1}^{N} \cdots \sum_{i_T}^{N} \sum_{j_1}^{M} \cdots \sum_{j_T}^{M} p(\underline{\mathbf{S}} = (i_1, \dots, i_T), \underline{\mathbf{U}} = (j_1, \dots, j_T) | \underline{\mathbf{x}}, \lambda') \cdot$$
$$\ln p(\underline{\mathbf{x}}, \underline{\mathbf{S}} = (i_1, \dots, i_T), \underline{\mathbf{U}} = (j_1, \dots, j_T) | \lambda)$$

In particular,

$$Q_3(\lambda, \lambda') = \sum_{t=1}^{T} \sum_{i_t=1}^{N} \sum_{j_t=1}^{N} p(\mathbf{S}_t = i_t, \mathbf{U}_t = j_t | \underline{\mathbf{x}}, \lambda') \ln p(\mathbf{S}_t, \mathbf{U}_t = j_t | \mathbf{S}_t = i_t, \lambda)$$

$$Q_3(\lambda, \lambda') = \sum_{t=1}^{T} \sum_{i_t=1}^{N} \sum_{j_t=1}^{N} p(\mathbf{S}_t = i_t, \mathbf{U}_t = j_t | \underline{\mathbf{x}}, \lambda') \ln p(\mathbf{x}_t | \mathbf{U}_t = j_t, \mathbf{S}_t = i_t, \lambda) p(\mathbf{U}_t = j_t | \mathbf{S}_t = i_t, \lambda)$$

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_3

GMM case: With our new variable $\mathbf{U}_t$, Q_3 (with new indices) is written:

$$Q_3(\lambda, \lambda') = \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{m=1}^{M} \underbrace{p(\mathbf{S}_t = i, \mathbf{U}_t = m | \underline{\mathbf{x}}, \lambda')}_{\gamma_{i,m,t}} \ln p(\mathbf{x}_t, \mathbf{U}_t = m | \mathbf{s}_t = i, \lambda)$$

$$= \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{m=1}^{M} \gamma_{i,m,t} \left(\ln w_{i,m} + \ln \mathcal{N}(\mathbf{x}_t; \mu_{i,m}, C_{i,m})\right)$$

$$\text{and } \gamma_{i,m,t} = \gamma_{i,t} \frac{w_{im} \mathcal{N}(\mathbf{x}_t; \mu_{i,m}, C_{i,m})}{\sum_{k=1}^{M} w_{i,k} \mathcal{N}(\mathbf{x}_t; \mu_{i,k}, C_{i,k})}$$

EM + HMM = Baum-Welch

The maximization step

Optimizing Q_3

The final update is similar to updating a GMM model, $\forall i, m \in [N] \times [M]$:

$$w^*_{im} = \frac{\sum_t \gamma_{i,m,t}}{\sum_{k=1}^{M} \sum_t \gamma_{i,m,t}}$$

$$\boldsymbol{\mu}^*_{im} = \frac{\sum_t \gamma_{i,m,t} \mathbf{x}_t}{\sum_t \gamma_{i,m,t}}$$

$$C^*_{im} = \frac{\sum_t \gamma_{i,m,t} (\mathbf{x}_t - \boldsymbol{\mu}^*_{im})(\mathbf{x}_t - \boldsymbol{\mu}^*_{im})^T}{\sum_t \gamma_{i,m,t}}$$

EM + HMM = Baum-Welch

The maximization step

Summary

We now have all the update rules to iteratively update the Q function !

References

Based on *Pattern Recognition Fundamental Theory and Exercise Problems* by ARNE LEIJON & GUSTAV EJE HENTER

Lecture 3 Lagrange multipliers

Optimization with an equality constraint

Remember the maximization problems that we encounter in the maximization steps of the Baum-Welch algorithm.

Problem Statement:

$$\begin{array}{cc} \text{Maximize a function} & f(x,y) \\ \text{Subject to a constraint} & g(x,y)=0 \end{array} \tag{11}$$

The method of Lagrange multipliers is a method for solving optimization problems with equality constraints.

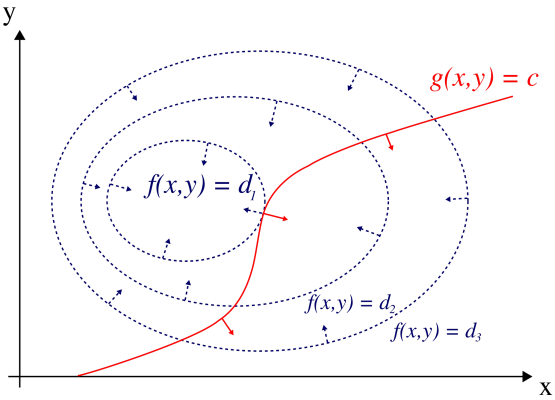
Main theorem: If it exists, a local maximum is where the level curves of f are tangent to the constraint curve g, i.e. where the gradients of f and g are parallel.

Practically:

- Maximizing $l(x,y,\lambda) = f(x,y) - \lambda g(x,y)$
- By solving for λ and the variables: $\begin{cases} \nabla f & = \lambda \nabla g \\ g(x,y) & = 0 \end{cases}$
- Solves the constrained problem in (1)

Geometrically

- The gradients ∇f and ∇g determine the direction of greatest increase.
- At an optimal point, these gradients must be **parallel**: $\nabla f = \lambda \nabla g$.
- This ensures that moving along the constraint does not increase or decrease f.



(Source: Wikipedia)

Analytically

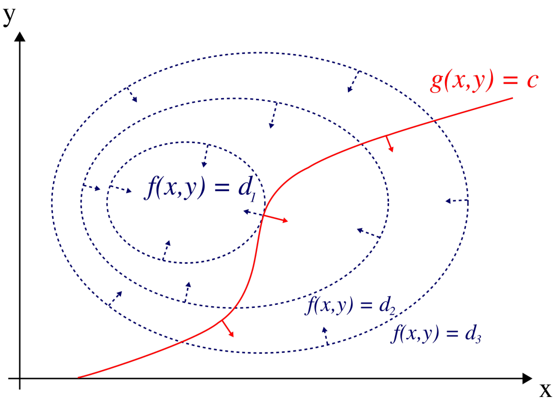
- Take a vector space with an inner product $(\mathbb{R}^d, \langle ., . \rangle)$ and two functions $f, g : \mathbb{R}^d \to \mathbb{R}$, such that both are $\mathcal{C}^1$ (derivable with continuous derivatives)
- Suppose that a local maximum of f exists at a point $P = (x_1^*, \dots, x_d^*)$ on the constraint surface $\mathcal{S} = \{x_1, \dots, x_d \mid g(x_1, \dots, x_d) = 0\}$.
- Let $r(t) = (x_1(t), \dots, x_d(t))$ denote a parameterized curve on $\mathcal{S}$, i.e. such that $\forall t \in \mathbb{R}\ g(r(t)) = 0$, and such that $r(0) = P$.
- Let $h(t) = f(r(t)) = f(x_1(t), \dots, x_d(t))$, h has a local maximum at $t = 0$.
- The derivative of h is written

$$h'(t) = \langle \nabla f|_{r(t)}, r'(t) \rangle$$

- then what ?

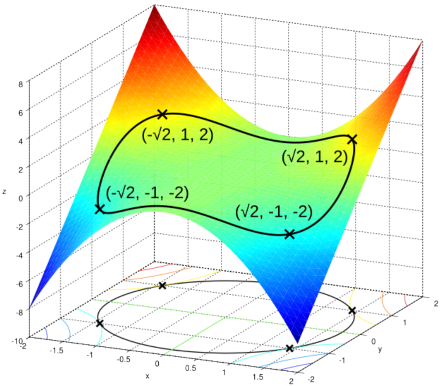
Analytically

- Since P is a local maximum for $h(t) = f(r(t))$, at $t = 0$: $h'(0) = \langle \nabla f|_P , r'(0) \rangle = 0$
- This is true $\forall\ r(t)$, implying $\nabla f|_P$ is perpendicular to every curves on the surface at P. Implying $\nabla f|_P$ is perpendicular to the constraint surface at P, in particular it is parallel with $\nabla g|_P$ (which is also perpendicular to the surface).



(Source: Wikipedia)

An example in $\mathbb{R}^2$



(Source: Wikipedia, $r = \sqrt{3}$)

- **Problem**: Maximize a function $f(x, y)$, Subject to constraint $g(x, y) = 0$, with

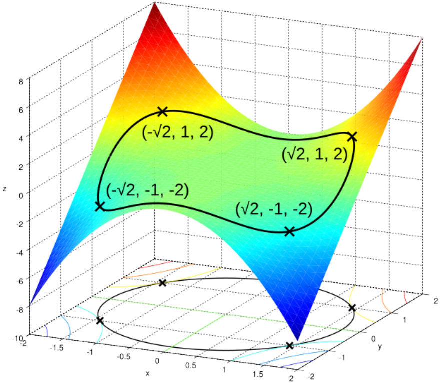
  $f(x, y) = x^2 y$, $g(x, y) = x^2 + y^2 - r^2 = 0$
- Question: What shape is g ? A circle of radius r
- $\mathcal{L}(x, y, \lambda) = f(x, y) + \lambda g(x, y) = x^2 y + \lambda(x^2 + y^2 - r^2)$

$$\nabla_{x,y,\lambda}\mathcal{L}(x, y, \lambda) = \left(\frac{\partial \mathcal{L}}{\partial x}, \frac{\partial \mathcal{L}}{\partial y}, \frac{\partial \mathcal{L}}{\partial \lambda}\right)$$

$$\nabla_{x,y,\lambda}\mathcal{L}(x, y, \lambda) = 0 \Leftrightarrow \begin{cases} 2xy + 2\lambda x = 0 \\ x^2 + 2\lambda y = 0 \\ x^2 + y^2 - r^2 = 0 \end{cases}$$

$$\Leftrightarrow \begin{cases} x(y + \lambda) = 0 & \text{(a)} \\ x^2 = -2\lambda y & \text{(b)} \\ x^2 + y^2 = r^2 & \text{(c)} \end{cases}$$

An example in $\mathbb{R}^2$



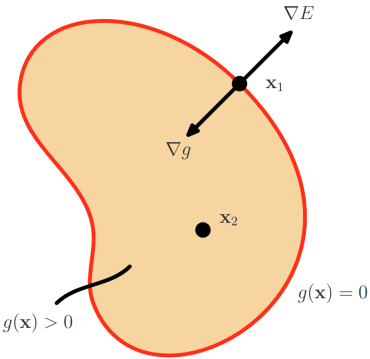
(Source: Wikipedia, $r = \sqrt{3}$)

- $f(x, y) = x^2 y$, $g(x, y) = x^2 + y^2 - r^2 = 0$

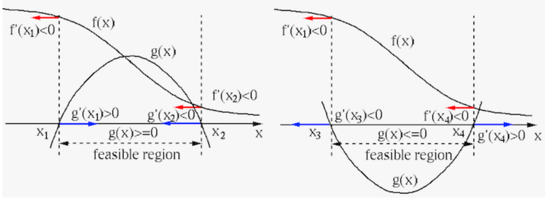
$$\dots \Leftrightarrow \begin{cases} x(y+\lambda) = 0 & \text{(a)} \\ x^2 = -2\lambda y & \text{(b)} \\ x^2 + y^2 = r^2 & \text{(c)} \end{cases}$$

- (a) $\implies x = 0$ or $\lambda = -y$
  - $x = 0 \implies y = \pm r\,(c)$ and thus $\lambda = 0\,(b)$
  - $\lambda = -y \implies x = \pm y\sqrt{2}\,(b)$, $y = \pm\frac{r}{\sqrt{3}}\,(c)$
- 6 possible critical points for $\mathcal{L}$: $(0, r, 0)$, $(0, -r, 0)$; $(r\sqrt{\frac{2}{3}}, \frac{r}{\sqrt{3}}, -\frac{r}{\sqrt{3}})$, $(r\sqrt{\frac{2}{3}}, -\frac{r}{\sqrt{3}}, \frac{r}{\sqrt{3}})$; $(-r\sqrt{\frac{2}{3}}, \frac{r}{\sqrt{3}}, -\frac{r}{\sqrt{3}})$,$(-r\sqrt{\frac{2}{3}}, -\frac{r}{\sqrt{3}}, \frac{r}{\sqrt{3}})$.
- the objective: $f(\pm r\sqrt{\frac{2}{3}}, \frac{r}{\sqrt{3}}) = \frac{2r^3}{3\sqrt{3}}$; $f(r\sqrt{\frac{2}{3}}, \pm\frac{r}{\sqrt{3}}) = -\frac{2r^3}{3\sqrt{3}}$; $f(0, \pm r) = 0$.

Inequality constraints



(Source: Pattern Recognition and Machine Learning by Chris Bishop)



- Suppose a constraint $g(x) \geq 0$. Then ∇g on the border points "inside" the feasible region.
- If maximizing, (unless the global max is inside the feasible region), ∇f must point outside, in opposite direction to ∇g, i.e.$\nabla f = -\lambda \nabla g$ for $\lambda > 0$.
- KKT conditions formalize Lagrangian multipliers to inequality constraints.

Reference

More Material:

- https://pages.hmc.edu/ruye/MachineLearning/lectures/ch3/node13.html
- https://www.cs.toronto.edu/~mbrubake/teaching/C11/Handouts/LagrangeMultipliers.pdf
- https://ocw.mit.edu/courses/18-02sc-multivariable-calculus-fall-2010/ebbeb8e61827a8058d2c45b674d003b3_MIT18_02SC_notes_22.pdf
- Convex optimization by Steph Boyd: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf
- Pattern Recognition and Machine Learning by Chris Bishop

Lecture 4 Bayesian learning and variational inference.

Bayesian Learning: Choosing priors

Frequentist vs bayesian

Problem with maximum likelihood

- Sometimes the maximum likelihood estimate of the parameters of a statistical models lead to un-intuitive results.
- e.g. coin toss: estimate "fairness" of a coin
  - Observe T throws, modeled with observations $x_1, \dots, x_T$ either 1 or 0, of a random variable X.
  - Estimate w, the parameter of a binomial distribution: $p(X = +1) = w$.
  - The ML estimate is then $w_{ML} = \frac{\sum_t x_t}{T}$.
  - Say you have the results of 3 throws, observing 3 times 1 will make you conclude that $w = 1$ and all the future will for sure be 1.
  - it could be a coincidence, but your estimation does not take into account your *prior* knowledge about the problem.

$\Longrightarrow$ there are ways to incorporate apriori knowledge in a parameter estimation problem, one such way is called Bayesian learning.

Bayesian Learning: Choosing priors

Frequentist vs bayesian

Problem with maximum likelihood

When learning from data $\underline{\mathbf{x}}$

- the ML estimate formulated as

$$w_{ML} = \arg\max_{w} p(\underline{\mathbf{x}}|W = w) \tag{12}$$

- can be replaced with

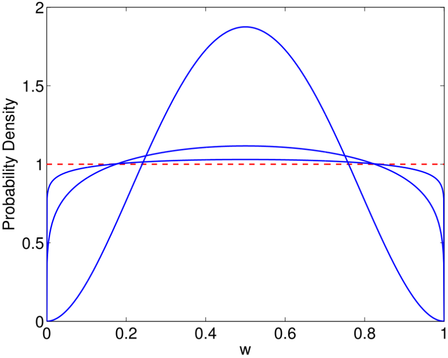
$$w_{MAP} = \arg\max_{w} p(W = w|\underline{\mathbf{x}}) \propto p(\underline{\mathbf{x}}|W = w)p(W = w), \tag{13}$$

  provided that we formulate our apriori knowledge as a density $p(W = w)$, (e.g. a uniform, Gaussian, ...).

Bayesian Learning: Choosing priors

Different kinds of priors

- **Subjective informative prior**: Our belief inform the statistical model for W.
- **Subjective non-informative prior**: Our belief is that we don't have any information about a parameter W.
  - Say that the new parameter U is related to W as $U = g(W)$. One must make sure to have a uniform prior also on U.
  - How to do this ?
- **Objective non-informative prior**: Jeffreys prior is a unique way to define a non-informative prior, which is the same regardless of the choice of g.



Bayesian Learning: Choosing priors

Jeffreys prior

Prior invariance

- Suppose a statistical model $p(X|W)$, a principle to choose a prior gives you $p(W)$.
- Suppose you re-parameterize W as $U = g(W)$. This gives a new statistical model $p(X|U)$. Applying the same principle to obtain a prior gives you $p(U)$.
- Instead of reparameterizing W, you reparameterize the prior $p(W)$. This gives yet another prior

$$\forall u \in \mathcal{U} \quad \bar{p}(u) = p(g^{-1}(u))|g'(g^{-1}(u))|^{-1}.$$

- Prior invariance means that

$$\begin{aligned}\forall u \in \mathcal{U} \quad p(u) &= \bar{p}(g(w)) \\ &= p(w)|g'(w)|^{-1},\end{aligned}$$

i.e. a prior obtained applying a principle should remain the same after transformation.

Bayesian Learning: Choosing priors

Jeffreys prior

Fisher information

- if the prior on W is defined according to Jeffreys principle:

$$\forall w \in \mathcal{W} \quad p(W=w) \propto \sqrt{\det I(w)},$$

  where $I(w)$ is the fisher information matrix, $\forall (i,j) \in [K]^2$

$$\begin{aligned} I_{ij}(w) &= E_{p(X|W=w)}\left[\left(\frac{\partial \ln p(X|W=w)}{\partial w_i}\right)\left(\frac{\partial \ln p(X|W=w)}{\partial w_j}\right)\right] \\ &= -E_{p(X|W=w)}\left[\frac{\partial^2 \ln p(X|W=w)}{\partial w_i \partial w_j}\right] \end{aligned}$$

- then the prior is invariant

Bayesian Learning: Choosing priors

Jeffreys prior

Fisher information

Let's prove the equality in the definition for a parameter $w \in \mathbb{R}$

$$I(w) = E_{p(X|W=w)}\left[\left(\frac{\partial \ln p(X|W=w)}{\partial w}\right)\left(\frac{\partial \ln p(X|W=w)}{\partial w}\right)\right]$$
$$= -E_{p(X|W=w)}\left[\left(\frac{\partial^2 \ln p(X|W=w)}{\partial w^2}\right)\right].$$

This is true because $\forall w \in \mathcal{W}$:

$$\frac{\partial^2 \ln p(X|W=w)}{\partial w^2} = \frac{\partial}{\partial w}\left[\frac{\partial \ln p(X|W=w)}{\partial w}\right]$$
$$= \frac{\frac{\partial^2 p(X|w)}{\partial w^2}}{p(X|w)} - \left(\frac{\frac{\partial p(X|w)}{\partial w}}{p(X|w)}\right)^2$$

Bayesian Learning: Choosing priors

Jeffreys prior

Fisher information

Also,

$$E_{p(X|W=w)}\left[\frac{\frac{\partial^2 p(X|w)}{\partial w^2}}{p(X|w)}\right] = \int p(X=x|w)\frac{\frac{\partial^2 p(X=x|w)}{\partial w^2}}{p(X=x|w)}dx = \int \frac{\partial^2 p(X=x|w)}{\partial w^2}dx$$

$$= \frac{\partial^2}{\partial w^2}\int p(X=x|w)dx = \frac{\partial^2}{\partial w^2}1 = 0$$

thus

$$E_{p(X|W=w)}\left[\frac{\partial^2 \ln p(X|W=w)}{\partial w^2}\right] = -E_{p(X|W=w)}\left[\left(\frac{\frac{\partial p(X|w)}{\partial w}}{p(X|w)}\right)^2\right]$$

$$= -E_{p(X|W=w)}\left[\left(\frac{\partial \ln p(X|w)}{\partial w}\right)^2\right]$$

Bayesian Learning: Choosing priors

Jeffreys prior

Applications

What is the Jeffreys prior density for the standard deviation parameter of a Gaussian density with unknown mean and standard dev.? llh: $\ln f(X|\mu,\sigma) = -\ln\sigma - \frac{(X-\mu)^2}{2\sigma^2} + \text{cst}$
Let's calculate the Fisher information:

$$\begin{aligned}
I_{11}(\mu,\sigma) &= -E_{p(X|\mu,\sigma)}\left[\frac{\partial^2 \ln p(X|\mu,\sigma)}{\partial\mu^2}\right] = \frac{1}{\sigma^2} \\
I_{22}(\mu,\sigma) &= -E_{p(X|\mu,\sigma)}\left[\frac{\partial^2 \ln p(X|\mu,\sigma)}{\partial\sigma^2}\right] \\
&= E_{p(X|\mu,\sigma)}\left[-\frac{1}{\sigma^2} + \frac{3X^2}{\sigma^4}\right] \\
&= -\frac{1}{\sigma^2} + \frac{3E_{p(X|\mu,\sigma)}[(X-\mu)^2]}{\sigma^4} = -\frac{1}{\sigma^2} + \frac{3\sigma^2}{\sigma^4} = \frac{2}{\sigma^2} \\
I_{12}(\mu,\sigma) = I_{21}(\mu,\sigma) &= -E_{p(X|\mu,\sigma)}\left[\frac{\partial^2 \ln p(X|\mu,\sigma)}{\partial\mu\partial\sigma}\right] = 0
\end{aligned}$$

Bayesian Learning: Choosing priors

Jeffreys prior

Applications

What is the Jeffreys prior density for the standard deviation parameter of a Gaussian density with zero mean? $\ln p(X|\mu,\sigma) = -\ln\sigma - \frac{(X-\mu)^2}{2\sigma^2} + \text{cst}$.
The Jeffreys prior is

$$p(\mu,\sigma) \propto \sqrt{\det I(\mu,\sigma)} \propto \frac{1}{\sigma^2}$$

which is not a proper density function (does integrate to 1, cannot be normalized).

Bayesian Learning: Choosing priors

Jeffreys prior

Applications

- However, the joint prior can be used as the asymptote of another distribution:

$$p(\mu, \sigma) = p_1(\mu|\sigma)p_2(\sigma) = \frac{\sqrt{\beta}}{\sqrt{2\pi\sigma}} e^{\frac{\mu^2\beta}{2\sigma^2}} . \frac{(b\sigma)^a}{\Gamma(a)} \frac{1}{\sigma} e^{-b\sigma},$$

  the Normal-Gamma$(\mu, \sigma, \beta, a, b)$, when $\beta, a, b \to 0$.
- The Normal-Gamma distribution is a *conjugate prior* for the joint prior $p(\mu, \sigma)$,
- i.e. given a likelihood, $p(X|\mu, \sigma)$, $p(\mu, \sigma)$ is Normal-Gamma $\implies$ the posterior $p(\mu, \sigma|X)$ is also Normal-Gamma. This is convenient and so we always try to choose a conjugate prior for the likelihood.

Variational Inference (VI)

Motivation

- Suppose a Latent variable model $p(X, S)$.
- What if the posteriors cannot be written in closed form ?
- Then we make a model for it: $q(S|X)$, or simply $q(S)$.
- And we learn that model by minimizing $D_{KL}(q(S|X)||p(S|X))$ wrt. $q(S|X)$.
- KL divergence ?

Variational Inference (VI)

KL divergence

KL divergence

The Kullback–Leibler (KL) divergence measures how a probability distribution q is different from a another distribution p.

- q and p must have the same support $\mathcal{X}$.
- $\forall x \in \mathcal{X}\ p(x) = 0 \implies q(x) = 0$

Definition for a random variable $X \in \mathcal{X}$:

$$D_{KL}(q(X)||p(X)) = E_q\left[\ln \frac{q}{p}\right] = \int_{\mathcal{X}} q(x) \ln \frac{q(x)}{p(x)} dx$$

Questions:

1. What is $D_{KL}(\mathcal{U}(a,b)||\mathcal{U}(c,d))$?
2. What is $D_{KL}(\mathcal{N}(\mu_p, \Sigma_p)||\mathcal{N}(\mu_q, \Sigma_q))$, both in k-dimension?

Variational Inference (VI)

KL divergence

$$D_{KL}(q(X)||p(X)) = E_q\left[\ln \frac{q}{p}\right] = \int_{\mathcal{X}} q(x) \ln \frac{q(x)}{p(x)} dx$$

KL divergence between uniform distributions

Let $p(X) = \mathcal{U}(a, b)$, $q(X) = \mathcal{U}(c, d)$. Assumptions on a, b, c, d ? $[a, b] \subseteq [c, d]$.

$$\begin{aligned} D_{KL}(p||q) &= \int_a^b p(x) \ln \frac{p(x)}{q(x)} dx \\ &= \int_a^b \frac{1}{b-a} \ln \frac{d-c}{b-a} dx \\ &= \frac{1}{b-a} \ln \frac{d-c}{b-a} \left[\int_a^b dx\right] \\ &= \ln \frac{d-c}{b-a} \end{aligned}$$

Variational Inference (VI)

KL divergence

$$D_{KL}(q(X)||p(X)) = E_q\left[\ln \frac{q}{p}\right] = \int_{\mathcal{X}} q(x) \ln \frac{q(x)}{p(x)} dx$$

KL divergence between Normal distributions

Let $p(x) = \mathcal{N}(x; \mu_p, \Sigma_p),\ q(x) = \mathcal{N}(x; \mu_q, \Sigma_q)$.
Recall, $p(x) = \frac{1}{(2\pi)^{k/2}|\Sigma_1|^{1/2}} e^{-\frac{1}{2}(x-\mu_p)^T \Sigma_p^{-1}(x-\mu_p)}$.

$$D_{KL}(p||q) = E_p[\ln p - \ln q]$$

$$\begin{aligned} D_{KL}(p||q) &= E_p\left[\frac{1}{2}\ln\frac{|\Sigma_p|}{|\Sigma_q|} - \frac{1}{2}(x-\mu_p)^T\Sigma_p^{-1}(x-\mu_p) + \frac{1}{2}(x-\mu_q)^T\Sigma_q^{-1}(x-\mu_q)\right] \\ &= \frac{1}{2}\left[\ln\frac{|\Sigma_p|}{|\Sigma_q|} - E_p\left[(x-\mu_p)^T\Sigma_p^{-1}(x-\mu_p)\right] + E_p\left[(x-\mu_q)^T\Sigma_q^{-1}(x-\mu_q)\right]\right] \end{aligned}$$

Variational Inference (VI)

KL divergence

KL divergence between Normal distributions

$$D_{KL}(p||q) = \frac{1}{2}\left[\ln\frac{|\Sigma_p|}{|\Sigma_q|} - \underbrace{E_p\left[(x-\mu_p)^T\Sigma_p^{-1}(x-\mu_p)\right]}_{(1)} + \underbrace{E_p\left[(x-\mu_q)^T\Sigma_q^{-1}(x-\mu_q)\right]}_{(2)}\right]$$

$$(1) \quad (x-\mu_p)^T\Sigma_p^{-1}(x-\mu_p) \in \mathbb{R}, \text{ thus } = \text{tr}((x-\mu_p)(x-\mu_p)^T\Sigma_p^{-1})$$

$$E_p[\dots] = \text{tr}(E_p\left[(x-\mu_p)(x-\mu_p)^T\right]\Sigma_p^{-1}) = \text{tr}(I_k) = k$$

$$(2) \quad E_p\left[(x-\mu_q)^T\Sigma_q^{-1}(x-\mu_q)\right] = (\mu_p-\mu_q)^T\Sigma_q^{-1}(\mu_p-\mu_q) + \text{tr}(\Sigma_q^{-1}\Sigma_p)$$

Eq. 380 in Matrix Cookbook

Finally: $D_{KL}(p||q) = \frac{1}{2}\left[\ln\frac{|\Sigma_p|}{|\Sigma_q|} - k + (\mu_p-\mu_q)^T\Sigma_q^{-1}(\mu_p-\mu_q) + \text{tr}(\Sigma_q^{-1}\Sigma_p)\right]$

Variational Inference (VI)

Minimizing KL divergence

- What if the posteriors cannot be written in closed form ?
- then we make a model for it: $q(S|X)$, or simply $q(S)$.
- and we learn that model by minimizing $D_{KL}(q(S|X)||p(S|X))$ wrt. $q(S|X)$.
- How do we do that computationally ? We said we couldn't write the true posterior in closed form ? Let's look at the KL divergence more in details.

$$
\begin{aligned}
D_{KL}(q(S|X)||p(S|X)) &= \sum_s q(S=s|X) \ln \frac{q(S=s|X)}{p(S=s|X)} \\
&= \sum_s q(S=s|X) \ln \frac{q(S=s|X)p(X)}{p(X|S)p(S)} \\
&= \sum_s q(S=s|X) \left[\ln \frac{1}{p(X|S)} + \ln \frac{q(S=s|X)}{p(S)} + \ln p(X) \right]
\end{aligned}
$$

Variational Inference (VI)

ELBO

$$D_{KL}(q(S|X)||p(S|X)) = \sum_s q(S=s|X)\left[\ln\frac{1}{p(X|S=s)} + \ln\frac{q(S=s|X)}{p(S=s)} + \ln p(X)\right]$$
$$= -E_{q(S|X)}[\ln p(X|S)] + D_{KL}[q(S|X)||p(S)] + \ln p(X)$$

Finally:

$$\ln p(X) = \underbrace{D_{KL}(q(S|X)||p(S|X))}_{\geq 0} \underbrace{-D_{KL}[q(S|X)||p(S)] + E_{q(S|X)}[\ln p(X|S)]}_{\mathcal{L}_{ELBO}(q)}$$

$$\ln p(X) \geq \mathcal{L}_{ELBO}(q)$$

Variational Inference (VI)

ELBO

$$\ln p(X) \geq \mathcal{L}_{ELBO}(q) = -D_{KL}[q(S|X)||p(S)] + E_{q(S|X)}[\ln p(X|S)]$$

- Maximizing the ELBO, minimizes $D_{KL}(q(S|X)||p(S|X))$, and learns to approximate the posterior distribution
- The ELBO can also be expressed as follows:

$$\begin{aligned}
\mathcal{L}_{ELBO}(q) &= -D_{KL}[q(S|X)||p(S)] + E_{q(S|X)}[\ln p(X|S)] \\
&= E_{q(S|X)}\left[\ln \frac{p(S)}{q(S|X)}\right] + E_{q(S|X)}[\ln p(X|S)] \\
&= E_{q(S|X)}\left[\ln \frac{p(X,S)}{q(S|X)}\right] \\
&= E_{q(S|X)}\left[\ln p(X,S)\right] - E_{q(S|X)}\left[\ln q(S|X)\right]
\end{aligned}$$

Variational Inference (VI)

Coordinate Ascent Variational Inference (CAVI)

- We are approximating the posterior distribution with a distribution $q(S|X)$, we are free to choose it's form.
- A simple one is the mean field approximation:

$$p(\underline{\mathbf{S}}|\underline{\mathbf{X}}) \approx q(\underline{\mathbf{S}}|\underline{\mathbf{X}}) = q(\underline{\mathbf{S}}) = q_1(\mathbf{S}_1)\ldots q_T(\mathbf{S}_T) = q_1 \ldots q_T$$

- The q_t factors are learnt one by one, let's optimize $t = i$.

$$\begin{aligned}
\mathcal{L}_{ELBO} &= E_{q(\mathbf{S})}\left[\ln p(\underline{\mathbf{X}}, \underline{\mathbf{S}})\right] - E_{q(\underline{\mathbf{S}})}\left[q(\underline{\mathbf{S}}|\underline{\mathbf{X}})\right] \\
&= E_{q(\underline{\mathbf{S}})}\left[\ln p(\underline{\mathbf{X}}, \underline{\mathbf{S}})\right] - E_{q(\underline{\mathbf{S}})}\left[\sum_t^T \ln q_t\right] \\
&= E_{q(\underline{\mathbf{S}})}\left[\ln p(\underline{\mathbf{X}}, \underline{\mathbf{S}})\right] - E_{q_1\ldots q_T}\left[\sum_t^T \ln q(\mathbf{S}_t)\right] \\
&= E_{q(\underline{\mathbf{S}})}\left[\ln p(\underline{\mathbf{X}}, \underline{\mathbf{S}})\right] - \sum_t^T E_{q_1\ldots q_T}\left[\ln q_t\right]
\end{aligned}$$

Variational Inference (VI)

Coordinate Ascent Variational Inference (CAVI)

- We are approximating the posterior distribution with a distribution $q(S|X)$, we are free to choose it's form.
- A simple one is the so-called mean field approximation:

$$p(\underline{\mathbf{S}}|\underline{\mathbf{X}}) \approx q(\underline{\mathbf{S}}|\underline{\mathbf{X}}) = q(\underline{\mathbf{S}}) = q_1(\mathbf{S}_1)\dots q_T(\mathbf{S}_T) = q_1 \dots q_T$$

- The q_t factors are learnt one by one, let's optimize $t = i$.

$$\begin{aligned}\mathcal{L}_{ELBO} = \dots &= E_{q(\underline{\mathbf{S}})}\left[\ln p(\underline{\mathbf{X}}, \underline{\mathbf{S}})\right] - \sum_t^T E_{q_t}\left[\ln q_t\right] \\ &= E_{q_1 \dots q_T}\left[\ln p(\underline{\mathbf{X}}, \mathbf{S}_1, \dots, \mathbf{S}_T)\right] - E_{q_i}\left[\ln q_i\right] + C \\ &= E_{q_i}\left[E_{q_j \neq i}\left[\ln p(\underline{\mathbf{X}}, \mathbf{S}_i, \mathbf{S}_{j\neq i})\right]\right] - E_{q_i}\left[\ln q_i\right] + C\end{aligned}$$

$$\text{Let } \ln \tilde{p}(\mathbf{S}_i) = E_{q_j \neq i}\left[\ln p(\underline{\mathbf{X}}, \mathbf{S}_i, \mathbf{S}_{j\neq i})\right] + cst$$

Variational Inference (VI)

Coordinate Ascent Variational Inference (CAVI)

- The q_t factors are learnt one by one, let's optimize $t = i$.

$$\begin{aligned}\mathcal{L}_{ELBO} &= \cdots = E_{q_i}\left[\ln \tilde{p}(\mathbf{S}_i)\right] - E_{q_i}\left[\ln q_i\right] + C \\ &= -D_{KL}\left(q_i || \tilde{p}(\mathbf{S}_i)\right) + C\end{aligned}$$

The divergence is minimized ($\mathcal{L}_{ELBO}$ maximized) when $q_i = \tilde{p}(\mathbf{S}_i)$, i.e.

$$q_i^* \propto \exp\left(E_{q_{j \neq i}}\left[\ln p(\underline{\mathbf{X}}, \mathbf{S}_i, \mathbf{S}_{j \neq i})\right]\right)$$

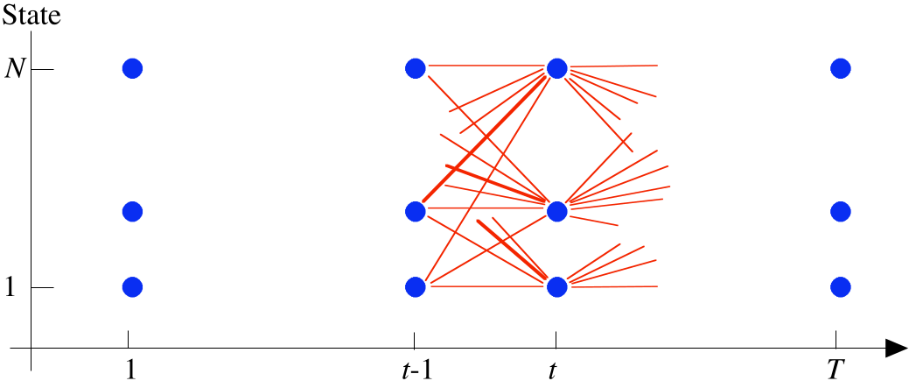
References

Based on *Pattern Recognition Fundamental Theory and Exercise Problems* by ARNE LEIJON & GUSTAV EJE HENTER

- `https://brunomaga.github.io/Variational-Inference-GMM`

Lecture 5 Viterbi

Viterbi decoding



(Source: Figure 5.14)

Viterbi

Decoding a sequence of observed variables

- Finding the best value of the latent sequence

$$
\begin{aligned}
(\widehat{i_1 \cdots i_T}) &= \arg \max_{(i_1 \cdots i_T)} P[\mathbf{S}_1 = i_1, \ldots, \mathbf{S}_T = i_T \mid \mathbf{x}_1, \ldots, \mathbf{x}_T, \lambda] \\
&= \arg \max_{(i_1 \cdots i_T)} \frac{P[\mathbf{S}_1 = i_1, \ldots, \mathbf{S}_T = i_T, \mathbf{x}_1, \ldots, \mathbf{x}_T \mid \lambda]}{P[\mathbf{x}_1, \ldots, \mathbf{x}_T \mid \lambda]} \\
&= \arg \max_{(i_1 \cdots i_T)} P[\mathbf{S}_1 = i_1, \ldots, \mathbf{S}_T = i_T, \mathbf{x}_1, \ldots, \mathbf{x}_T \mid \lambda] \\
&= \arg \max_{(i_1 \cdots i_T)} \log P[\mathbf{S}_1 = i_1, \ldots, \mathbf{S}_T = i_T, \mathbf{x}_1, \ldots, \mathbf{x}_T \mid \lambda]
\end{aligned}
$$

This is saying that the sequence of states which maximizes the posterior distribution, also maximizes the log-joint distribution.

Viterbi encoding

- We define the Viterbi variable:

$$\chi_{j,t} = \max_{(i_1,\ldots,i_{t-1})} P[\mathbf{S}_1 = i_1, \ldots, \mathbf{S}_{t-1} = i_{t-1}, \mathbf{S}_t = j, \mathbf{x}_1, \ldots, \mathbf{x}_t | \lambda]$$

- The probability that the best path ends in j at time t after having observed $\mathbf{x}_t$.

It can be computed recursively !

$$\begin{aligned}
\chi_{j,t} &= \max_{(i_1,\ldots,i_{t-1})} P[\mathbf{S}_t = j, \mathbf{x}_t | i_1, \ldots, i_{t-1}, \mathbf{x}_1, \ldots, \mathbf{x}_{t-1}, \lambda] P[i_1, \ldots, i_{t-1}, \mathbf{x}_1, \ldots, \mathbf{x}_{t-1} | \lambda] \\
&= \max_{i_{t-1}} P[\mathbf{S}_t = j, \mathbf{x}_t | \mathbf{S}_{t-1} = i_{t-1}, \lambda] \max_{(i_1,\ldots,i_{t-2})} P[i_1, \ldots, \mathbf{S}_{t-1} = i_{t-1}, \mathbf{x}_1, \ldots, \mathbf{x}_{t-1} | \lambda] \\
&= \max_{i} P[\mathbf{x}_t | \mathbf{S}_t = j, \lambda] P[\mathbf{S}_t = j | \mathbf{S}_{t-1} = i] \chi_{i,t-1} \\
&= P[\mathbf{x}_t | \mathbf{S}_t = j, \lambda] \max_{i} a_{i,j}\, \chi_{i,t-1}
\end{aligned}$$

Viterbi decoding

- Viterbi variable:

$$\chi_{j,t} = \max_{(i_1,\ldots,i_{t-1})} P[\mathbf{S}_1 = i_1, \ldots, \mathbf{S}_{t-1} = i_{t-1}, \mathbf{S}_t = j, \mathbf{x}_1, \ldots, \mathbf{x}_t | \lambda]$$

- Probability of the best path ending in state j after having observed $\mathbf{x}_t$ at time t.

After iterating up to time $t = T$:

- When we have computed the variable up to time T,
- $\max_j \chi_{j,T}$ is the value of the joint probability of the best sequence of states.
- However, we want the sequence of states it self

At $t = T$:

- We can get $\hat{i}_T = \arg\max_j \chi_{j,T}$
- We decode the rest of the indices backwards, for $t = T-1, \ldots, 1$:

$$\hat{i}_t = \arg\max_i \chi_{i,t} a_{i,\hat{i}_{t+1}} = \arg\max_i \chi_{i,t} P[\mathbf{S}_{t+1} = \hat{i}_{t+1} | \mathbf{S}_t = i]$$

Viterbi

There is a factorization of the joint distribution that is useful:

$$
\begin{aligned}
&P[i_1, \ldots, i_t, \ldots, i_T, \mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T \mid \lambda] \\
&= P[i_{t+1}, \ldots, i_T, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_T \mid i_1, \ldots, i_t, \mathbf{x}_1, \ldots, \mathbf{x}_t, \lambda] \cdot P[i_1, \ldots, i_t, \mathbf{x}_1, \ldots, \mathbf{x}_t \mid \lambda] \\
&= P[i_{t+1}, \ldots, i_T, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_T \mid i_t, \lambda] \cdot P[i_1, \ldots, i_t, \mathbf{x}_1, \ldots, \mathbf{x}_t \mid \lambda]
\end{aligned}
$$

- Next we maximize

$$
\begin{aligned}
&\max_{(i_1 \cdots i_T)} P[i_1, \ldots, i_t, \ldots, i_T, \mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T \mid \lambda] \\
&= \max_{i_t} \max_{(i_1 \cdots i_{t-1})} \max_{(i_{t+1} \cdots i_T)} \underbrace{P[i_{t+1}, \ldots, i_T, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_T \mid i_t, \lambda]}_{f(i_t)} \cdot \underbrace{P[i_1, \ldots, i_t, \mathbf{x}_1, \ldots, \mathbf{x}_t \mid \lambda]}_{g(i_t)} \\
&= \max_{i_t} \left(\max_{(i_{t+1} \cdots i_T)} P[i_{t+1}, \ldots, i_T, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_T \mid i_t, \lambda] \right) \cdot \left(\max_{(i_1 \cdots i_{t-1})} P[i_1, \ldots, i_t, \mathbf{x}_1, \ldots, \mathbf{x}_t \mid \lambda] \right)
\end{aligned}
$$

Viterbi

- Maximization

$$\max_{(i_1 \cdots i_T)} P[i_1, \ldots, i_t, \ldots, i_T, \mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T \mid \lambda]$$

$$= \max_{i_t} \left(\max_{(i_{t+1} \cdots i_T)} P[i_{t+1}, \ldots, i_T, \mathbf{x}_{t+1}, \ldots, \mathbf{x}_T \mid i_t, \lambda] \right) \cdot \left(\max_{(i_1 \cdots i_{t-1})} P[i_1, \ldots, i_t, \mathbf{x}_1, \ldots, \mathbf{x}_t \mid \lambda] \right)$$

- Decoding:

$$\hat{i}_t = \arg\max_i \chi_{i,t}\, a_{i,\hat{i}_{t+1}}$$

Reference

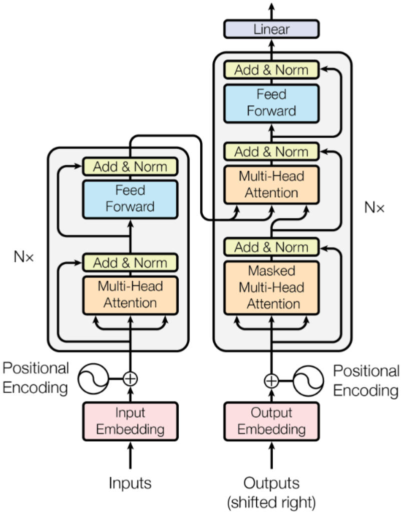
More Material:

- Pattern Recognition and Machine Learning by Chris Bishop
- https://www.cl.cam.ac.uk/teaching/1617/MLRD/slides/slides9.pdf

Lecture 6 Transformers

Transformers

General Architecture

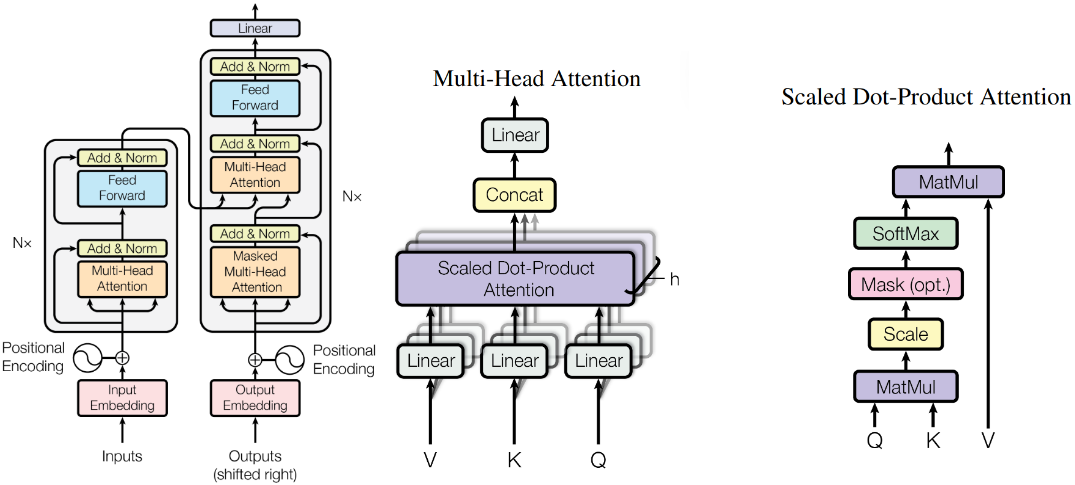


(Source: Vaswani et al.)

- From a sequence $X = [\mathbf{x}_1, \ldots, \mathbf{x}_T]^T \in \mathbb{R}^{T \times d}$ produces another sequence $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_T]^T \in \mathbb{R}^{T \times q}$.
- An encoding-decoding architecture for sequence to sequence tasks, i.e. there is an intermediate sequence: $\underline{\mathbf{z}} = [\mathbf{z}_1, \ldots, \mathbf{z}_T]$.
- Linear : $X' = XW \in \mathbb{R}^{T \times d'}$.
- Feed-Forward : $X' = \mathrm{MLP}(X) \in \mathbb{R}^{T \times d'}$.

Transformers

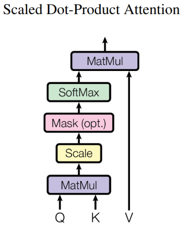
General Architecture



(Source: Vaswani et al.)

Transformers

Scaled Dot-product Attention


Scaled Dot-Product Attention

- $Q \in \mathbb{R}^{T\times D}, K \in \mathbb{R}^{T\times D}, V \in \mathbb{R}^{T\times q}$ are transforms of $X \in \mathbb{R}^{T\times d}$

$$Y = AV,\ \text{ with } A = \text{softmax}\left(\frac{QK^T}{\sqrt{D}}\right) \in \mathbb{R}^{T\times T}$$

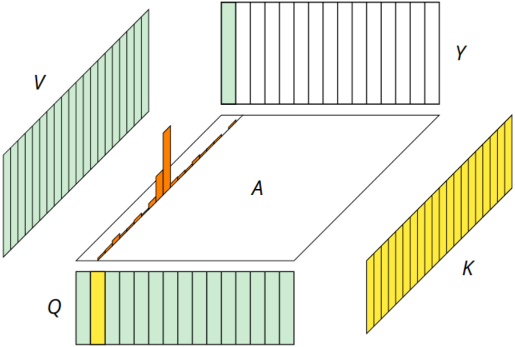
- $\forall i \in [T], \quad \mathbf{y}_i = \text{softmax}\left(\frac{\mathbf{q}_i K^T}{\sqrt{D}}\right) V = \sum_{j=1}^{T} \mathbf{v}_j \alpha_{i,j} \in \mathbb{R}^q$. The output is a weighted sum of the values.
- The attention weights are: $\alpha_{i,j} = \frac{e^{\mathbf{q}_i \mathbf{k}_j^T/\sqrt{D}}}{\sum_{j'} e^{\mathbf{q}_i \mathbf{k}_{j'}^T/\sqrt{D}}} = \frac{f(\mathbf{q}_i,\mathbf{k}_j)}{\sum_{j'} f(\mathbf{q}_i,\mathbf{k}_{j'})}$ with a kernel defined

  $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^D \times \mathbb{R}^D, \quad f(\mathbf{x}, \mathbf{y}) = e^{\mathbf{x}\mathbf{y}^T/\sqrt{D}} > 0.$
- Kernels (similarity) can be used to define conditional probabilities: $p(\mathbf{k}_j|\mathbf{q}_i) = \frac{f(\mathbf{q}_i,\mathbf{k}_j)}{\sum_{j'} f(\mathbf{q}_i,\mathbf{k}_{j'})}$.
- This means that $\forall i \in [T], \quad \mathbf{y}_i = \sum_j p(\mathbf{k}_j|\mathbf{q}_i)\mathbf{v}_j = E_{p(\mathbf{k}_j|\mathbf{q}_i)}[\mathbf{v}_j]$

Transformers

Scaled Dot-product Attention

Visually: Given a query sequence Q, a key sequence K, and a value sequence V, compute an attention matrix A by matching Qs to Ks, and weight V with it to get the sequence Y.



A big issue is that we have to represent matrix A in memory, making the memory footprint quadratic in T !

(Source: DLC, F. Fleuret)

Transformers

Linear Scaled Dot-product Attention

- The quadratic complexity issue can be addressed by replacing the softmax function (work by Fleuret et al.).
- Express the kernel f as a scalar product of some feature mapping $\varphi : \mathbb{R}^D \to \mathbb{R}^{D'}$

$$f(\mathbf{x}, \mathbf{y}) = \varphi(\mathbf{x})\varphi(\mathbf{y})^T$$

- This leads to

$$\mathbf{y}_i = \sum_j \frac{f(\mathbf{q}_i, \mathbf{k}_j)\mathbf{v}_j}{\sum_{j'} f(\mathbf{q}_i, \mathbf{k}_{j'})} = \frac{\varphi(\mathbf{q}_i) \sum_j \varphi(\mathbf{k}_j)^T \mathbf{v}_j}{\varphi(\mathbf{q}_i) \sum_{j'} \varphi(\mathbf{k}_{j'})^T}$$

- With the numerator in matrix form: $\left(\varphi(Q)\varphi(K)^T\right) V = \varphi(Q) \left(\varphi(K)^T V\right)$
- i.e. $\left(\varphi(K)^T V\right)$ is computed once and reused for every query, reducing the complexity from $O(T^2)$ to $O(T)$!
- The price to pay is that we only get an approximation of the softmax kernel.

Transformers

Positional encoding

- Position information is lost in transformers: invariance to row swaps in K and V
- Also, timestamp are in general unbounded, can differ from sequence to sequence
- PE's goal: Representing timestamps in high dimension D (an even number): $f(t) = [PE_1(t), PE_2(t), \dots]$

$$\begin{cases} PE_{2k}(t) = \sin\left(\frac{t}{L^{\frac{2k}{D}}}\right) \\ PE_{2k+1}(t) = \cos\left(\frac{t}{L^{\frac{2k}{D}}}\right) \end{cases}, k = 0, \dots, D/2 - 1$$

- A sin wave of frequency $f[Hz]$:

$$t \mapsto \sin(2\pi f t) = \sin(\omega t),$$

- i.e. positional encoding represents time in high dimension by sampling $D/2$ sine waves of increasing wavelength: $\omega_k = L^{2k/D}$, where L is the maximum frequency.

Transformers

Positional encoding

- Representing timestamps in high dimension D (an even number): $f(t) = [PE_1(t), PE_2(t), \ldots]$

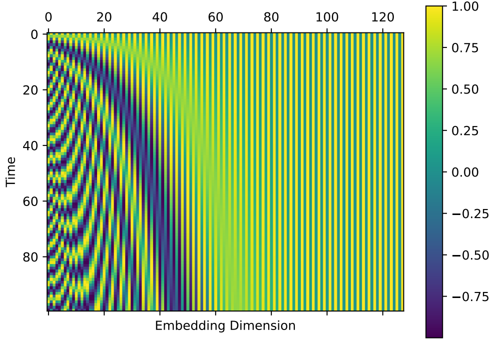
$$\begin{cases} PE_{2k}(t) = \sin\left(\frac{t}{L^{\frac{2k}{D}}}\right) \\ PE_{2k+1}(t) = \cos\left(\frac{t}{L^{\frac{2k}{D}}}\right) \end{cases}, k = 0, \ldots, D/2 - 1$$

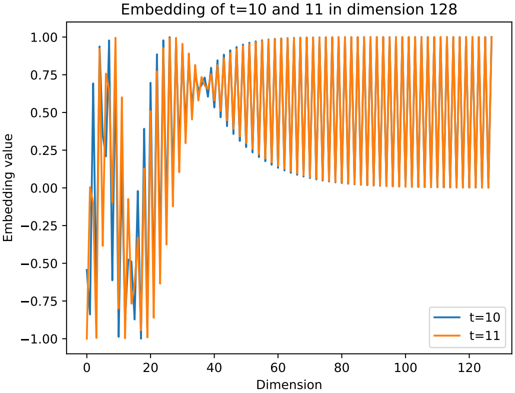
- Suppose that PE is used such that $Q = XW^Q + PE$ and $K = XW^K + PE$ where W^Q and W^K are two trainable linear transforms
- Question: Write the scalar product between a query at instant t and a key at instant t'.

Transformers

Positional encoding

Example with a time indices $t = 1, \dots, 100$, $L = 10000$, $D = 128$.





Applications

GPT

- Used as a generative model for time series $\underline{\mathbf{x}}$

$$p(\underline{\mathbf{x}}) = \prod_{t=1}^{T} p(\mathbf{x}_t | \mathbf{x}_{t-1}, \dots, \mathbf{x}_1)$$

- The pretraining loss of GPT models is the log-likelihood!
- Question : Draw this joint distribution
- Question : in this case what is the output time series ?
- $\mathbf{y}_t = p(\mathbf{x}_t | \mathbf{x}_{t-1}, \dots, \mathbf{x}_1)$

Applications

seq2seq

- Example sequence to sequence task.

References

- Vaswani et al. https://arxiv.org/pdf/1706.03762v5.pdf
- Blog post on positional encoding: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1
- Deep learning course by F. Fleuret https://fleuret.org/public/EN_20220809-Transformers/transformers-slides.pdf
- Linear transformers by F. Fleuret et al. https://proceedings.mlr.press/v119/katharopoulos20a.html

Lecture 7: Variational Auto-encoders

Architecture

Motivation

- There are multiple ways to represent data (e.g. colors: RGB, HSV, HSL, CMYK,...).
- Going from one way to another is called encoding.
- Decoding means going back to the previous representation.

Questions:

1. Why is it interesting to represent data differently ?
   - For doing something with it, i.e. for downstream tasks: transmission, inference. Certain ways to represent data are more efficient.
   - Typically there are redundancy in raw signals (e.g. images, speech)

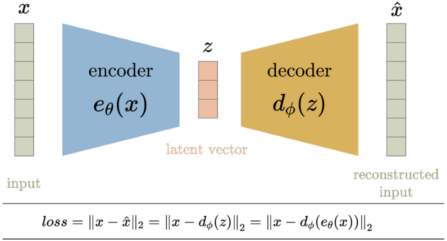
- It's not always clear what's the best representation for a particular downstream task.
- Best to learn it !

Architecture

Auto-encoders

- A compressed version of the data is an interesting representation
- Especially when the downstream task is unspecified.
- Auto-encoding is one way to compress data.
- The compressed representation is also called a latent representation.

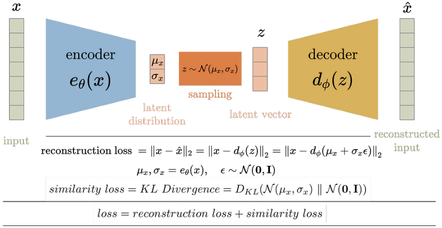
Auto-encoders



$$loss = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(e_\theta(x))\|_2$$

- Hard to control the structure of the latent space

Variational Auto-encoders



$$\text{reconstruction loss } = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(\mu_x + \sigma_x \epsilon)\|_2$$

$$\mu_x, \sigma_x = e_\theta(x), \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$similarity\ loss = KL\ Divergence = D_{KL}(\mathcal{N}(\mu_x, \sigma_x) \parallel \mathcal{N}(\mathbf{0}, \mathbf{I}))$$

$$loss = reconstruction\ loss + similarity\ loss$$

- Structures the latent space
- Can perform data generation

Architecture

Variational

- A variational auto-encoder can be seen as a latent variable model:

$$p(X, S) = p_\theta(X|S)p(S)$$

- Cannot be used directly to maximize $p(X)$.
- Questions:
  1. What is $p_\theta(X|S)$ called ?
     - The decoder
  2. What is $p(S)$ called ?
     - The latent distribution, or the prior.
  3. Another quantity is required (for inference), which one ?
     - The posterior (i.e. the encoder) $p(S|X)$
- The true encoder is unknown and so we approximate it with a distribution that we parameterize $q_\phi(S|X)$.
- We learn its parameters such that $D_{KL}(q_\phi(S|X)||p(S|X))$ is minimized.

ELBO

Formulation

- Remember how to deal with the minimization of $D_{KL}(q_\phi(S|X)||p(S|X))$?

$$D_{KL}(q_\phi(S|X)||p(S|X)) = \sum_s q_\phi(S=s|X) \ln \frac{q_\phi(S=s|X)}{p(S=s|X)}$$
$$= \sum_s q_\phi(S=s|X) \ln \frac{q_\phi(S=s|X)p(X)}{p(X|S)p(S)}$$
$$= \sum_s q_\phi(S=s|X) \left[\ln \frac{1}{p_\theta(X|S)} + \ln \frac{q_\phi(S=s|X)}{p(S)} + \ln p(X) \right]$$

- $$\ln p(X) = \underbrace{D_{KL}(q_\phi(S|X)||p(S|X))}_{\geq 0} \underbrace{- D_{KL}[q_\phi(S|X)||p(S)] + E_{q_\phi(S|X)}[\ln p_\theta(X|S)]}_{\mathcal{L}_{ELBO}(q)}$$

$$\ln p(X) \geq \mathcal{L}_{ELBO}(q)$$

ELBO

Formulation

- And so we end up maximizing

$$\mathcal{L}_{ELBO}(q) = -\underbrace{D_{KL}[q_\phi(S|X)||p(S)]}_{(1)} + \underbrace{E_{q_\phi(S|X)}[\ln p_\theta(X|S)]}_{(2)}$$

- Question: What is (2) ? The reconstruction loss.
- The computation of this term requires sampling:
    - but leads variance issues when differentiating the expectation directly.
- We resort to something called the reparameterization trick to compute the expectation

ELBO

Reparameterization trick

- Given a data point $\mathbf{x} \in \mathbb{R}^d$, the reparameterization trick is used

$$\mathbf{s} \sim q_\phi(S|X = \mathbf{x}) \Leftrightarrow \mathbf{s} = g_\phi(\boldsymbol{\epsilon}; \mathbf{x}), \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$$

- Question: example in $\mathbb{R}^D$ with $p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, I_D)$:

$$\begin{aligned} &\mathbf{s} \sim \mathcal{N}(\mathbf{s}; \boldsymbol{\mu}, \Sigma) \Leftrightarrow \\ &\mathbf{s} = \boldsymbol{\mu} + A\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}), \quad AA^T = \Sigma \end{aligned}$$

- The reconstruction loss is then approximated

$$\begin{aligned} E_{q_\phi(S|X=\mathbf{x})}[\ln p_\theta(X|S)] &= E_{p(\boldsymbol{\epsilon})}\left[\ln p_\theta(X|S = g_\phi(\boldsymbol{\epsilon}; \mathbf{x}))\right] \\ &\approx \frac{1}{L}\sum_{l=1}^{L} \ln p_\theta(X|S = g_\phi(\boldsymbol{\epsilon}^{(l)}; \mathbf{x})) \end{aligned}$$

with $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$.

ELBO

Reparameterization trick

- The reconstruction loss is $E_{q_\phi(S|X=\mathbf{x})}[\ln p_\theta(X|S)]$.
- Question: How does this relate to e.g. the mean squared error ?
  - The decoder is formulated with the reparameterization trick:

$$\text{for } \mathbf{s} \sim q_\phi\left(S|X=\mathbf{x}\right) \text{ we compute } \hat{\mathbf{x}} = h_\theta(\mathbf{s}) \text{ for some function } h_\theta,$$

  - The loss function is a Gaussian centered on the input $\mathbf{x} \in \mathbb{R}^d$

$$\text{i.e. } p_\theta(X=\hat{\mathbf{x}}|S=\mathbf{s}) = \mathcal{N}\left(h_\theta(\mathbf{s}); \mathbf{x}, \sigma^2 I_d\right)$$

  - Finally, the reconstruction loss:

$$E_{q_\phi(S|X=\mathbf{x})}[\ln p(X|S)] \approx \frac{1}{L}\sum_{l=1}^{L}\left[\text{cst} - \frac{1}{2\sigma^2}(h_\theta(\mathbf{s}^{(l)})-\mathbf{x})^T(h_\theta(\mathbf{s}^{(l)})-\mathbf{x})\right],$$

where $\mathbf{s}^{(l)} \sim q_\phi(S|X=\mathbf{x})$.

Prior Distributions

- Recall, we are maximizing

$$\mathcal{L}_{ELBO}(q) = -\underbrace{D_{KL}[q_\phi(S|X)||p(S)]}_{(1)} + \underbrace{E_{q_\phi(S|X)}[\ln p_\theta(X|S)]}_{(2)}$$

- We spoke about the reconstruction term in (2).
- Question: What about (1) ? What do we need to compute it ?
  - To specify the prior and variational distribution.
- The form of the variational distribution will depend on the prior

Prior Distributions

Gaussian Prior

- The original paper proposes a Gaussian prior, e.g. in $\mathbb{R}^D$: $p(S) = \mathcal{N}(\mathbf{0}, I_D)$
- In that case the encoder of a data point $\mathbf{x} \in \mathbb{R}^d$, is also a Gaussian:

$$q_\phi(S|X = \mathbf{x}) = \mathcal{N}(S; \mu_{\phi_1}(\mathbf{x}), \sigma^2_{\phi_2}(\mathbf{x}) I_D),$$

  where $\phi = \{\phi_1, \phi_2\}$ are parameters of neural networks for instance.
- Question: What is the expression of $D_{KL}[q_\phi(S|X = \mathbf{x})||p(S)]$?

$$D_{KL}[q_\phi(S|X = \mathbf{x})||p(S)] = \frac{1}{2}\sum_{i=1}^{D}(1 + \ln \sigma_i^2 - \mu_i^2 - \sigma_i^2),$$

  where μ_i is the i-th component of $\mu_{\phi_1}(\mathbf{x}) \in \mathbb{R}^D$.

Prior Distributions

More priors

- What other prior can be used ? Gaussian mixture models ! e.g. `https://arxiv.org/pdf/1611.02648`
- Model:

$$\left\{ p_{\boldsymbol{\beta},\boldsymbol{\theta}}(\mathbf{x},\mathbf{s},\mathbf{w},\mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}\mid\mathbf{s})p_{\boldsymbol{\beta}}(\mathbf{s}\mid\mathbf{w},\mathbf{z})p(\mathbf{w})\,p(\mathbf{z}) \right.$$

- Prior:

$$\begin{cases} \mathbf{w} \sim \mathcal{N}(0,\mathbf{I}) \\ \mathbf{z} \sim \mathrm{Mult}(\boldsymbol{\pi}) \\ \mathbf{s}\mid\mathbf{z},\mathbf{w} \sim \prod_{k=1}^{K} \mathcal{N}\left(\mu_{z_k}(\mathbf{w};\boldsymbol{\beta}),\, \mathrm{diag}\left(\sigma^2_{z_k}(\mathbf{w};\boldsymbol{\beta})\right)\right)^{z_k} \end{cases}$$

- Decoder:

$$\left\{ \mathbf{x}\mid\mathbf{s} \sim \mathcal{N}\left(\mu(\mathbf{s};\boldsymbol{\theta}),\, \mathrm{diag}\left(\sigma^2(\mathbf{s};\boldsymbol{\theta})\right)\right) \quad \text{or} \quad \mathcal{B}\left(\mu(\mathbf{s};\boldsymbol{\theta})\right) \right.$$

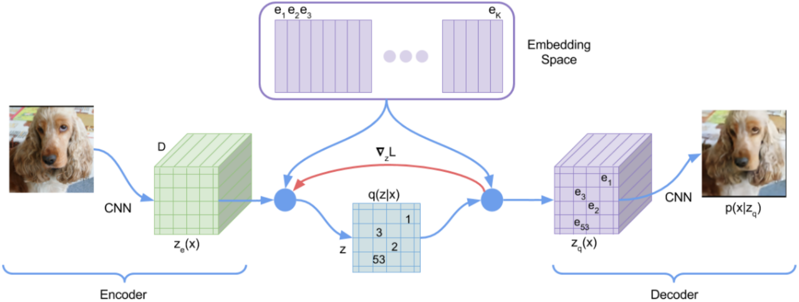
- $\mathcal{L}_{\text{ELBO}}$:

$$\left\{ \begin{aligned} & E_q\left[\frac{p_{\boldsymbol{\beta},\boldsymbol{\theta}}(\mathbf{x},\mathbf{s},\mathbf{w},\mathbf{z})}{q(\mathbf{s}|\mathbf{x},\mathbf{w},\mathbf{z})}\right] \\ & = \mathbb{E}_{q(\mathbf{s}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{s})\right] - \mathbb{E}_{q(\mathbf{w}|\mathbf{x})p(\mathbf{z}|\mathbf{s},\mathbf{w})}\left[\mathrm{KL}\left(q_{\phi_x}(\mathbf{s}|\mathbf{x}) \,\|\, p_{\boldsymbol{\beta}}(\mathbf{s}|\mathbf{w},\mathbf{z})\right)\right] \\ & \quad -\mathrm{KL}\left(q_{\phi_w}(\mathbf{w}|\mathbf{x}) \,\|\, p(\mathbf{w})\right) - \mathbb{E}_{q(\mathbf{s}|\mathbf{x})q(\mathbf{w}|\mathbf{x})}\left[\mathrm{KL}\left(p_{\boldsymbol{\beta}}(\mathbf{z}|\mathbf{s},\mathbf{w}) \,\|\, p(\mathbf{z})\right)\right]. \end{aligned} \right.$$

Prior Distributions

More priors

- What other prior can be used ? Discrete ! e.g. `https://arxiv.org/pdf/1711.00937`

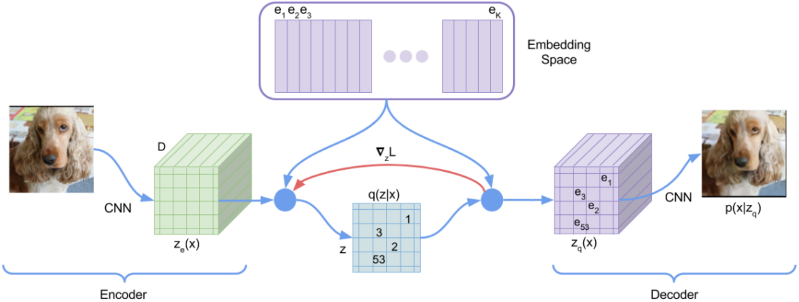


- Encoding: $q(z_k = 1|x) = \begin{cases} 1 & \text{for } k = k^* = \arg\min_j ||z_e(x) - e_j|| \\ 0 & \text{otherwise} \end{cases}$, and $z_q(x) = e_{k^*}$
- Deterministic (zero entropy)! With a uniform prior, constant KL divergence

Prior Distributions

More priors

- What other prior can be used ? Discrete ? e.g. `https://arxiv.org/pdf/1711.00937`



- Training: $Loss = \underbrace{\ln p(x|z_q(x))}_{(1)} + \underbrace{||\mathrm{sg}(z_e(x)) - e_{k^*}|| + \beta||z_e(x) - \mathrm{sg}(e_{k^*})||}_{(2)}$
- sg(.) is identity during forward, and cuts gradient during backward.
- (2) ensures that embeddings and encodings get closer during training.

References

- https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c
- Kingma et al. paper http://arxiv.org/abs/1312.6114
- Deep unsupervised clustering https://arxiv.org/pdf/1611.02648
- VQ-VAE https://arxiv.org/pdf/1711.00937
- More in details: http://arxiv.org/pdf/2410.06424

Lecture 8: Overall Recap

Q & A I

1. What is an HMM ?
   1.1 A statistical model for timeseries. Assuming observations (1) explained with corresponding latent variables modeled with a Markov chain in time, and (2) independent to each other given the corresponding latent variable.
2. Why is the EM algorithm required to learn the parameters of a hidden Markov model?
   2.1
3. In EM, why is an auxiliary function required ?
   3.1 Too computationally expensive to compute the evidence.
4. In the context of HMMs, what is $p(\underline{\mathbf{x}})$?
   4.1 Likelihood function of an observed sequence $\underline{\mathbf{x}}$.
5. In the context of HMMs, what is $p(\underline{\mathbf{x}}, \underline{\mathbf{s}})$ called ?
   5.1 The joint distribution of the observation and latent variables.
6. In the context of HMMs, how are $p(\underline{\mathbf{x}})$ and $p(\underline{\mathbf{x}}, \underline{\mathbf{s}})$ related ?
   6.1 The Joint distribution of the observation and latent variables.
7. In the context of HMMs, what is $p(\mathbf{s}_t | \underline{\mathbf{x}})$ for a time t.

Q & A II

7.1 The posterior distribution of the latent variable at time t given the observed data.

8. What is the difference between Bayesian and frequentists statistics ?

   8.1

9. What is the joint density function $f_{X_1,X_2}(x_1, x_2)$ of independent random variables X_1, X_2 ?

   9.1

10. What is the expected value of a random variable with a mixture of gaussian probability model ?

    10.1

11. What is the preferred model for a feature vector ?

    11.1 Random variables page 15

12. Write suppose three events A, B, C, write Bayes rule for the joint distribution $p(A, B|C)$.

    12.1

13. For binary classification, what decision rule should you use when there are much more data in one class ?

    13.1

Q & A III

14. In classification, what is a decision function ?
    14.1 returns a class index from data
15. What is are discriminant functions ?
    15.1 functions returning a real score for each class
16. What is a general form for a generative statistical model with latent variables
    16.1 p(X,S) = p(X,S)p(S)
17. What do we call the likelihood of data ?
    17.1
18. What's the difference between a fine-state and an infinite duration HMM ?
    18.1
19. What parameter estimation paradigm have we seen in the course ?
    19.1 Maximum likelihood and Bayesian learning
20. How can the parameters of a Markov chain be expressed ?
    20.1
21. What is a left-right HMM ?

Q & A IV

21.1

22. What does it mean to factorize a joint distribution ?

    22.1

23. What does the forward algorithm do ?

    23.1

24. What does the backward algorithm do ?

    24.1

25. What does the Viterbi algorithm do ?

    25.1

26. What is the difference between the Baum-Welch and the EM algorithm

    26.1

27. Describe the EM algorithm

    27.1

28. What are the convergence guarantees of the EM algorithm ?

    28.1

Q & A V

29. What is the difference between a subjective and objective uninformative prior ?
    29.1 the same up to a change of variable
30. What is the Jeffreys prior ?
    30.1
31. What is a conjugate probability distribution ?
    31.1
32. What is a conjugate probability distribution ?
    32.1
33. What is variational inference ?
    33.1
34. What is a conjugate probability distribution ?
    34.1