UNIVERSITY OF CAMBRIDGE

4F10: Latent Variable and Sequence Models

Mark Gales

Michaelmas 2021

Latent Variable Models

- Previous lecture examined relationships in data samples
  - conditional-independence important
- It is possible to introduce latent variables into the model
  - do not have to have any "meaning"
  - these variables are never observed in test (possibly in training)
  - marginalised over to get probabilities
  - discrete (mixture models, HMMs), continuous (factor-analysis)
- Revise/extend material from 3F8 & previous 4F10 lecture
  - simple language models
  - latent variable models
  - Expectation-Maximisation (EM)
  - hidden Markov models
  - conditional random fields

Language Modelling

Language Modelling

- Consider the word sequence

  The cat sat on the ???

  - language modelling can be viewed as answering

    What is the missing word ???

- Writing this in terms of probabilities

$$P(w_i|\hat{w}_1,\ldots,\hat{w}_{i-1}) = P(w_i|\hat{\mathbf{w}}_{1:i-1})$$

  - $\hat{\mathbf{w}}_{1:i-1} = \hat{w}_1,\ldots,\hat{w}_{i-1}$ observed words at positions 1 to $i-1$
  - w_i unknown word to be estimated (guessed) at position i

Language Modelling - Data Generation

- Predicted word can then be used as input for data generation
  - simple auto-regressive data generation process

$$\hat{w}_i \sim P(w_i|\hat{w}_1, \ldots, \hat{w}_{i-1}); \quad P(w_{i+1}|\hat{w}_1, \ldots, \hat{w}_i)$$

- Very simple process for generating data
  - initialise process with $w_0 = \texttt{<s>}$, sentence start symbol

<s> ???	$P(w_1\|\texttt{<s>})$
<s> The ???	$P(w_2\|\texttt{<s>}, \texttt{The})$
<s> The cat ???	$P(w_3\|\texttt{<s>}, \texttt{The}, \texttt{cat})$
<s> The cat sat ???	$P(w_4\|\texttt{<s>}, \texttt{The}, \texttt{cat}, \texttt{sat})$
<s> The cat sat on ???	$P(w_5\|\texttt{<s>}, \texttt{The}, \texttt{cat}, \texttt{sat}, \texttt{on})$

Language Modelling - Simple?

- How large would a language model be - consider
  - vocabulary size $|V|$: (number of possible words) $65536 = 2^{16}$
  - sentence length L: (length of sentence to generate) $8 = 2^3$

- How large does the model need to be to predict the last word:
  - number of possible histories (any sentence): $(|V|)^{L-1} = 2^{112}$
  - number of possible words (any sentence): $|V| = 2^{16}$
  - total number of parameters: $(|V|)^L = 2^{128} = 3.4 \times 10^{38}$

- Directly modelling all sentences not possible!
  - size of model dominated by number of histories
  - how to get a compact history representation?

N-Gram Language Models (Markov-Chain 3M1)

- Simplest approach to make history compact - truncate

$$P(w_i|\boldsymbol{w}_{1:i-1}) \approx P(w_i|\boldsymbol{w}_{i-N+1:i-1})$$

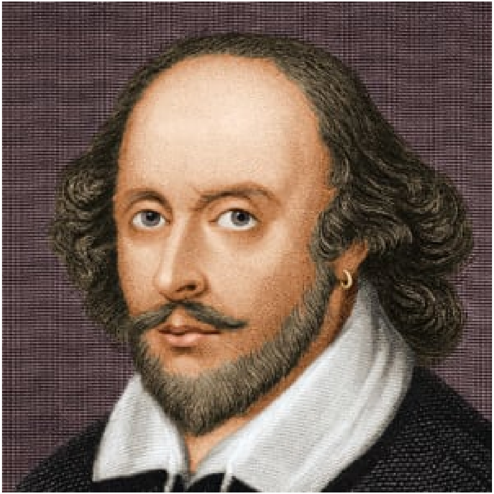
notation: $\boldsymbol{w}_{1:i-1}$ (rather than $\hat{\boldsymbol{w}}_{1:i-1}$) as consider all "histories"
  - N determines the length of the history to consider
  - $N = 1$ unigram, $N = 2$ bigram, $N = 3$ trigram ...

- A discrete model estimated from data

$$P(w_i|\boldsymbol{w}_{i-N+1:i-1}) \approx \frac{\mathcal{C}(w_{i-N+1}, \ldots, w_i)}{\mathcal{C}(w_{i-N+1}, \ldots, w_{i-1})}$$

  - $\mathcal{C}()$ counts number of occurences in the training data
  - "separate" probability computed for every word
  - general probability mass function (PMF) smoothing approaches

William Shakespeare (1564-1616)



Writing Shakespeare

- Unigram - No prior information
  - Every enter now severally so, let
  - Will rash been and by I the me loves gentle me not slavish page, the and hour; ill let
- Bigram - Current word
  - What means, sir. I confess she? then all sorts, he is trim, captain.
  - The world shall- my lord!
- Trigram - Current and previous word
  - Indeed the duke; and had a very good friend.
  - Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
- 4-gram - Current and previous two words
  - It cannot be but so.
  - Enter Leonato's brother Antonio, and the rest, but seek the weary beds of people sick.

Guardian Article - 9th September 2020

Opinion Artificial intelligence (AI)

A robot wrote this entire article. Are you scared yet, human?

GPT-3

- Deep-learning has revolutionised language modelling
  - GPT-3 is a large language model from OpenAI
  - about 175 billion model parameters trained on 500 billion tokens

- Architecure of model will be discussed later in course

Latent Variable Models

"Static" Latent Variable Generative Models

- N-gram language model use word sequence as context
  - causes exponential grow in size of model with history length
- Alternative: introduce a variable that represents the history
  - this variable may be discrete or continuous
  - just needs to represent the history in an appropriate form
- This section introduces latent variables
  - initially for "static" data - for example mixture models
  - the next section will discuss application to sequence models - for example HMMs

Gaussian Mixture Models

- Gaussian mixture models (GMMs) are based on Gaussians
  - form of the Gaussian distribution:

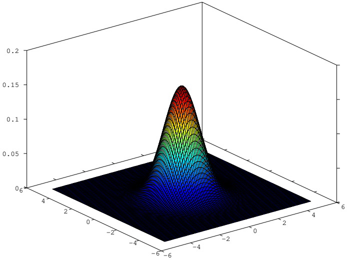
$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^{\mathsf{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- Each component modelled using a Gaussian distribution

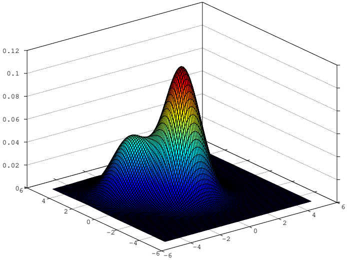
$$p(\boldsymbol{x}) = \sum_{m=1}^{M} P(c_m) p(\boldsymbol{x}|c_m) = \sum_{m=1}^{M} P(c_m) \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$$

  - component prior: $P(c_m)$
  - component distribution: $p(\boldsymbol{x}|c_m) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$
- Flexible model, able to model range of distributions

Gaussian Mixture Model Example



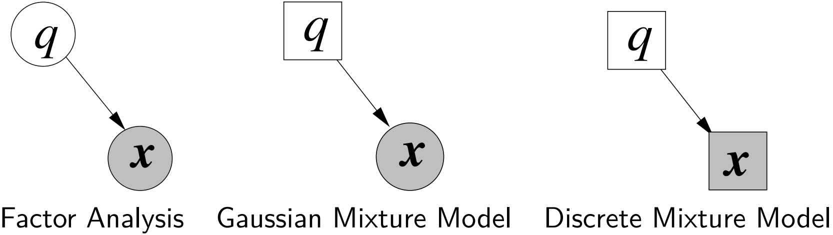
$\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$



$0.7\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) + 0.3\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma})$

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} -2 \\ 2 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Latent Variable Generative Models



- Three Bayesian Networks (BNs) for an observation $\boldsymbol{x}$
  - indicator variable $\boldsymbol{q}$: continuous $\boldsymbol{z}$ or discrete c_m space

factor analysis	$\int p(\boldsymbol{x}\|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}$
Gaussian mixture models	$\sum_{m=1}^{M} P(c_m)p(\boldsymbol{x}\|c_m)$
discrete mixture model	$\sum_{m=1}^{M} P(c_m)P(\boldsymbol{x}\|c_m)$

- Extensively used in many machine learning applications

Factor Analysis

- Two views of Factor Analysis (FA)
  - low-dimensional manifold representation
  - compact covariance matrix for multi-variate Gaussians

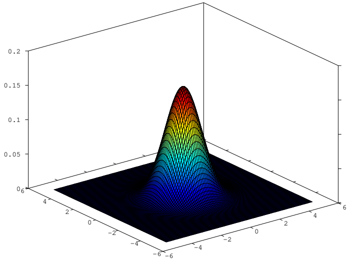
- Form of model dimensionality of $\boldsymbol{z}$ less than $\boldsymbol{x}$, $p < d$
  - $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I})$: low-dimensional subspace representation
  - $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{C}\boldsymbol{z}, \boldsymbol{\Sigma}_{\texttt{diag}})$:
    $\boldsymbol{C}$ loading matrix, $\boldsymbol{\Sigma}_{\texttt{diag}}$ diagonal covariance matrix
  - if $\boldsymbol{\Sigma}_{\texttt{diag}} = \sigma^2\boldsymbol{I}$ model is probabilistic PCA

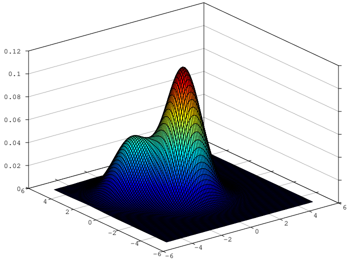
- As all elements Gaussian closed-form solution

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z} = \mathcal{N}(\boldsymbol{x}; \boldsymbol{0}, \boldsymbol{C}\boldsymbol{C}^{\mathsf{T}} + \boldsymbol{\Sigma}_{\texttt{diag}})$$

Expectation Maximisation

Gaussian Mixture Model Example





Parameter Estimation

- Would like to estimate latent variable model parameters, $\boldsymbol{\theta}$
  - training data $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
  - generative model: maximum (log-)likelihood

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \{\mathcal{L}(\boldsymbol{\theta})\} = \arg\max_{\boldsymbol{\theta}} \left\{ \sum_{i=1}^{N} \log\left(p(\boldsymbol{x}_i; \boldsymbol{\theta})\right) \right\}$$

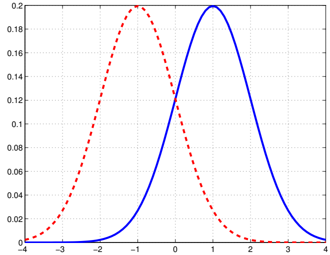
- Gradient descent could be used
  - need to determine learning rate ...

Is there any alternative?

- Use Gaussian Mixture Models as the example
  - discrete latent variable (for continuous see examples paper)

“Simple” Expectation Maximisation

- If you knew the component for each observation - simple
  - standard (multi-variate) Gaussian distribution
  - but you don’t - make an initial guess of the parameters $\boldsymbol{\theta}^{(0)}$



- set $k = 0$, using parameters $\boldsymbol{\theta}^{(k)}$

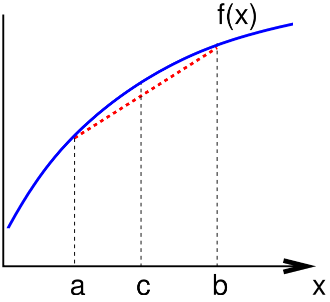
  “assign” each observation to a component

  use Bayes’ decision rule for assignment
- yields component id for each observation
- simple to update model $\rightarrow \boldsymbol{\theta}^{(k+1)}$
- repeat process, $k = k + 1$

Each iteration guaranteed not to decrease likelihood

- “Hard” assignment not always reasonable (overlapping data)
  - derive a more general iterative approach

Jensen's Inequality



- Jensen's inequality states:

$$f\left(\sum_{m=1}^{M} \lambda_m x_m\right) \geq \sum_{m=1}^{M} \lambda_m f(x_m)$$

$f()$ is any concave function and

$$\sum_{m=1}^{M} \lambda_m = 1, \quad \lambda_m \geq 0 \;\; m = 1, \ldots, M$$

- From the diagram above (left)

$$f(c) = f((1-\lambda)a + \lambda b) \geq (1-\lambda)f(a) + \lambda f(b)$$

- Interested in applying to GMMs - λ_m component prior $P(c_m)$

$$\log\left(\sum_{m=1}^{M} \lambda_m p(\boldsymbol{x}|c_m)\right) \geq \sum_{m=1}^{M} \lambda_m \log\left(p(\boldsymbol{x}|c_m)\right), \quad \sum_{m=1}^{M} \lambda_m = 1, \quad \lambda_m \geq 0$$

EM for Discrete Latent Variables

- Changing parameters from $\boldsymbol{\theta}^{(k)}$ to $\boldsymbol{\theta}^{(k+1)}$ - desire

$$\mathcal{L}(\boldsymbol{\theta}^{(k+1)}) - \mathcal{L}(\boldsymbol{\theta}^{(k)}) = \sum_{i=1}^{N} \log\left(\frac{p(\mathbf{x}_i;\boldsymbol{\theta}^{(k+1)})}{p(\mathbf{x}_i;\boldsymbol{\theta}^{(k)})}\right) \geq 0$$

- This can be written as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}^{(k+1)}) - \mathcal{L}(\boldsymbol{\theta}^{(k)}) &= \sum_{i=1}^{N} \log\left(\frac{1}{p(\mathbf{x}_i;\boldsymbol{\theta}^{(k)})}\sum_{m=1}^{M}\left(p(\mathbf{x}_i,\mathrm{c}_m;\boldsymbol{\theta}^{(k+1)})\right)\right)\\ &= \sum_{i=1}^{N} \log\left(\frac{1}{p(\mathbf{x}_i;\boldsymbol{\theta}^{(k)})}\sum_{m=1}^{M}\left(\frac{P(\mathrm{c}_m|\mathbf{x}_i;\boldsymbol{\theta}^{(k)})p(\mathbf{x}_i,\mathrm{c}_m;\boldsymbol{\theta}^{(k+1)})}{P(\mathrm{c}_m|\mathbf{x}_i;\boldsymbol{\theta}^{(k)})}\right)\right)\end{aligned}$$

- Jensen's inequality - use $P(\mathrm{c}_m|\mathbf{x}_i;\boldsymbol{\theta}^{(k)})$ as λ_m

$$\mathcal{L}(\boldsymbol{\theta}^{(k+1)}) - \mathcal{L}(\boldsymbol{\theta}^{(k)}) \geq \sum_{i=1}^{N}\sum_{m=1}^{M} P(\mathrm{c}_m|\mathbf{x}_i;\boldsymbol{\theta}^{(k)}) \log\left(\frac{p(\mathbf{x}_i,\mathrm{c}_m;\boldsymbol{\theta}^{(k+1)})}{p(\mathbf{x}_i;\boldsymbol{\theta}^{(k)})P(\mathrm{c}_m|\mathbf{x}_i;\boldsymbol{\theta}^{(k)})}\right)$$

Auxiliary Functions

- This inequality can be written as

$$\mathcal{L}(\boldsymbol{\theta}^{(k+1)}) - \mathcal{L}(\boldsymbol{\theta}^{(k)}) \geq \mathcal{Q}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k+1)}) - \mathcal{Q}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k)})$$

  - where

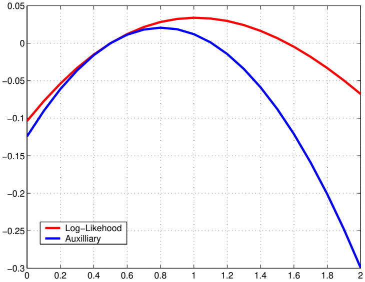
$$\mathcal{Q}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k+1)}) = \sum_{i=1}^{N} \sum_{m=1}^{M} P(\mathrm{c}_m | \boldsymbol{x}_i; \boldsymbol{\theta}^{(k)}) \log \left(p(\boldsymbol{x}_i, \mathrm{c}_m; \boldsymbol{\theta}^{(k+1)}) \right)$$

- This is known as the auxiliary function
  - optimising the log-likelihood requires

$$\mathcal{Q}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k+1)}) - \mathcal{Q}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k)}) \geq 0$$

  - increase is lower bound on the increase in the log likelihood.

EM Example



- Data generated from the following GMM:

$$x \sim 0.4 \times \mathcal{N}(1, 1) + 0.6 \times \mathcal{N}(-1, 1)$$

  - find first component mean: initial estimate is 0.5:

$$x^{(0)} \sim 0.4 \times \mathcal{N}(0.5, 1) + 0.6 \times \mathcal{N}(-1, 1)$$

  - log-likelihood and auxillary function differences to $\mathcal{L}(\boldsymbol{\theta}^{(0)})$ and $\mathcal{Q}(\boldsymbol{\theta}^{(0)}, \boldsymbol{\theta}^{(0)})$ plotted above

General form and Continuous Auxiliary Functions

- For GMMs the latent variable of interest is c_m, generalise
  - consider a set of latent variable $\boldsymbol{Z} = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$
  - associated with the set of observations $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
- The general form of auxiliary function becomes

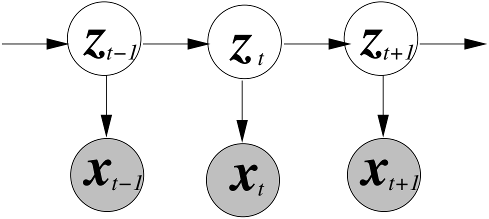
$$\mathcal{Q}(\boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^{(k+1)}) = \sum_{\forall \boldsymbol{Z}} P(\boldsymbol{Z}|\boldsymbol{X}; \boldsymbol{\theta}^{(k)}) \log \left(p(\boldsymbol{X}, \boldsymbol{Z}; \boldsymbol{\theta}^{(k+1)}) \right)$$

  - require posteriors over the latent variables $\boldsymbol{Z}$
- The continuous latent variable case version is

$$\mathcal{Q}(\boldsymbol{\theta}^{(k)}; \boldsymbol{\theta}^{(k+1)}) = \int p(\boldsymbol{Z}|\boldsymbol{X}; \boldsymbol{\theta}^{(k)}) \log \left(p(\boldsymbol{X}, \boldsymbol{Z}; \boldsymbol{\theta}^{(k+1)}) \right) d\boldsymbol{Z}$$

  - applicable to HMMs, FA models, Kalman Smoothers ...

Hidden Markov Models



- Continuous latent variable (similar to FA)
  - linear relationship between variables

$$\boldsymbol{z}_t = \boldsymbol{A}\boldsymbol{z}_{t-1} + \boldsymbol{\nu}_t; \quad \boldsymbol{x}_t = \boldsymbol{C}\boldsymbol{z}_t + \boldsymbol{\epsilon}_t$$

  - $\boldsymbol{\nu}_t$ are Gaussian distributed (and independent) $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_\nu)$
  - $\boldsymbol{\epsilon}_t$ are Gaussian distributed (and independent) $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_\epsilon)$

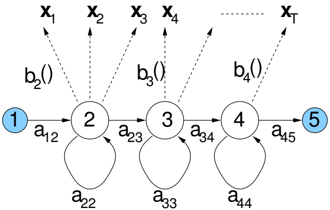
- This yields the following equations

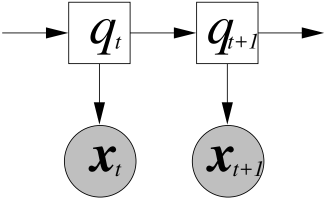
$$p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{A}\boldsymbol{z}_{t-1}, \boldsymbol{\Sigma}_\nu); \quad p(\boldsymbol{x}_t|\boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{x}_t; \boldsymbol{C}\boldsymbol{z}_t, \boldsymbol{\Sigma}_\epsilon)$$

- Everything is Gaussian and linear closed-form solutions
  - don't remember these - for illustration purposes

$$\begin{aligned}
p(\boldsymbol{x}_t|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}) \\
&= \int p(\boldsymbol{x}_t|\boldsymbol{z}_t) p(\boldsymbol{z}_t|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}) d\boldsymbol{z}_t \\
&= \int p(\boldsymbol{x}_t|\boldsymbol{z}_t) \int p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) p(\boldsymbol{z}_{t-1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}) d\boldsymbol{z}_{t-1} d\boldsymbol{z}_t \\
&= \int p(\boldsymbol{x}_t|\boldsymbol{z}_t) \int p(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \frac{p(\boldsymbol{x}_{t-1}|\boldsymbol{z}_{t-1}) p(\boldsymbol{z}_{t-1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-2})}{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-2})} d\boldsymbol{z}_{t-1} d\boldsymbol{z}_t
\end{aligned}$$

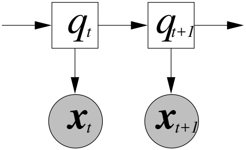
Hidden Markov Model





Hidden Markov Models

- Important sequence data is the hidden Markov model (HMM)
  - an example of a dynamic Bayesian network (DBN)
  - consider a sequence of observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$



- add discrete latent variables
  - q_t describes discrete state-space
  - conditional independence assumptions

$$P(q_t|q_0, \ldots, q_{t-1}) = P(q_t|q_{t-1})$$
$$p(\boldsymbol{x}_t|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{t-1}, q_0, \ldots, q_t) = p(\boldsymbol{x}_t|q_t)$$

- The likelihood of the data is

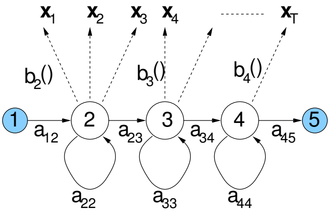
$$p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \sum_{\boldsymbol{q} \in \boldsymbol{Q}_T} P(\boldsymbol{q}) p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T|\boldsymbol{q}) = \sum_{\boldsymbol{q} \in \boldsymbol{Q}_T} P(q_0) \prod_{t=1}^{T} P(q_t|q_{t-1}) p(\boldsymbol{x}_t|q_t)$$

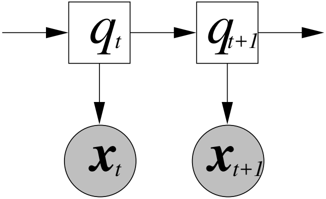
$\boldsymbol{Q}_T$ is all possible state sequences for T observations

HMM Parameters

- Two types of states often defined for HMMs (total N states)
  - emitting states: produce the observation sequence
  - non-emitting states: used to define valid state and end states
- The parameters are normally split into two
  - states are denoted s_i, assume s_1 and s_N are non-emitting
  - transition matrix $\boldsymbol{A}$:
    $a_{ij} = P(q_t = \mathrm{s}_j | q_{t-1} = \mathrm{s}_i)$ is the probability of transitioning from state s_i to state s_j
  - state output probability $\{b_2(\boldsymbol{x}_t), \ldots, b_{N-1}(\boldsymbol{x}_t)\}$:
    $b_j(\boldsymbol{x}_t) = p(\boldsymbol{x}_t | q_t = \mathrm{s}_j)$ is the output distribution for state s_j
- Need to estimate $\boldsymbol{\theta} = \{\boldsymbol{A}, b_2(\boldsymbol{x}_t), \ldots, b_{N-1}(\boldsymbol{x}_t)\}$
  - usually trained using Expectation-Maximisation (EM)

Hidden Markov Model





- To design a classifier need to determine:
  - transition matrix: discrete state-space and allowed transitions
  - state output distribution: form of distribution $p(\boldsymbol{x}_t|q_t)$
- Can be used as a generative classifier (HMM for exch class ω)

$$\hat{\omega} = \arg\max_{\omega} \left\{ P(\omega|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \right\} = \arg\max_{\omega} \left\{ P(\omega) p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T|\omega) \right\}$$

need to be able to compute $p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T|\omega)$ efficiently

Viterbi Approximation

- Viterbi Algorithm important technique (inc. for HMMs)
  - approximate likelihood as

$$p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \sum_{\boldsymbol{q} \in \boldsymbol{Q}_T} p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, \boldsymbol{q}) \approx p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, \hat{\boldsymbol{q}})$$

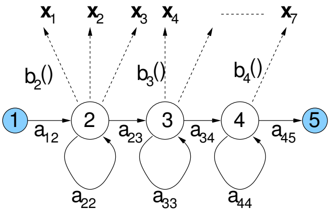
  where

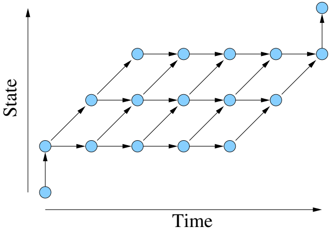
$$\hat{\boldsymbol{q}} = \{\hat{q}_0, \ldots, \hat{q}_{T+1}\} = \arg\max_{\boldsymbol{q} \in \boldsymbol{Q}_T} \{p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, \boldsymbol{q})\}$$

- This yields:
  - an approximate likelihood (lower bound) for the model
  - the best state-sequence through the discrete-state space

Viterbi Algorithm

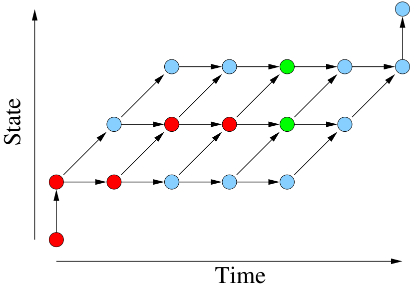
- Need to find the best state-sequence, $\hat{\boldsymbol{q}}$,
  - searching through all possible state-sequences impractical ...





- Consider generating the observation sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_7$
  - topology - 3 emitting states with strict left-to-right
  - representation of all possible state sequences on the right

Extending Partial Paths with Time

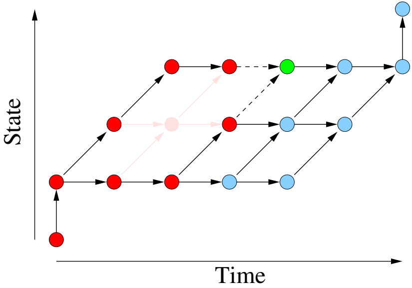


- Red partial path to time 4
- Green possible extensions

- Partial path state sequence $\{1, 2, 2, 3, 3\}$ ($\phi_3(4)$) extend:
  - stay in state s_3 and generate $\boldsymbol{x}_5$: $\log(a_{33} b_3(\boldsymbol{x}_5))$
  - move to state s_4 and generate $\boldsymbol{x}_5$: $\log(a_{34} b_4(\boldsymbol{x}_5))$
- Hence:

$$\phi_3(5) = \phi_3(4) + \log(a_{33} b_3(\boldsymbol{x}_5)), \quad \phi_4(5) = \phi_3(4) + \log(a_{34} b_4(\boldsymbol{x}_5))$$

Best Partial Path to a State/Time



- Red possible partial paths
- Green state of interest

- Consider best partial path to state s_4 at time 5 ($\phi_4(5)$)
  - move from state s_3 and generate $\boldsymbol{x}_5$: $\log(a_{34} b_4(\boldsymbol{x}_5))$
  - stay in state s_4 and generate $\boldsymbol{x}_5$: $\log(a_{44} b_4(\boldsymbol{x}_5))$
- Select "best":

$$\phi_4(5) = \max\left\{\phi_3(4) + \log(a_{34}), \phi_4(4) + \log(a_{44})\right\} + \log(b_4(\boldsymbol{x}_5))$$

Viterbi Algorithm for HMMs

- The Viterbi algorithm for HMMs can then be expressed as:
  - Initialisation: ($\texttt{LZERO}= \log(0)$)

$$\phi_1(0) = 0.0, \quad \phi_j(0) = \texttt{LZERO}, 1 < j < N, \quad \phi_1(t) = \texttt{LZERO}, 1 \leq t \leq T$$

  - Recursion:

    for $t = 1, \ldots, T$

        for $j = 2, \ldots, N-1$

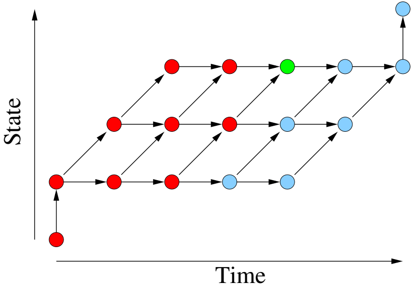
$$\phi_j(t) = \max_{1 \leq k < N} \{\phi_k(t-1) + \log(a_{kj})\} + \log(b_j(\boldsymbol{x}_t))$$

  - Termination:

$$\log(p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, \hat{\boldsymbol{q}})) = \max_{1<k<N} \{\phi_k(T) + \log(a_{kN})\}$$

- Can also store best previous state to yield best sequence $\hat{\boldsymbol{q}}$.

All Paths to a State/Time



- Red possible partial paths
- Green state of interest

$$\text{LAdd}(a, b) = \log\left(\exp(a) + \exp(b)\right)$$
$$\exp(\text{LAdd}(a, b)) = \exp(a) + \exp(b)$$

- Total path to state $\mathtt{s}_i$ at time t is $\alpha_i(t)$
  - total path to state $\mathtt{s}_4$ at time 5 given by (see Viterbi)

$$\alpha_4(5) = \text{LAdd}\left(\alpha_3(4) + \log(a_{34}), \alpha_4(4) + \log(a_{44})\right) + \log(b_4(\boldsymbol{x}_5))$$

Forward-Backward Algorithm

- $\alpha_j(t)$ is related to the forward-probability used to train HMMs
  - recursion for this form of model can be expressed as

$$\begin{aligned}\alpha_j(t) &= \log\left(p(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_t, q_t = \mathrm{s}_j)\right)\\ &= \log\left(\sum_{k=1}^{N}\exp\left(\alpha_k(t-1)+\log(a_{kj})\right)\right)+\log(b_j(\boldsymbol{x}_t))\end{aligned}$$

- There's also a term related to the backward-probability
  - consider observation at time t given state s_j, $\beta_j(t)$

$$\begin{aligned}\beta_j(t) &= \log\left(p(\boldsymbol{x}_{t+1},\ldots,\boldsymbol{x}_T|q_t = \mathrm{s}_j)\right)\\ &= \log\left(\sum_{k=1}^{N}\exp\left(\beta_k(t+1)+\log(a_{jk})+\log(b_k(\boldsymbol{x}_{t+1}))\right)\right)\end{aligned}$$

- The posterior required for EM can be expressed as

$$P(q_t = \mathrm{s}_j|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_T) = \exp(\alpha_j(t)+\beta_j(t))/Z, \quad Z = \sum_{i=1}^{N}\exp\left(\alpha_i(t)+\beta_i(t)\right)$$

Conditional Random Fields

Discriminative Sequence Models

- HMMs compute the joint distribution of words and tags
  - the posterior probability of the tag sequence $\boldsymbol{t}_{1:L}$

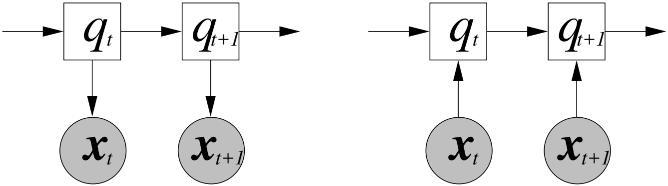
$$P(\boldsymbol{t}_{1:L}|\boldsymbol{w}_{1:L}) = \frac{P(\boldsymbol{t}_{1:L}, \boldsymbol{w}_{1:L})}{P(\boldsymbol{w}_{1:L})} = \frac{P(\boldsymbol{t}_{1:L}, \boldsymbol{w}_{1:L})}{\sum_{\tilde{\boldsymbol{t}}_{1:L}} P(\tilde{\boldsymbol{t}}_{1:L}, \boldsymbol{w}_{1:L})}$$

    summation is over all possible L-length tag sequences
  - a generative sequence model

- Discrimative sequence models directly model $P(\boldsymbol{t}_{1:L}|\boldsymbol{w}_{1:L})$
  - required to be a valid probability mass function (PMF)
  - but only sequence probability needs to be a valid PMF
  - not the individual tag predictions, or word probabilities

- Here tags will be states ($\boldsymbol{q}_{1:T}$) and words features ($\boldsymbol{x}_{1:T}$)
  - similar to discussion of HMMs

Simple Discriminative Sequence Models



Generative Model　　Discriminative Model

- Generative model (left) and discriminative model (right)
    - right BN a maximum entropy Markov model

$$P(q_0, \ldots, q_T | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \prod_{t=1}^{T} P(q_t | q_{t-1}, \boldsymbol{x}_t)$$

state posterior probability (Z_t normalisation term time t):

$$P(q_t | q_{t-1}, \boldsymbol{x}_t) = \frac{1}{Z_t} \exp\left(\sum_{i=1}^{D} \lambda_i f_i(q_t, q_{t-1}, \boldsymbol{x}_t) \right)$$

Sequence Maximum Entropy Models

- State posteriors modelled in Maximum Entropy Markov model
  - can extend to the complete sequence

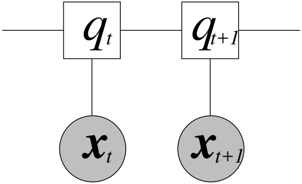
$$P(q_0, \ldots, q_T | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \frac{1}{Z} \exp\left(\sum_{i=1}^{D} \lambda_i f_i(q_0, \ldots, q_T, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \right)$$

- Problem is that there are a vast number of possible features:

  What features to extract from state/observation sequence?

  - need to be able to handle variations in length of the sequence
  - keep the number of model parameters $\boldsymbol{\lambda}$ reasonable

(Simple) Linear Chain Conditional Random Fields



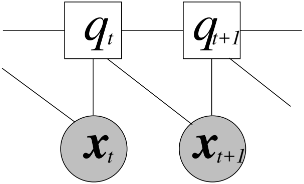
- Extract features based on undirected graph
  - conditional independence assumptions similar to HMM (though undirected)
- Posterior model becomes

$$P(q_0, \ldots, q_T | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \frac{1}{Z} \exp\left(\sum_{t=1}^{T} \left(\sum_{i=1}^{D_{\mathrm{t}}} \lambda_i^{\mathrm{t}} f_i(q_t, q_{t-1}) + \sum_{i=1}^{D_{\mathrm{a}}} \lambda_i^{\mathrm{a}} f_i(q_t, \boldsymbol{x}_t) \right) \right)$$

  - D_{t} number of transition style features with parameters $\boldsymbol{\lambda}^{\mathrm{t}}$
  - D_{a} number of acoustic style features with parameters $\boldsymbol{\lambda}^{\mathrm{a}}$
- Directly related to (unnormalised) HMM parameters

$$\sum_{i=1}^{D_{\mathrm{t}}} \lambda_i^{\mathrm{t}} f_i(\mathrm{s}_i, \mathrm{s}_j) \;\; \log(a_{ij}) \quad \text{and} \quad \sum_{i=1}^{D_{\mathrm{a}}} \lambda_i^{\mathrm{a}} f_i(\mathrm{s}_i, \boldsymbol{x}_t) \;\; \log(b_i(\boldsymbol{x}_t))$$

Linear Chain Conditional Random Fields



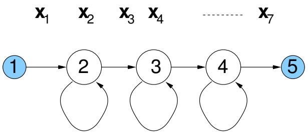
- Extract features based on undirected graph
  - conditional independence assumptions extended to previous state
- Posterior model becomes

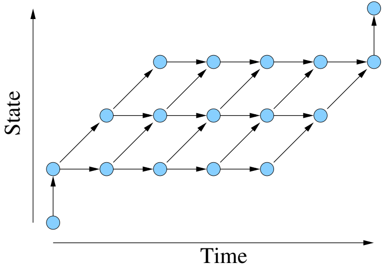
$$P(q_0, \ldots, q_T | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \frac{1}{Z} \exp\left(\sum_{t=1}^{T}\left(\sum_{i=1}^{D} \lambda_i f_i(q_t, q_{t-1}, \boldsymbol{x}_t)\right)\right)$$

- More interesting than HMM-like features
  - features the similar to (general) MaxEnt Markov model
  - BUT normalised globally not locally

- Need to be able to compute the normalisation term efficiently
  - initially consider the simple linear chain case

$$Z = \sum_{\boldsymbol{q} \in \boldsymbol{Q}_T} \exp\left(\sum_{t=1}^{T}\left(\sum_{i=1}^{D_{\text{t}}} \lambda_i^{\text{t}} f_i(q_t, q_{t-1}) + \sum_{i=1}^{D_{\text{a}}} \lambda_i^{\text{a}} f_i(q_t, \boldsymbol{x}_t)\right)\right)$$





- Left-to-right topology and observation sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_7$
  - use equivalent of forward-backward algorithm

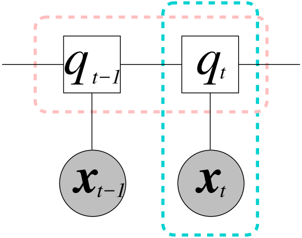
General Sequence CRFs (reference)

- General CRF use undirected graphical model to define features
  - need to be able to handle sequence data - dynamic CRF
  - undirected graph repeated each time instance - set of cliques $\boldsymbol{C}$
- The posterior probability for this form of model is

$$P(q_0, \ldots, q_T | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) = \frac{1}{Z} \exp\left(\sum_{t=1}^{T} \sum_{\mathcal{C} \in \boldsymbol{C}} \boldsymbol{\lambda}_{\mathcal{C}}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{q}_{\mathcal{C}t}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, t) \right)$$

  - $\boldsymbol{\lambda}_{\mathcal{C}}^{\mathsf{T}}$ time-independent parameters associated with clique $\mathcal{C}$
  - $\boldsymbol{f}(\boldsymbol{q}_{\mathcal{C}t}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, t)$ time-dependent features extracted from clique $\mathcal{C}$ with time-dependent label sequence $\boldsymbol{q}_{\mathcal{C}t}$

Example of a Sequence CRF (reference)



- Cliques associated with linear CRF

  $\boldsymbol{C} = \{\mathcal{C}_1, \mathcal{C}_2\}$

  1. transitions: $\mathcal{C}_1 = \{q_t, q_{t-1}\}$
  2. acoustics: $\mathcal{C}_2 = \{q_t, \boldsymbol{x}_t\}$

- Posterior model for the simple linear chain CRF

$$
\begin{aligned}
P(q_0, \ldots, q_T | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) &= \frac{1}{Z} \exp\left(\sum_{t=1}^{T} \sum_{\mathcal{C} \in \boldsymbol{C}} \boldsymbol{\lambda}_{\mathcal{C}}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{q}_{\mathcal{C}t}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, t) \right) \\
&= \frac{1}{Z} \exp\left(\sum_{t=1}^{T} \left(\boldsymbol{\lambda}^{\mathrm{t}\mathsf{T}} \boldsymbol{f}(q_t, q_{t-1}) + \boldsymbol{\lambda}^{\mathrm{a}\mathsf{T}} \boldsymbol{f}(q_t, \boldsymbol{x}_t) \right) \right)
\end{aligned}
$$

Training CRFs

- Training for CRFs is normally fully observed

training observation sequence	$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$
training label sequence	$y_1, \ldots, y_T$

  - where $y_\tau \in \{\omega_1, \ldots, \omega_K\}$
- No need to use EM (or related approaches)
- Need to find the model parameters $\boldsymbol{\lambda}$ so that

$$
\begin{aligned}
\hat{\boldsymbol{\lambda}} &= \arg\max_{\boldsymbol{\lambda}} \left\{ P(y_1, \ldots, y_T | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T; \boldsymbol{\lambda}) \right\} \\
&= \arg\max_{\boldsymbol{\lambda}} \left\{ \frac{1}{Z} \exp \left(\sum_{i=1}^{D} \lambda_i f_i(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T, y_1, \ldots, y_T) \right) \right\}
\end{aligned}
$$