COMS30035, Machine learning: Sequential Data (HMMs)

James Cussens

School of Computer Science
University of Bristol

27th September 2024

Acknowledgement

- These slides are adapted from ones originally created by Edwin Simpson.

Agenda

- Markov Models
- Hidden Markov Models
- EM for HMMs
- Linear Dynamical Systems

i.i.d. Data

- Up to now, we have considered the data points in our datasets to be *independent and identically distributed* (i.i.d.)
- Independent: the value of one data point does not affect the others,$p(\mathbf{x}_1, \mathbf{x}_2) = p(\mathbf{x}_1)p(\mathbf{x}_2)$
- Identically distributed: all data points have the same distribution, $p(\mathbf{x}_i) = p(\mathbf{x}_j), \forall i, \forall j$

i.i.d. Data

- Up to now, we have considered the data points in our datasets to be *independent and identically distributed* (i.i.d.)
- Independent: the value of one data point does not affect the others,$p(\boldsymbol{x}_1, \boldsymbol{x}_2) = p(\boldsymbol{x}_1)p(\boldsymbol{x}_2)$
- Identically distributed: all data points have the same distribution, $p(\boldsymbol{x}_i) = p(\boldsymbol{x}_j), \forall i, \forall j$

i.i.d. Data

- Up to now, we have considered the data points in our datasets to be *independent and identically distributed* (i.i.d.)
- Independent: the value of one data point does not affect the others,$p(\boldsymbol{x}_1, \boldsymbol{x}_2) = p(\boldsymbol{x}_1)p(\boldsymbol{x}_2)$
- Identically distributed: all data points have the same distribution, $p(\boldsymbol{x}_i) = p(\boldsymbol{x}_j), \forall i, \forall j$

i.i.d. Data

- So, once you have trained a classifier or regressor, you can predict the output for each data point independently.
- Can you think of situations where the i.i.d. assumption does not apply?

Sequential Data

- The i.i.d. assumption ignores any ordering of the data points.
- Data points often occur in a sequence, such as words in a sentence, frames in a video, sensor observations over time, stock prices...
- This can be generalised to more than one dimension: object in different parts of an image, geographical data on a map... (not covered in this lecture).
- Can you think of some classification or regression tasks for these types of data?

Sequential Data

- The i.i.d. assumption ignores any ordering of the data points.
- Data points often occur in a sequence, such as words in a sentence, frames in a video, sensor observations over time, stock prices...
- This can be generalised to more than one dimension: object in different parts of an image, geographical data on a map... (not covered in this lecture).
- Can you think of some classification or regression tasks for these types of data?

Modelling Sequential Data

- How have we modelled relationships between data points so far? – Through their input features.
- Can we model sequential relationships by simply making *time* or *position in the sequence* into another feature?
- No – The timestamp or positional index is not in itself an informative feature
- But the data observed at other points in the sequence tells us about our current data point

Modelling Sequential Data

- How have we modelled relationships between data points so far? – Through their input features.
- Can we model sequential relationships by simply making *time* or *position in the sequence* into another feature?
- No – The timestamp or positional index is not in itself an informative feature
- But the data observed at other points in the sequence tells us about our current data point

Modelling Sequential Data

- How have we modelled relationships between data points so far? – Through their input features.
- Can we model sequential relationships by simply making *time* or *position in the sequence* into another feature?
- No – The timestamp or positional index is not in itself an informative feature
- But the data observed at other points in the sequence tells us about our current data point

Modelling Sequential Data

- How have we modelled relationships between data points so far? – Through their input features.
- Can we model sequential relationships by simply making *time* or *position in the sequence* into another feature?
- No – The timestamp or positional index is not in itself an informative feature
- But the data observed at other points in the sequence tells us about our current data point

Modelling Sequential Data

- Look at the following two texts from Bishop's book, both with a missing word:
    - "later termed Bayes' ____ by Poincaré"
    - "The evaluation of this conditional can be seen as an example of Bayes' ____"
- Can you guess the missing words? How did you guess them?
- You can guess that the missing word in both cases is "theorem" or maybe "rule", because of the word "Bayes"' right before it.
- The first missing word is at position 3, the second is at position 13, but these position indexes don't help to identify the missing word.

Modelling Sequential Data

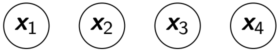
- Look at the following two texts from Bishop's book, both with a missing word:
  - "later termed Bayes' ____ by Poincaré"
  - "The evaluation of this conditional can be seen as an example of Bayes' ____"
- Can you guess the missing words? How did you guess them?
- You can guess that the missing word in both cases is "theorem" or maybe "rule", because of the word "Bayes"' right before it.
- The first missing word is at position 3, the second is at position 13, but these position indexes don't help to identify the missing word.

Modelling Sequential Data

- Look at the following two texts from Bishop's book, both with a missing word:
  - "later termed Bayes' ____ by Poincarré"
  - "The evaluation of this conditional can be seen as an example of Bayes' ____"
- Can you guess the missing words? How did you guess them?
- You can guess that the missing word in both cases is "theorem" or maybe "rule", because of the word "Bayes"' right before it.
- The first missing word is at position 3, the second is at position 13, but these position indexes don't help to identify the missing word.

How Can We Model the Dependencies?

i.i.d., $p(\boldsymbol{x}_n | \boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}) = p(\boldsymbol{x}_n)$



How Can We Model the Dependencies?

i.i.d., $p(\boldsymbol{x}_n | \boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}) = p(\boldsymbol{x}_n)$

$$\begin{tikzcd}
\boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 & \boldsymbol{x}_4
\end{tikzcd}$$

Modelling all connections, $p(\boldsymbol{x}_n | \boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1})$ – *intractable*

$$\begin{tikzcd}
\boldsymbol{x}_1 \arrow[r] \arrow[rr, bend right] \arrow[rrr, bend right] & \boldsymbol{x}_2 \arrow[r] \arrow[rr, bend left] & \boldsymbol{x}_3 \arrow[r] & \boldsymbol{x}_4
\end{tikzcd}$$

How Can We Model the Dependencies?

i.i.d., $p(\boldsymbol{x}_n|\boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}) = p(\boldsymbol{x}_n)$

$$\begin{tikzcd}
\boldsymbol{x}_1 & \boldsymbol{x}_2 & \boldsymbol{x}_3 & \boldsymbol{x}_4
\end{tikzcd}$$

Modelling all connections, $p(\boldsymbol{x}_n|\boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1})$ − *intractable*

$$\begin{tikzcd}
\boldsymbol{x}_1 \arrow[r] \arrow[rr, bend right] \arrow[rrr, bend right] & \boldsymbol{x}_2 \arrow[r] \arrow[rr, bend left] & \boldsymbol{x}_3 \arrow[r] & \boldsymbol{x}_4
\end{tikzcd}$$

1st order Markov chain, $p(\boldsymbol{x}_n|\boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}) = p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$

$$\begin{tikzcd}
\boldsymbol{x}_1 \arrow[r] & \boldsymbol{x}_2 \arrow[r] & \boldsymbol{x}_3 \arrow[r] & \boldsymbol{x}_4
\end{tikzcd}$$

$$p(\boldsymbol{x}_1, ..., \boldsymbol{x}_N) = p(\boldsymbol{x}_1) \prod_{n=2}^{N} p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$$

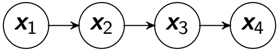
Homogeneous Markov Chains

- *Stationary* distribution: the probability distribution remains the same over time.
- This leads to a *homogeneous* Markov chain.
- E.g., the parameters of the distribution remain the same while the data evolves.
- Contrast with non-stationary distributions that change over time.

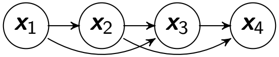
Higher-Order Markov Models

- Sometimes it is necessary to consider earlier observations using a higher-order chain.
- However, the number of parameters increases with the order of the Markov chain, meaning higher-order models are often impractical.

1st order Markov chain, $p(\boldsymbol{x}_n|\boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}) = p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1})$



2nd order Markov chain, $p(\boldsymbol{x}_n|\boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}) = p(\boldsymbol{x}_n|\boldsymbol{x}_{n-1}, \boldsymbol{x}_{n-2})$



State Space Models

- What if we don't directly observe the states we want to model?
- E.g., we want to predict the state of the weather (raining, sunny, cloudy, rainfall)
- We observe noisy measurements of temperature, wind, rainfall over a period of time

State Space Models

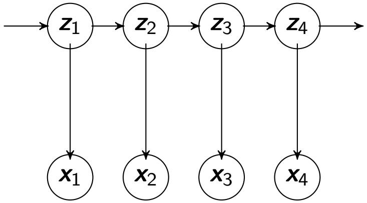
- What if we don't directly observe the states we want to model?
- E.g., we want to identify different actions in a video of a game of tennis, such as backhand volley
- We observe the frames in a video, each one of which is a tensor of pixel values
- We encounter the same problem as we do in i.i.d. classification and regression: the sequential variable we wish to predict is not directly observed.

State Space Models

- What if we don't directly observe the states we want to model?
- E.g., we want to identify different actions in a video of a game of tennis, such as backhand volley
- We observe the frames in a video, each one of which is a tensor of pixel values
- We encounter the same problem as we do in i.i.d. classification and regression: the sequential variable we wish to predict is not directly observed.

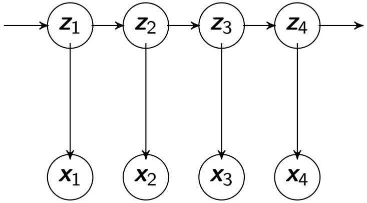
State Space Models

- Introduce latent variables, $\boldsymbol{z}_n$ that form a Markov chain;
- Each observation $\boldsymbol{x}_n$ depends on $\boldsymbol{z}_n$;
- This means we do not need to model the dependencies between observations $\boldsymbol{x}_n$ directly;
- Latent variables model the state of the system, while observations may be of different types, contain noise...



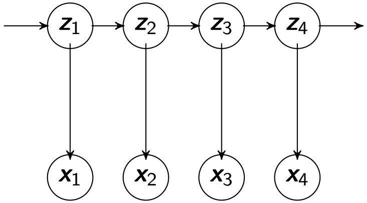
State Space Models

- Does this look similar to any classifiers you have come across before?



State Space Models

- *Hidden Markov Models (HMMs):* Discrete state z, observations may be continuous or discrete according to any distribution. $\rightarrow$ next part of this lecture
- *Linear Dynamical Systems (LDS):* Continuous state z, observations are continuous, both have Gaussian distributions $\rightarrow$ next lecture
- We will consider both supervised and unsupervised settings.

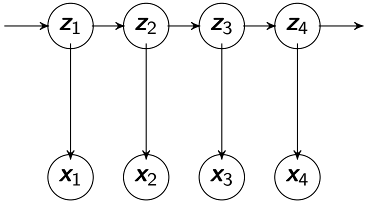


Agenda

- Markov Models
- Hidden Markov Models
- EM for HMMs
- Linear Dynamical Systems

Hidden Markov Models (HMMs)

- A state space model
- $\boldsymbol{z}_n$ are latent (unobserved) discrete state variables.
- $\boldsymbol{x}_n$ are observations, which may be discrete or continuous values depending on the application.



Uses of HMMs: Sequence Labelling for Text

- *Sequence labelling*, i.e., classifying data points in a sequence.
- E.g., classifying words in a text document into grammatical categories such as “noun”, “verb”, “adjective”, etc.
- This is called part-of-speech (POS) tagging and is used by natural language understanding systems, e.g., to extract facts and events from text data.

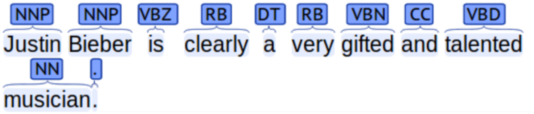


Image from *Automatic Annotation Suggestions and Custom Annotation Layers in WebAnno*, Yimam et al., 2014, ACL System Demonstrations.

Uses of HMMs: Human Action Recognition

- Observations: sequence of images (video frames) of a person playing tennis.
- Latent states: the actions being taken:
  - Backhand volley;
  - Forehand volley;
  - Forehand stroke;
  - Smash;
  - Serve.
- Why use an HMM? Actions typically follow a temporal sequence.

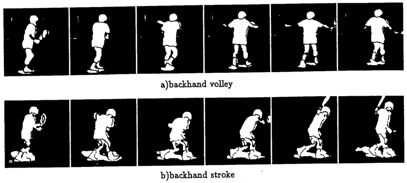


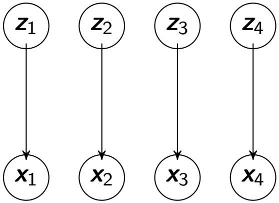
Image from Yamato, J., Ohya, J., Ishii, K. (1992). *Recognizing human action in time-sequential images using hidden Markov mode*. In CVPR (Vol. 92, pp. 379-385).

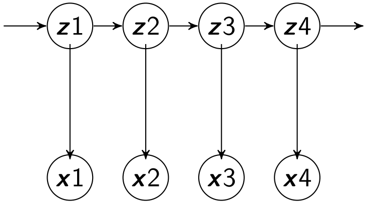
Uses of HMMs: In General

- HMMs can be used with different goals in mind:
  - Inferring the latent states (sequence labelling);
  - Predicting the next latent state;
  - Predicting the next observation;
- They can also be used with different levels of supervision:
  - Supervised: the latent states are given in the training set.
  - Unsupervised: no labels for the latent states, so the model seeks an assignment that best explains the observations given the model.
  - Semi-supervised: some labels are given, but the model is learned over both labelled and unlabelled data. Avoid overfitting to a very small labelled dataset while identifying latent states that follow the desired labelling scheme.

HMM is an Extension to Mixture Models

- Recall the latent variables, $\boldsymbol{z}_n$, in a mixture model, which identify the component responsible for an observation.
- These are also discrete variables, like latent states $\boldsymbol{z}_n$ in an HMM.
- In a mixture model, latent variables are i.i.d. rather than Markovian.





Anatomy of the HMM

- The probabilistic model of the HMM is made up of two main parts:
- The *transition* distribution, which can be represented as a *transition matrix* and models the dependencies between the latent states;
- The *emission* distributions, which model the observations given each latent state value.

Transition Matrix

- The probability of $\boldsymbol{z}_n$ depends on the previous state: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$.
- Given K labels (state values), we can write all the values of $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1} = l)$ in a *transition matrix*, $\boldsymbol{A}$.
  - Rows correspond to values of the previous state, $\boldsymbol{z}_{n-1}$.
  - Columns are values of the current state, $\boldsymbol{z}_n$.

$p(\boldsymbol{z}_n\|\boldsymbol{z}_{n-1}, \boldsymbol{A})$		$\boldsymbol{z}_n$		
		1	2	3
$\boldsymbol{z}_{n-1}$	1	0.5	0.1	0.4
	2	0.3	0.1	0.6
	3	0.01	0.19	0.8

- A vector of probabilities, $\boldsymbol{\pi}$ is used for $\boldsymbol{z}_1$, since it has no predecessor.
- What would the transition matrix for a mixture model look like?

Transition Matrix

- The probability of $\boldsymbol{z}_n$ depends on the previous state: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$.
- Given K labels (state values), we can write all the values of $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1} = l)$ in a *transition matrix*, $\boldsymbol{A}$.
  - Rows correspond to values of the previous state, $\boldsymbol{z}_{n-1}$.
  - Columns are values of the current state, $\boldsymbol{z}_n$.

$p(\boldsymbol{z}_n\|\boldsymbol{z}_{n-1}, \boldsymbol{A})$		$\boldsymbol{z}_n$		
		1	2	3
	1	0.5	0.1	0.4
$\boldsymbol{z}_{n-1}$	2	0.3	0.1	0.6
	3	0.01	0.19	0.8

- A vector of probabilities, π is used for z_1, since it has no predecessor.
- What would the transition matrix for a mixture model look like?

Transition Matrix

- The probability of $\boldsymbol{z}_n$ depends on the previous state: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$.
- Given K labels (state values), we can write all the values of $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1} = l)$ in a *transition matrix*, $\boldsymbol{A}$.
  - Rows correspond to values of the previous state, $\boldsymbol{z}_{n-1}$.
  - Columns are values of the current state, $\boldsymbol{z}_n$.

$p(\boldsymbol{z}_n\|\boldsymbol{z}_{n-1}, \boldsymbol{A})$		$\boldsymbol{z}_n$		
		1	2	3
	1	0.5	0.1	0.4
$\boldsymbol{z}_{n-1}$	2	0.3	0.1	0.6
	3	0.01	0.19	0.8

- A vector of probabilities, $\boldsymbol{\pi}$ is used for $\boldsymbol{z}_1$, since it has no predecessor.
- What would the transition matrix for a mixture model look like?

Transition Matrix

- The probability of $\boldsymbol{z}_n$ depends on the previous state: $p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1})$.
- Given K labels (state values), we can write all the values of $p(\boldsymbol{z}_n = k|\boldsymbol{z}_{n-1} = l)$ in a *transition matrix*, $\boldsymbol{A}$.
  - Rows correspond to values of the previous state, $\boldsymbol{z}_{n-1}$.
  - Columns are values of the current state, $\boldsymbol{z}_n$.

$p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}, \boldsymbol{A})$		$\boldsymbol{z}_n$		
		1	2	3
	1	0.5	0.1	0.4
$\boldsymbol{z}_{n-1}$	2	0.3	0.1	0.6
	3	0.01	0.19	0.8

- A vector of probabilities, $\boldsymbol{\pi}$ is used for $\boldsymbol{z}_1$, since it has no predecessor.
- What would the transition matrix for a mixture model look like?

Emission Distributions

- Distribution over the observed variables, $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \phi)$, where ϕ are parameters of the distributions, for example:
  - Real-valued observations may use Gaussian emissions;
  - If there are multiple observations, we may use a multivariate Gaussian;
  - Discrete observations may use a categorical distribution.
- For each observation there are K values of $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \phi)$, one for each possible value of the unobserved $\boldsymbol{z}_n$.

The Complete HMM Model

- The complete HMM can be defined by the joint distribution over observations and latent states:

$$p(\boldsymbol{X}, \boldsymbol{Z} | \boldsymbol{A}, \boldsymbol{\pi}, \phi) = p(\boldsymbol{z}_1 | \boldsymbol{\pi}) \prod_{n=2}^{N} p(\boldsymbol{z}_n | \boldsymbol{z}_{n-1}, \boldsymbol{A}) \prod_{n=1}^{N} p(\boldsymbol{x}_n | \boldsymbol{z}_n, \phi) \quad (1)$$

- $\boldsymbol{A}$, $\boldsymbol{\pi}$ and ϕ are parameters that must be learned or marginalised.
- Generative model: think of generating each of the state variables $\boldsymbol{z}_n$ in turn, then generating the observation $\boldsymbol{x}_n$ for each generated state.
- It's ancestral sampling (see Bayesian network lecture), once again.

Agenda

- Markov Models
- Hidden Markov Models
- EM for HMMs
- Linear Dynamical Systems

Hidden Markov Models (HMMs)

- We want to use maximum likelihood to estimate the HMM parameters:
  1. $\boldsymbol{A}$ - transition matrix
  2. $\boldsymbol{\pi}$ - initial state probabilities
  3. ϕ - parameters of the emission distributions
- We examine the *unsupervised* case where the sequence of states $\boldsymbol{Z}$ is not observed.
- $\ln p(\boldsymbol{X}|\boldsymbol{A}, \pi, \phi) =$
  $\ln \sum_{\boldsymbol{Z}} \left\{ p(\boldsymbol{z}_1|\boldsymbol{\pi}) \prod_{n=2}^{N} p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}, \boldsymbol{A}) \prod_{n=1}^{N} p(\boldsymbol{x}_n|\phi, \boldsymbol{z}_n) \right\}$

Likelihood for an HMM

- As with GMMs, there is no closed-form solution to the MLE, so we turn to EM
- Unlike GMM, the likelihood doesn't factorise over the data points:
  1. $\ln p(\boldsymbol{X}|\boldsymbol{A}, \boldsymbol{\pi}, \phi) =$
     $\ln \sum_{\boldsymbol{Z}} \left\{ p(\boldsymbol{z}_1|\boldsymbol{\pi}) \prod_{n=2}^{N} p(\boldsymbol{z}_n|\boldsymbol{z}_{n-1}, \boldsymbol{A}) \prod_{n=1}^{N} p(\boldsymbol{x}_n|\phi, \boldsymbol{z}_n) \right\}$
  2. The distribution of $\boldsymbol{z}_n$ depends on $\boldsymbol{z}_{n-1}$, which also depends on $\boldsymbol{z}_{n-2}$...
  3. Can't just sum over the values of z_n independently for each data point.
  4. So we have to sum over all K^N possible sequences $\boldsymbol{Z}$!

Expectation Maximisation (EM)

- Goal: maximise the expected log likelihood
- First, we define $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{old}) = \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}^{old}) \ln p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})$.

1. Initialise the parameters with a random guess: $\boldsymbol{\theta}^{old} = \{\boldsymbol{A}, \boldsymbol{\pi}, \phi\}$.
2. **E-step**: use $\boldsymbol{\theta}^{old}$ to compute expectations over $\boldsymbol{Z}$ required to compute $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{old})$.
3. **M-step**: choose the values of $\boldsymbol{\theta} = \{\boldsymbol{A}, \boldsymbol{\pi}, \phi\}$ that maximise $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{old})$.
4. Set $\boldsymbol{\theta}^{old} = \boldsymbol{\theta}$.
5. Repeat steps 2-4 until convergence.

E step

- We need to compute expectations of the latent states and pairs of latent states.
- (Note that a probability is just a special type of expectation: one for a binary random variable.)
- Responsibilities: $\gamma(z_{nk}) = p\left(z_n = k | \boldsymbol{X}, \boldsymbol{\theta}^{(old)}\right)$
- State pairs: $\xi(z_{n-1,j}, z_{nk}) = p\left(z_{n-1} = j, z_n = k | \boldsymbol{X}, \boldsymbol{\theta}^{(old)}\right)$
- To compute these efficiently, we need the *forward-backward* algorithm (coming up in a few slides...)

- “In the E step, we . . . find the posterior distribution of the latent variables $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}^{old})$”. [Bis06, p.616]
- But note that we don’t compute and store the entire distribution $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}^{old})$.
- Only the expectations (=probabilities) of the things we need for the subsequent M step.

M step

- $\pi_k = \gamma(z_{1k})$
- $A_{jk} = \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk}) / \sum_{n=2}^{N} \gamma(z_{n-1,j})$
- ϕ_k: parameters of posterior emission distributions, with observations weighted by responsibilities, $\gamma(z_{nk})$
  - If we have Gaussian emissions, the equations are the same as for GMM.
  - Discrete observations with value i:

$$\phi_{ki} = p(x_n = i | z_n = k) = \frac{\sum_{n=1}^{N} \gamma(z_{nk})[x_n = i]}{\sum_{n=1}^{N} \gamma(z_{nk})} \tag{2}$$

Forward-backward Algorithm

- A specific example of the *sum-product algorithm* used in the E-step
- Forward pass computes for each time-step n and state value k:

$$\begin{aligned}\alpha(z_{nk}) &= p(\boldsymbol{x}_1, ..., \boldsymbol{x}_n, z_n = k | \boldsymbol{\pi}, \boldsymbol{A}, \phi) \\ &= p(\boldsymbol{x}_n | z_n = k, \phi_k) \sum_{l=1}^{K} A_{lk} \alpha(z_{n-1,l}) \end{aligned} \tag{3}$$

- Backward pass computes:

$$\begin{aligned}\beta(z_{nk}) &= p(\boldsymbol{x}_{n+1}, ..., \boldsymbol{x}_N | z_n = k, \boldsymbol{A}, \phi) \\ &= \sum_{l=1}^{K} A_{kl} p(\boldsymbol{x}_{n+1} | z_{n+1} = l, \phi_l) \beta(z_{n+1,l}) \end{aligned} \tag{4}$$

Forward-backward Algorithm

- Use the computed α and β terms to compute our expectations over $\boldsymbol{z}$:

$$\begin{aligned}\tilde{\xi}(z_{n-1,l}, z_{nk}) =& p(\boldsymbol{x}_1, ..., \boldsymbol{x}_{n-1}, z_{n-1} = l | \boldsymbol{A}, \boldsymbol{\pi}, \phi) && \text{before} \\ & p(z_n = k | z_{n-1} = l, \boldsymbol{A}) p(\boldsymbol{x}_n | z_n = k, \phi) && \text{current} \\ & p(\boldsymbol{x}_{n+1}, ..., x_N | z_n = k, \boldsymbol{A}, \phi) && \text{after} \\ =& \alpha(z_{n-1,l}) A_{lk} p(\boldsymbol{x}_n | z_n = k, \phi) \beta(z_{nk}) && (5)\end{aligned}$$

- $\xi(z_{n-1,l}, z_{nk}) = \tilde{\xi}(z_{n-1,l}, z_{nk}) / \sum_{l=1}^{K} \sum_{k=1}^{K} \tilde{\xi}(z_{n-1,l}, z_{nk})$
- $\gamma(z_{nk}) = \sum_{l=1}^{K} \xi(z_{n-1,l}, z_{nk})$

Putting It All Together...

1. Initialise the parameters with a random guess: $\boldsymbol{\theta}^{old} = \{\boldsymbol{A}, \boldsymbol{\pi}, \phi\}$.
2. **E-step** using $\boldsymbol{\theta}^{old}$:
    2.1 Run forward pass to compute $\alpha(z_{nk})$
    2.2 Run backward pass to compute $\beta(z_{nk})$
    2.3 Use $\alpha(z_{n-1,l})$ and $\beta(z_{nk})$ to compute $\xi(z_{n-1,l}, z_{nk})$ and $\gamma(z_{nk})$.
3. **M-step** using $\xi(z_{n-1,l}, z_{nk})$ and $\gamma(z_{nk})$, update $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{A}, \phi\}$.
4. Set $\boldsymbol{\theta}^{old} = \boldsymbol{\theta}$.
5. Repeat steps 2-4 until convergence.

By summing inside each forward and backward computations, we now have an algorithm that is linear ($\mathcal{O}(N)$) rather than exponential ($\mathcal{O}(K^N)$) in the sequence length 😎.

Viterbi Algorithm

- Given our estimated model parameters $\boldsymbol{\theta} = \{\boldsymbol{\pi}, \boldsymbol{A}, \phi\}$, how can we predict a sequence of hidden states $\boldsymbol{Z}$?
- Most probable labels (given by the values of $\gamma(z_{nk})$) are not the same as the most probable *sequence*!
- We apply a *max-sum* algorithm called *Viterbi* to “decode” the sequence with $\mathcal{O}(N)$ computational cost.

Viterbi Algorithm

- Forward pass: compute the probability of the most likely sequence that leads to each possible state at time n.
- Backward pass: starting with the most likely final state and recursing backwards, choose the previous state $n-1$ that makes the chosen state at n most likely.

Viterbi Algorithm

- Forward pass:
  1. $\omega(z_{1k}) = \ln \pi_k + \ln p(\boldsymbol{x}_1 | z_1 = k)$
  2. For $n = 2$ to N compute for each state value k:
     - 2.1 $\omega(z_{nk}) = \max_l \left\{\omega(z_{n-1,l}) + \ln p(z_n = k | z_{n-1} = l)\right\} + \ln p(\boldsymbol{x}_n | z_n = k)$.
     - 2.2 $\psi(z_{nk}) = \operatorname*{argmax}_l \left\{\omega(z_{n-1,l}) + \ln p(z_n = k | z_{n-1} = l)\right\} + \ln p(\boldsymbol{x}_n | z_n = k)$.
     - 2.3 Passes messages from the start of the sequence to the end.
- Backward pass:
  1. Most likely final state: $\hat{z}_N = \operatorname*{argmax}_k \omega(z_{Nk})$.
  2. For $n = N - 1$ to 1: $\hat{z}_n = \psi(z_{n+1,\hat{z}_{n+1}})$.
- There are multiple paths leading to each possible state at each step n. We keep only the path with the highest probability, so we don't have to compute the likelihood of every complete path from 1 to N.

Summary

By computing sums and maximums at each timestep we can perform inference over an exponential number of sequences. We use the...

- Forward-backward algorithm, an instance of the more general *sum-product* algorithm to marginalise over sequences of hidden states.
- Viterbi algorithm, an instance of the more general *max-sum* algorithm to find the most likely sequence of hidden states.

Reading

- Bishop §13.1
- Bishop §13.2 up to §13.2.2
- Bishop §13.2.5
- Murphy **Book 2** [Mur23] §29.1
- Murphy **Book 2** §29.2
- Murphy **Book 2** §29.4.1

Problems and quizzes

- Bishop Exercise 13.6
- Bishop Exercise 13.7
- Quizzes:
  - Week 5: The EM algorithm

Christopher M. Bishop.
Pattern Recognition and Machine Learning.
Springer, 2006.

Kevin P. Murphy.
Probabilistic Machine Learning: Advanced Topics.
MIT Press, 2023.