Latent Variable Models and Hidden Markov Models

Mark Gales

Lent 2014

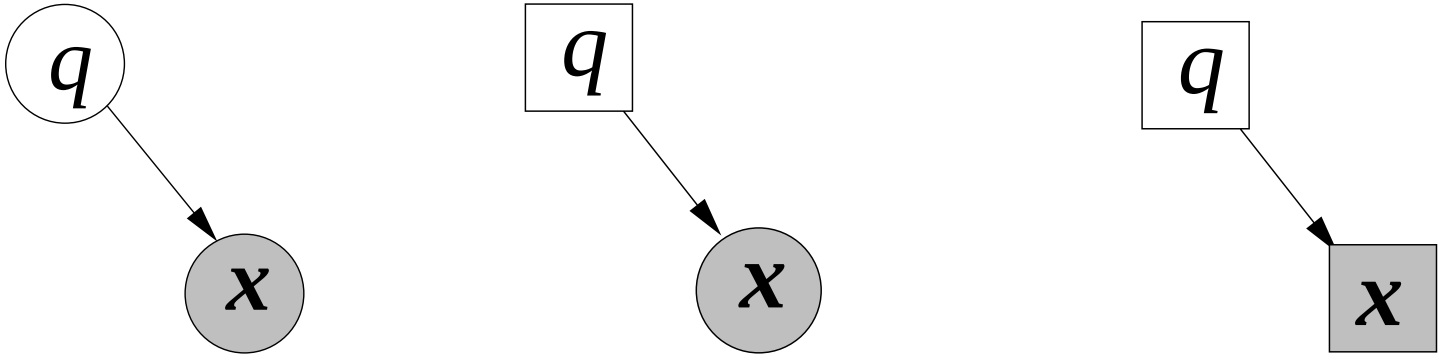
Machine Learning for Language Processing: Lecture 4

MPhil in Advanced Computer Science

Latent Variable Models

- The models generated to date have "meaning" for each variable
  - for topic detection, topic and words in text
- 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
  - may be discrete (mixture models, HMMs), continuous (factor-analysis)
- This lecture will concentrate on two forms model
  - mixture models
  - hidden Markov models

"Static" Latent Variable Models



Factor Analysis   Gaussian Mixture Model   Discrete Mixture Model

- Consider three forms of Byesian Network (BN) for an observation $\boldsymbol{x}$
  - indicator variable q (or $\boldsymbol{q}$) shows value of continuous $\boldsymbol{z}$ or discrete c_m space
  - probability found by marginalising over the latent variable

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

  - these models are extensively used in many machine learning applications

Gaussian Mixture Models

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

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

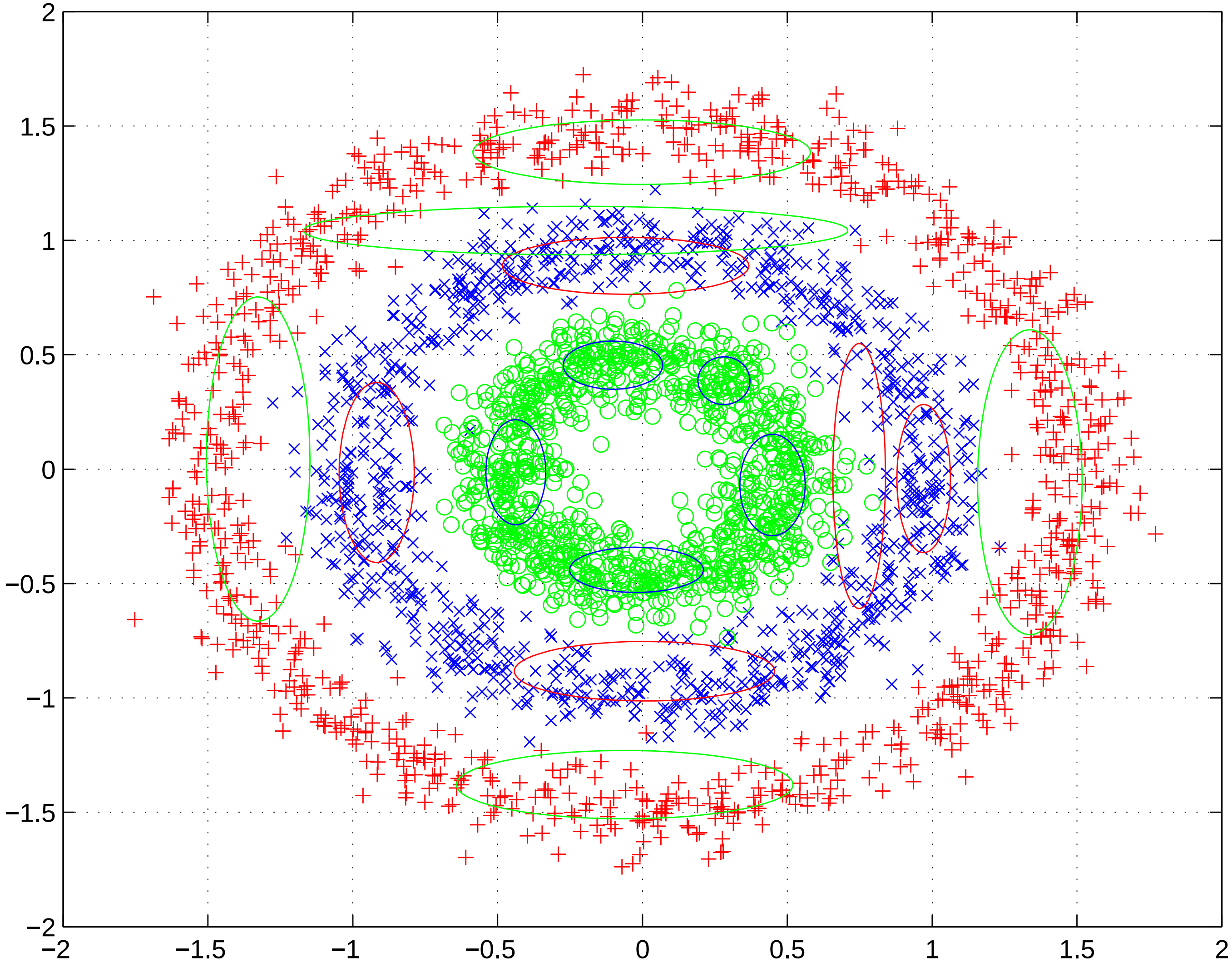
- For GMM each component modelled using a Gaussian distribution

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

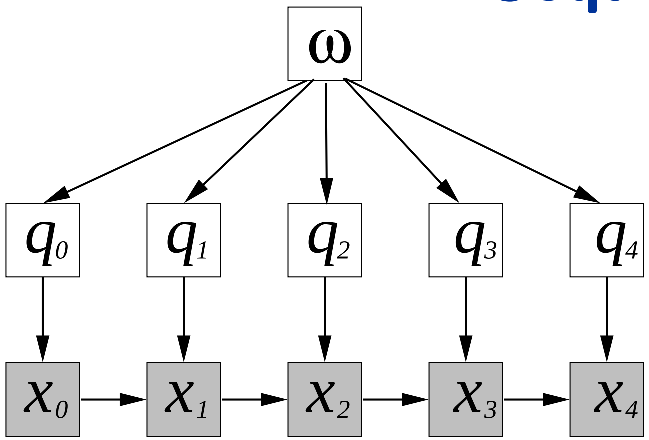
  - component prior: $P(\mathsf{c}_m)$
  - component distribution: $p(\boldsymbol{x}|\mathsf{c}_m) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$

- Highly flexible model, able to model wide-range of distributions

Classifying Doughnut Data using GMMs



Sequence Mixture Models



- Add latent variable to a sequence classifier
  - sequence $x_1, \ldots, x_3$, (x_0 start x_4 end)
  - feature additionally dependent on latent variable
  - latent variable is not observed

- Consider conditional independence and marginalising over the latent variable

$$\begin{aligned} P(x_i|x_o, \ldots, x_{i-1}, q_0, \ldots, q_i, \omega_j) &= P(x_i|x_{i-1}, q_i) \\ P(x_i|x_{i-1}, \omega_j) &= \sum_{m=1}^{M} P(\mathsf{c}_m|\omega_j)P(x_i|x_{i-1}, \mathsf{c}_m) \end{aligned}$$

- So the overall probability (similar to a mixture-model class-dependent LM)

$$P(\boldsymbol{x}|\omega_j) = \prod_{i=1}^{4} \left(\sum_{m=1}^{M} P(\mathsf{c}_m|\omega_j)P(x_i|x_{i-1}, \mathsf{c}_m) \right); \quad \text{Note } P(x_0|\omega_j) = 1$$

Mixture Language Model

- The general form of a mixture language model (for a trigram) is:

$$P(w_k|w_i, w_j) = \sum_{m=1}^{M} \lambda_m P_m(w_k|w_i, w_j); \quad \lambda_m = P(\mathsf{c}_m)$$

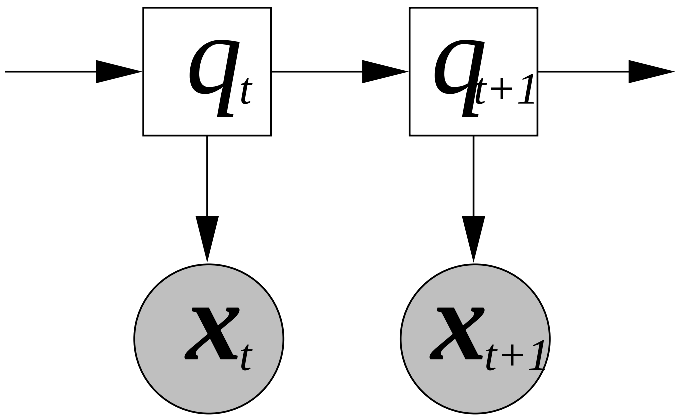
  - M is the number of mixture components
  - $P_m(w_k|w_i, w_j)$ is the language model probability for component m
  - λ_m is the language model component prior (tuned for the task) - note

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

- Each of the individual component language is trained on a different sources
- Component prior, λ_m, tuned for a particular task using development data

Hidden Markov Models

- An important model for sequence data is the hidden Markov model (HMM)
  - an example of a dynamic Bayesian network (DBN)
  - consider a sequence of multi-dimensional 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

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

$\boldsymbol{q} = \{q_0, \ldots, q_{T+1}\}$ and $\boldsymbol{Q}_T$ is all possible state sequences for T observations

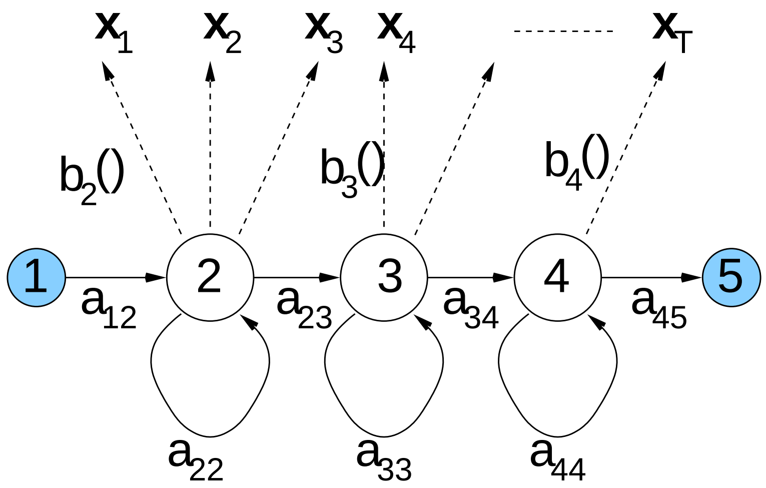
HMM Parameters

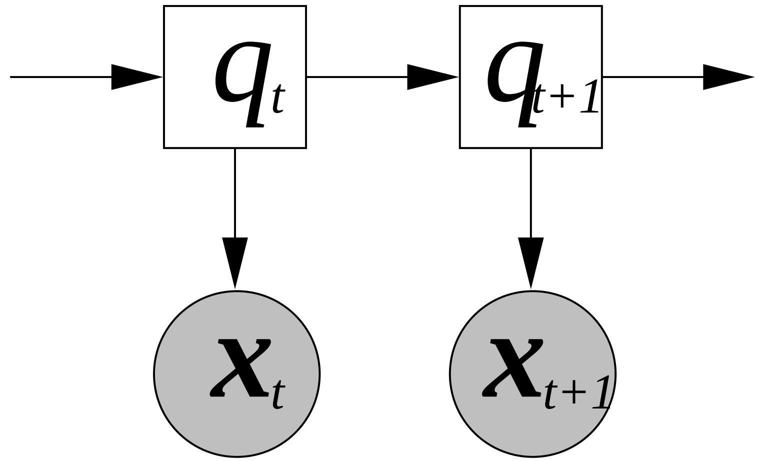
- Two types of states are often defined for HMMs (total N states)
  - emitting states: produce the observation sequence
  - non-emitting states: used to define valid state and end states

- The parameters are normally split into two (assume $\mathtt{s}_1$ and $\mathtt{s}_N$ are non-emitting)
  - transition matrix $\boldsymbol{A}$:
    $a_{ij} = P(q_t = \mathtt{s}_j | q_{t-1} = \mathtt{s}_i)$ is the probability of transitioning from state $\mathtt{s}_i$ to state $\mathtt{s}_j$
  - state output probability $\{b_2(\boldsymbol{x}_t), \ldots, b_{N-1}(\boldsymbol{x}_t)\}$:
    $b_j(\boldsymbol{x}_t) = p(\boldsymbol{x}_t | q_t = \mathtt{s}_j)$ is the output distribution for state $\mathtt{s}_j$

- The estimation of the parameters $\boldsymbol{\lambda} = \{\boldsymbol{A}, b_2(\boldsymbol{x}_t), \ldots, b_{N-1}(\boldsymbol{x}_t)\}$ will be discussed later in the course
  - usually trained using Expectation-Maximisation (EM)

Hidden Markov Model





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

$$\hat{\omega} = \underset{\omega}{\operatorname{argmax}} \{P(\omega|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)\} = \underset{\omega}{\operatorname{argmax}} \{P(\omega)p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T|\omega)\}$$

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

Viterbi Approximation

- An important technique for HMMs (and other models) is the Viterbi Algorithm
  - here the likelihood is approximated as (ignoring dependence on class ω)

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

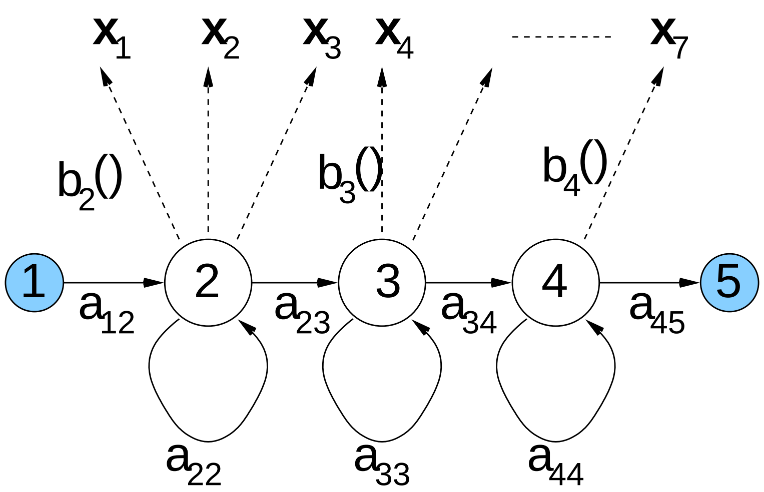
  where

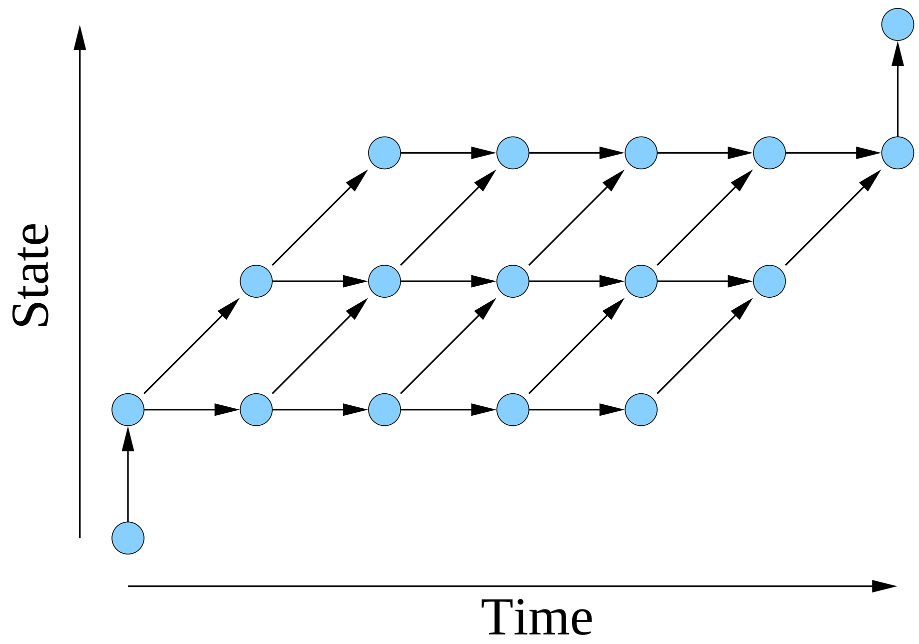
$$\hat{\boldsymbol{q}} = \{\hat{q}_0, \dots, \hat{q}_{T+1}\} = \underset{\boldsymbol{q} \in \boldsymbol{Q}_T}{\operatorname{argmax}} \{p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T, \boldsymbol{q})\}$$

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

Viterbi Algorithm

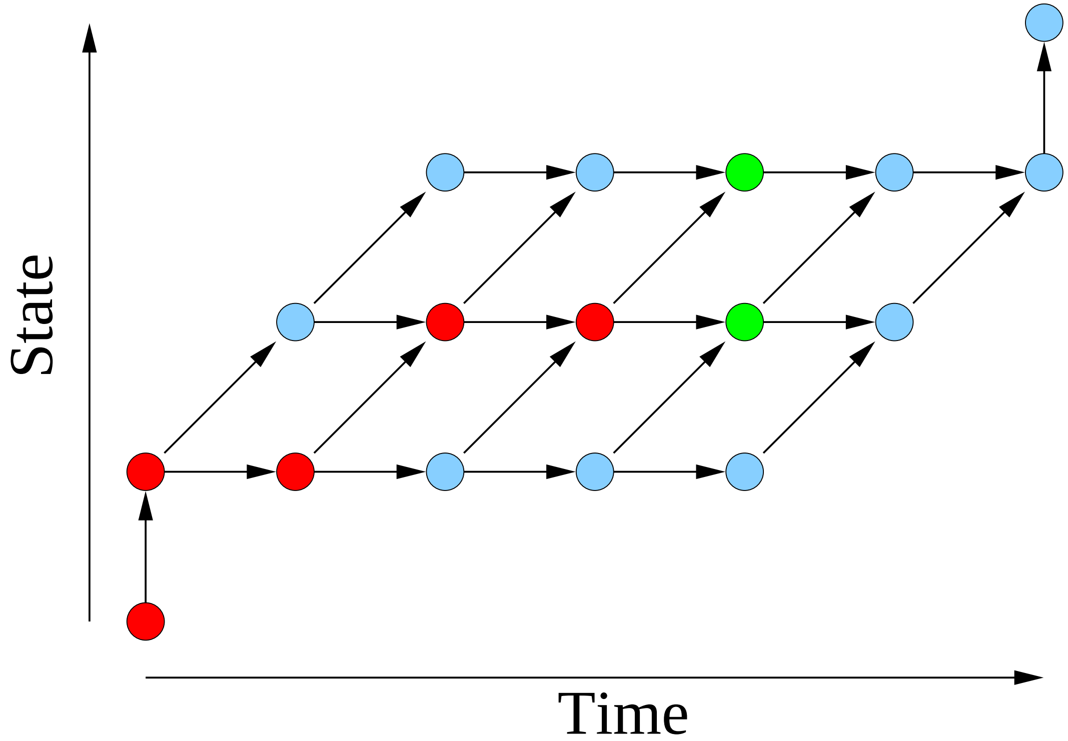
- Need an efficient approach to obtaining the best state-sequence, $\hat{q}$,
  - simply searching through all possible state-sequences impractical ...





- Consider generating the observation sequence $x_1, \ldots, x_7$
  - HMM topology - 3 emitting states with strict left-to-right topology (left)
  - representation of all possible state sequences on the right

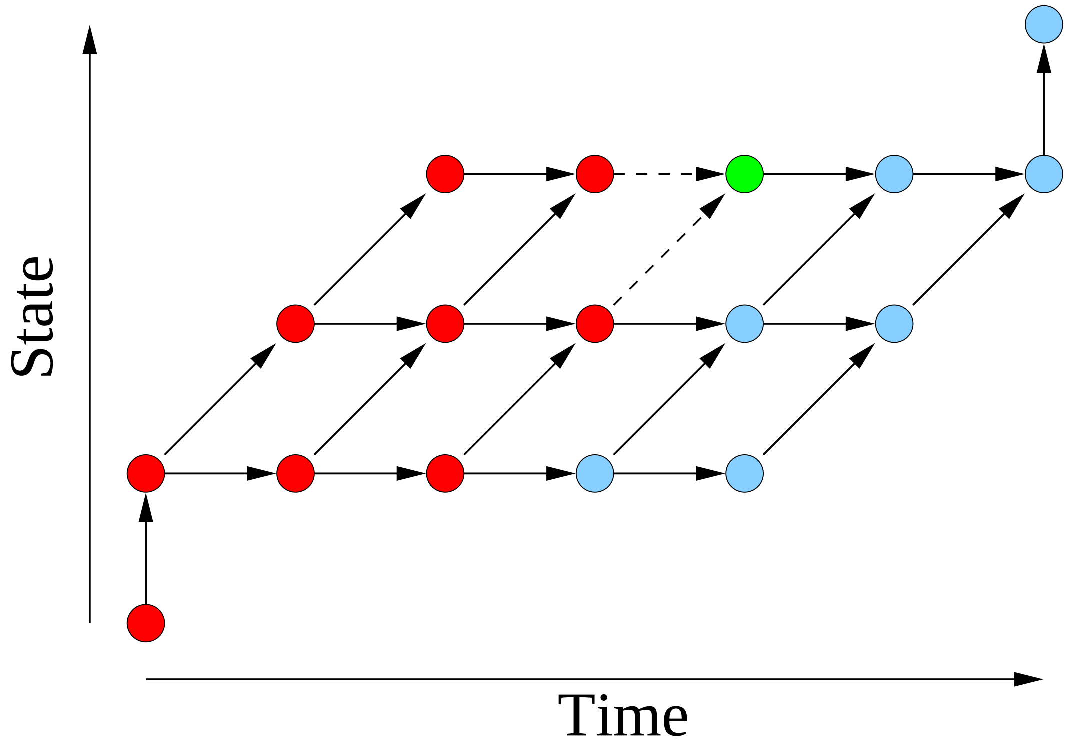
Extending Partial Paths with Time



- Red partial path to time 4
- Green possible extensions

- Partial path state sequence $\{1, 2, 2, 3, 3\}$ with cost $\phi_3(4)$: now extend path
  - cost of staying in state s_3 and generating observation $\boldsymbol{x}_5$: $\log(a_{33}b_3(\boldsymbol{x}_5))$
  - cost of moving to state s_4 and generating observation $\boldsymbol{x}_5$: $\log(a_{34}b_4(\boldsymbol{x}_5))$
- Hence: $\phi_3(5) = \phi_3(4) + \log(a_{33}b_3(\boldsymbol{x}_5))$ and $\phi_4(5) = \phi_3(4) + \log(a_{34}b_4(\boldsymbol{x}_5))$

Best Partial Path to a State/Time



- Red possible partial paths
- Green state of interest

- Require best partial path to state $\mathtt{s}_4$ at time 5 (with associated cost $\phi_4(5)$)
  - cost of moving from state $\mathtt{s}_3$ and generating observation $\boldsymbol{x}_5$: $\log(a_{34}b_4(\boldsymbol{x}_5))$
  - cost of staying in state $\mathtt{s}_4$ and generating observation $\boldsymbol{x}_5$: $\log(a_{44}b_4(\boldsymbol{x}_5))$
- Select "best: $\phi_4(5) = \max\left\{\phi_3(4) + \log(a_{34}b_4(\boldsymbol{x}_5)), \phi_4(4) + \log(a_{44}b_4(\boldsymbol{x}_5))\right\}$

Viterbi Algorithm for HMMs

- The Viterbi algorithm for HMMs can then be expressed as:
  - Initialisation: $(\texttt{LZERO}=\log(0))$
    $\phi_1(0) = 0.0, \quad \phi_j(0) = \texttt{LZERO}, 1 < j < N,$
    $\phi_1(t) = \texttt{LZERO}, 1 \leq t \leq T$
  - Recursion:
    $\texttt{for } t = 1, \dots, T$
    $\quad \texttt{for } j = 2, \dots, N-1$
    $$\phi_j(t) = \max_{1 \leq k < N} \{\phi_k(t-1) + \log(a_{kj})\} + \log(b_j(\boldsymbol{x}_t))$$
  - Termination:
    $$\log(p(\boldsymbol{x}_1, \dots, \boldsymbol{x}_T, \hat{\boldsymbol{q}})) = \max_{1<k<N} \{\phi_k(T) + \log(a_{kN})\}$$
- Can also store the best previous state to allow best sequence $\hat{\boldsymbol{q}}$ to be found.

State-Space

- The state-space can define many different attributes e.g.
  - sub-parts of phones/words/sentences in a speech recognition system
  - part-of-speech tags
  - word-alignments in machine translation
  - named entities

- HMMs can be combined together to form models of sequences of labels
  - many "classes" can be formed from combining sub-classes together
  - for examples words into phones

```
speech task = /s/ /p/ /iy/ /ch/ /t/ /ae/ /s/ /k/
```

  - number of observations and labels do not need to be the same