Probabilistic Graphical Models

Lecture 8: State-Space Models

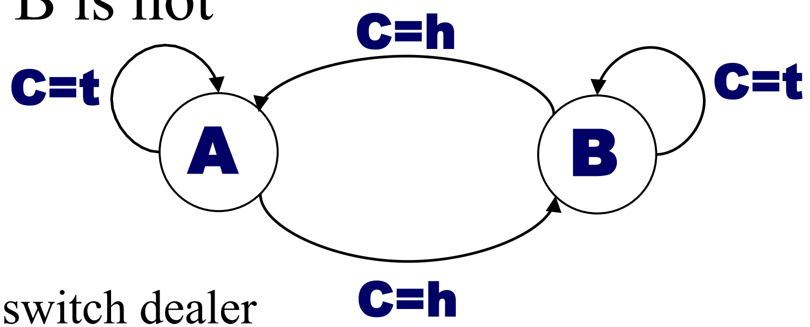
Based on slides by Richard Zemel

Sequential data

Turn attention to sequential data

- Time-series: stock market, speech, video analysis
- Ordered: text, gene

Simple example: Dealer A is fair; Dealer B is not



Process (let Z be dealer A or B):

Loop until tired:

1. Flip coin C, use it to decide whether to switch dealer
2. Chosen dealer rolls die, record result

Fully observable formulation: data is sequence of dealer selections

AAAABBBBAABBBBBBBAAAAABBBBB

Simple example: Markov model

- If underlying process unknown, can construct model to predict next letter in sequence
- In general, product rule expresses joint distribution for sequence

$$P(X_1, X_2, ..., X_T) = \prod_{t=1}^{T} P(X_t | X_{t-1}, ..., X_1)$$

- *First-order Markov chain*: each observation independent of all previous observations except most recent

$$P(X_t | X_{t-1}, ..., X_1) = P(X_t | X_{t-1})$$

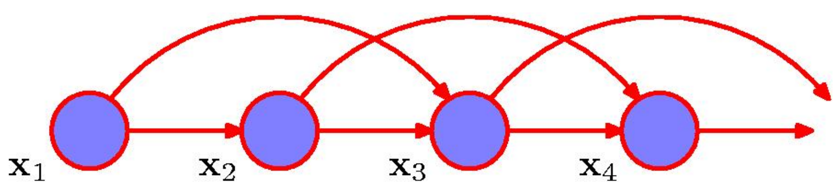
- ML parameter estimates are easy

- Each pair of outputs is a training case; in this example:

  $P(X_t = B | X_{t-1} = A) = \#[t \text{ s.t. } X_t = B, X_{t-1} = A] / \#[t \text{ s.t. } X_{t-1} = A]$

Higher-order Markov models

- Consider example of text
- Can capture some regularities with *bigrams* (e.g., **q** nearly always followed by **u**, very rarely by **j**) – probability of a letter given just its preceding letter
- But probability of a letter depends on more than just previous letter
- Can formulate as *second-order* Markov model (*trigram* model)



- Need to take care: many counts may be zero in training dataset

Character recognition: Transition probabilities

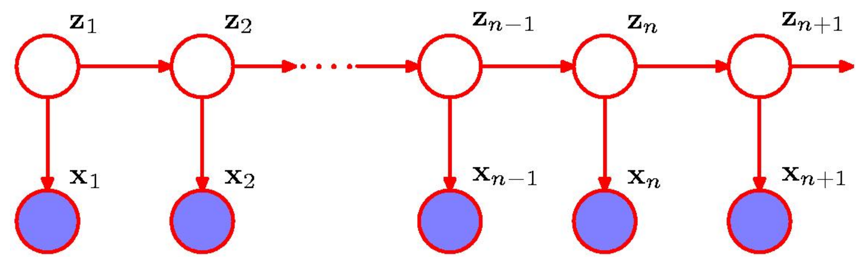
Table 3
Bigrams as Graphemes

Grapheme	Relative frequency as a string	Count as a string	Time occurs as a grapheme (%)	Example used as grapheme	Time combined to form grapheme (%)
AE	0.0002	39.0	84.62	**AE** R O	0.00
AH	0.0001	20.0	20.00	AU T O B **AH** N	0.00
AI	0.0017	325.0	83.08	W **AI** T	10.46
AL	0.0084	1,623.0	40.11	F R U G **AL**	0.00
AO	0.0000	8.0	25.00	E +X+ T R **AO** R D I N A R Y	0.00
AU	0.0012	239.0	81.17	S **AU** C E R	18.83
AW	0.0006	108.0	77.31	F L **AW**	3.70
AY	0.0008	160.0	99.38	E SS **AY**	0.62
BB	0.0004	87.0	99.43	R A **BB** I T	0.00
BT	0.0001	18.0	44.44	D E **BT**	0.00
CC	0.0005	99.0	72.73	S O **CC** E R	3.03
CE	0.0037	726.0	0.62	O **CE** A N	0.00
CH	0.0037	719.0	89.01	A R **CH**	10.99
CI	0.0024	473.0	16.03	F A **CI** AL	0.85
CK	0.0018	344.0	99.27	P I **CK**	0.00
CQ	0.0000	6.0	50.00	A **CQ** U I T	50.00
CS	0.0002	32.0	50.00	O P T I +**CS**+[a]	0.00
CT	0.0025	479.0	0.21	I N D I **CT**	0.00
CZ	0.0000	2.0	100.00	**CZ** A R	0.00
DD	0.0005	100.0	97.00	O **DD**	0.00
DG	0.0003	60.0	100.00	M I **DG** E T	0.00
DI	0.0039	750.0	0.33	C O R **DI** AL	0.00
DJ	0.0001	18.0	94.44	A **DJ** U S T	0.00
EA	0.0031	606.0	72.61	T W **EA** K	9.49
ED	0.0027	515.0	4.27	V O I C **ED**	0.00
EE	0.0017	337.0	93.62	D **EE** R	4.60
EI	0.0007	133.0	50.00	H **EI** S T	26.32
EL	0.0035	682.0	20.53	L E V **EL**	0.00
EN	0.0085	1,643.0	8.22	EA T **EN**	0.00
EO	0.0007	140.0	22.50	P I G **EO** N	6.43
ES	0.0043	831.0	3.53	H O O V **ES**	0.00
ET	0.0039	762.0	2.23	S A CH **ET**	0.00
EU	0.0004	86.0	81.40	F **EU** D	4.65
EW	0.0006	108.0	63.89	G R **EW**	7.41
EY	0.0004	81.0	92.59	M O N **EY**	7.41
FF	0.0009	176.0	100.00	O **FF**	0.00
FT	0.0004	69.0	1.45	S O **FT** E N	0.00
GG	0.0004	73.0	98.63	J U **GG** LE	0.00
GH	0.0009	171.0	12.28	R OU **GH**	82.46
GI	0.0014	267.0	8.05	R E **GI** O N	0.00
GM	0.0001	28.0	14.29	PH L E **GM**	0.00
GN	0.0006	126.0	33.33	**GN** A SH	0.00
IA	0.0034	667.0	6.37	P A R L **IA** M E N T	0.45
IE	0.0018	340.0	41.13	B R **IE** F	12.50
IL	0.0038	737.0	6.51	V I G **IL**	0.00
IN	0.0113	2,188.0	3.20	L A T **IN**	0.00
KH	0.0001	14.0	57.14	SH EI **KH**	0.00
KN	0.0002	43.0	94.19	**KN** O T	0.00
LD	0.0006	121.0	3.72	W OU **LD**	0.00
LE	0.0085	1,652.0	43.70	P I CK **LE**	0.00
LF	0.0002	46.0	13.04	H A **LF**	0.00
LK	0.0002	47.0	48.94	W A **LK**	0.00
LL	0.0033	649.0	97.92	D W E **LL**	0.00
LM	0.0002	48.0	29.17	P A **LM**	0.00
LV	0.0003	55.0	10.91	C A **LV** E S	0.00
MB	0.0011	221.0	16.06	TH U **MB**	0.00
MM	0.0009	178.0	98.88	R U **MM** Y	0.00
MN	0.0002	43.0	25.58	AU T U **MN**	0.00
NG	0.0032	616.0	58.44	P I **NG**	0.32
NN	0.0008	164.0	97.56	T E **NN** I S	0.00
OA	0.0008	148.0	87.50	L **OA** N	2.36
OE	0.0003	61.0	41.80	SH **OE**	0.00
OH	0.0001	19.0	31.58	**OH** M	0.00

Hidden Markov model (HMM)

- Return to casino example -- now imagine that do not observe ABBAA, but instead just sequence of die rolls (1-6)

- Generative process:
  Loop until tired:
    1. Flip coin C (Z = A or B)
    2. Chosen dealer rolls die, record result X



Z is now hidden *state* variable – 1st order Markov chain generates state sequence (path), governed by *transition matrix* **A**

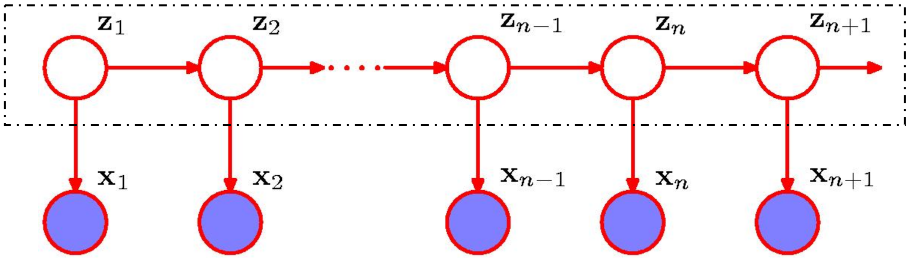
$$P(Z_t = k | Z_{t-1} = j) = A_{jk}$$

$$\text{State as multinomial variable}: \quad P(\mathbf{z}_t | \mathbf{z}_{t-1}) = \prod_k \prod_j A_{jk}^{z_{t-1,j} z_{t,k}}$$

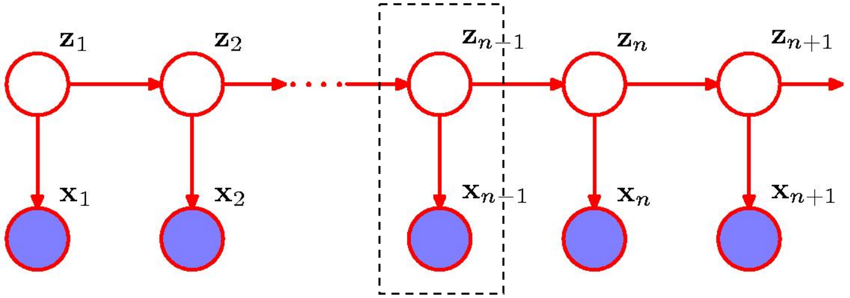
Observations governed by *emission probabilities,* convert state path into sequence of observable symbols or vectors: $P(X_t | Z_t)$

Relationship to other models

- Can think of HMM as:
  - Markov chain with stochastic measurements

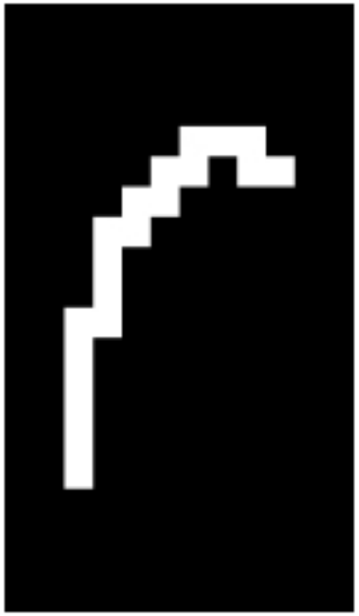


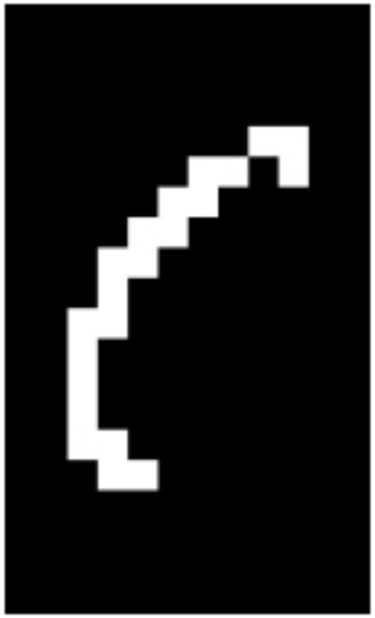
  - Mixture model with states coupled across time



- Hidden state is 1^{st}-order Markov, but output not Markov of any order
- Future is independent of past give present, but conditioning on observations couples hidden states

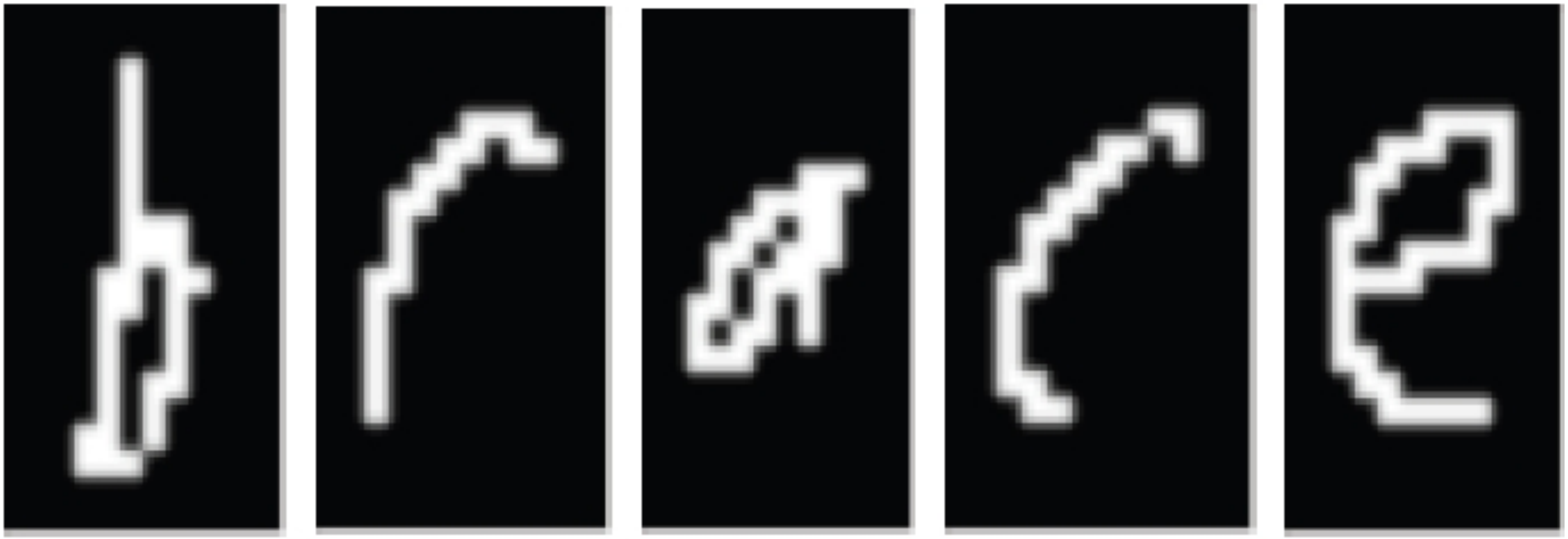
Character Recognition Example





Which letters are these?

HMM: Character Recognition Example

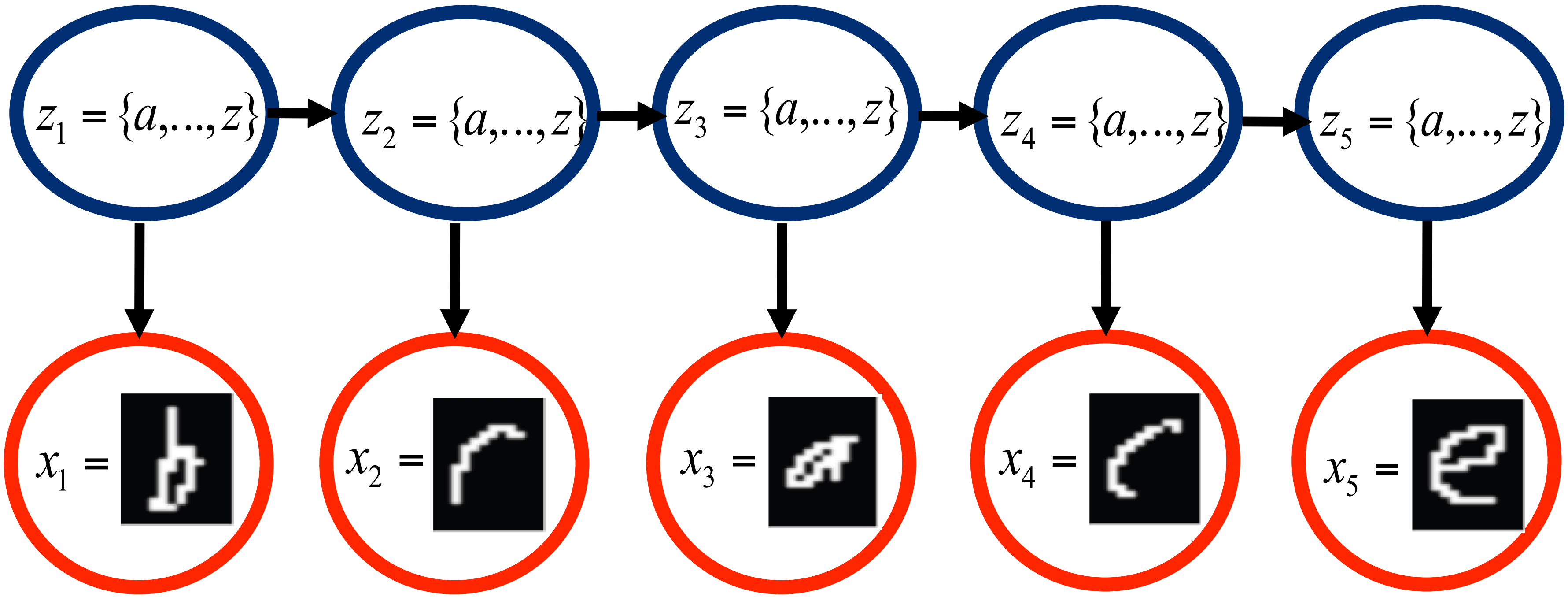


Context matters: recognition easier based on sequence of characters

How to apply HMM to this character string?

Main elements: states? emission, transition probabilities?

HMM: Semantics



Need 3 distributions:

1. Initial state: $P(Z_1)$
2. Transition model: $P(Z_t | Z_{t-1})$
3. Observation model (emission probabilities): $P(X_t | Z_t)$

HMM: Main tasks

- Joint probabilities of hidden states and outputs:

$$P(\mathbf{x},\mathbf{z}) = P(z_1)P(x_1 \mid z_1)\prod_{t=2}^{T} P(z_t \mid z_{t-1})P(x_t \mid z_t)$$

- Three problems
  1. Computing probability of observed sequence: forward-backward algorithm [good for recognition]
  2. Infer most likely hidden state sequence: Viterbi algorithm [useful for interpretation]
  3. Learning parameters: Baum-Welch algorithm (version of EM)

Fully observed HMM

Learning fully observed HMM (observe both X and Z) is easy:

1. Initial state: $P(Z_1)$ – proportion of words start with each letter
2. Transition model: $P(Z_t | Z_{t-1})$ – proportion of times a given letter follows another (bigram statistics)
3. Observation model (emission probabilities): $P(X_t | Z_t)$ – how often particular image represents specific character, relative to all images

But still have to do inference at test time: work out states given observations

HMMs often used where hidden states are identified: words in speech recognition; activity recognition; spatial position of rat; genes; POS tagging

HMM: Inference tasks

Important to infer distributions over hidden states:

- If states are interpretable, infer interpretations
- Also essential for learning

Can break down hidden state inference tasks to solve (each based on all observations up to current time, $X_{0:t}$)

1. Filtering: compute posterior over *current* hidden state: $P(Z_t \mid X_{0:t})$
2. Prediction: compute posterior over *future* hidden state: $P(Z_{t+k} \mid X_{0:t})$
3. Smoothing: compute posterior over *past* hidden state: $P(Z_k \mid X_{0:t})$, $0<k<t$
4. Fixed-lag smoothing: $P(Z_{t-a} \mid X_{0:t})$: compute posterior over hidden state a few steps back

Filtering, Smoothing & Prediction

$$P(Z_t \mid X_{1:t}) = P(Z_t \mid X_t, X_{1:t-1})$$

$$\propto P(X_t \mid Z_t, X_{1:t-1}) P(Z_t \mid X_{1:t-1})$$

$$= P(X_t \mid Z_t) P(Z_t \mid X_{1:t-1})$$

$$= P(X_t \mid Z_t) \sum_{z_{t-1}} P(Z_t \mid z_{t-1}, X_{1:t-1}) P(z_{t-1} \mid X_{1:t-1})$$

$$= P(X_t \mid Z_t) \sum_{z_{t-1}} P(Z_t \mid z_{t-1}) P(z_{t-1} \mid X_{1:t-1})$$

Filtering: for **online** estimation of state

Pr(state) =observation probability * transition-model

Smoothing: **post hoc** estimation of state (similar computation)

Prediction is filtering, but with no new evidence:

$$P(Z_{t+k} \mid X_{1:t}) = \sum_{z_{t+k-1}} P(Z_{t+k} \mid z_{t+k-1}) P(z_{t+k-1} \mid X_{1:t})$$

HMM: Maximum likelihood

Having observed some dataset, use ML to learn the parameters of the HMM

Need to marginalize over the latent variables:

$$p(\mathbf{X}|\theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\theta)$$

Difficult:

- does not factorize over time steps
- involves generalization of a mixture model

Approach: utilize EM for learning

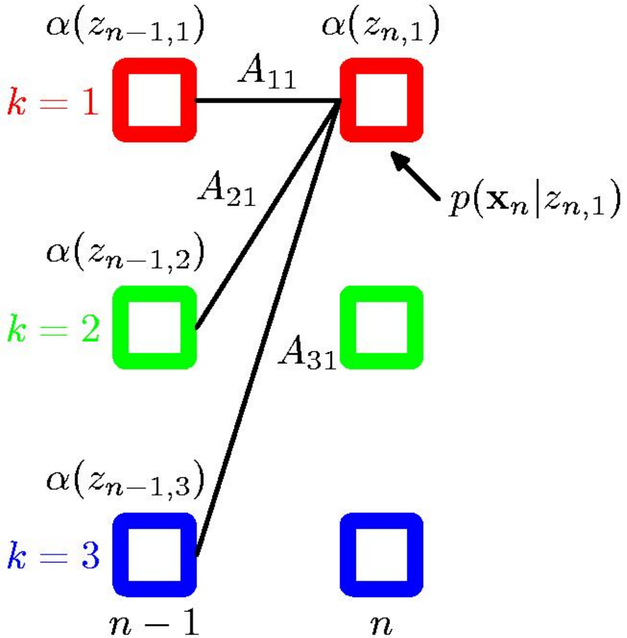
Focus first on how to do inference efficiently

Forward recursion (α)

$$\text{Define } \alpha(z_{t,j}) = P(x_1, \ldots, x_t, z_t = j)$$

Clever recursion can compute huge sum efficiently

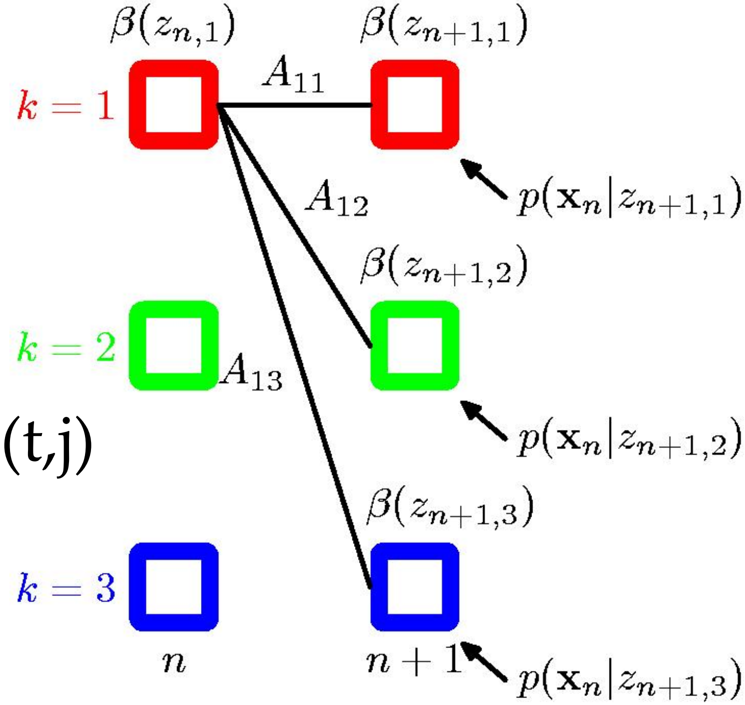
$$\begin{aligned}
\alpha(z_{1,j}) &= P(x_1, z_1 = j) = P(x_1 | z_1 = j) P(z_1 = j) \\
\alpha(z_{2,j}) &= P(x_2 | z_2 = j) \left[\sum_k P(z_2 = j | z_1 = k) P(x_1 | z_1 = k) P(z_1 = k) \right] \\
&= P(x_2 | z_2 = j) \left[\sum_k A_{kj} \alpha(z_{1,k}) \right] \\
\alpha(z_{t+1,j}) &= P(x_{t+1} | z_{t+1} = j) \left[\sum_k A_{kj} \alpha(z_{t,k}) \right]
\end{aligned}$$



Backward recursion (β)

Define $\beta(z_{t,j}) = P(x_{t+1}, ..., x_T | z_t = j)$

$$\begin{aligned}\beta(z_{t,j}) &= \left[\sum_k A_{jk} P(x_{t+1} | z_{t+1} = k) \beta(z_{t+1,k})\right] \\ \beta(z_{T,j}) &= 1\end{aligned}$$



α$(z_{t,j})$: total inflow of prob. to node (t,j)

β$(z_{t,j})$: total outflow of prob. from node (t,j)

Forward-Backward algorithm

Estimate hidden state given observations

Define $\gamma(z_{t,i}) = P(z_t = i | x_1, ..., x_T)$

$$\begin{aligned}
\gamma(z_{t,i}) &= P(\mathbf{X}|z_t = i)P(z_t = i)/P(\mathbf{X}) \\
&= P(x_1, ..., x_t|z_t = i)P(x_{t+1}, ..., x_T|z_t = i)P(z_t = i)/P(\mathbf{X}) \\
&= P(x_1, ..., x_t, z_t = i)P(x_{t+1}, ..., x_T|z_t = i)/P(\mathbf{X}) \\
&= \alpha(z_{t,i})\beta(z_{t,i})/P(\mathbf{X})
\end{aligned}$$

One forward pass to compute all $\alpha(z_{t,i})$, one backward pass to compute all $\beta(z_{t,i})$: total cost $O(K^2T)$

Can compute likelihood at any time t based on $\boldsymbol{\alpha}(z_{t,j})$ and $\boldsymbol{\beta}(z_{t,j})$

$$L = P(\mathbf{X}) = \sum_i \alpha(z_{t,i})\beta(z_{t,i})$$

Baum-Welch training algorithm: Summary

Can estimate HMM parameters using maximum likelihood

If state path known, then parameter estimation easy

Instead must estimate states, update parameters, re-estimate states, etc. ***--*** *Baum-Welch* (form of EM)

State estimation via forward-backward, also need transition statistics (see next slide)

Update parameters (transition matrix **A**, emission parameters) to maximize likelihood

Transition statistics

Need statistics for adjacent time-steps:

Define $\xi(z_{ij}(t)) = P(z_{t-1} = i, z_t = j|\mathbf{X})$

$$\begin{aligned}
\xi(z_{i,j}(t)) &= P(z_{t-1} = i, x_1, ..., x_{t-1}) \\
&\quad P(z_t = j, x_t, ..., x_T | z_{t-1} = i, x_1, ..., x_{t-1}) / P(\mathbf{X}) \\
&= P(z_{t-1} = i, x_1, ..., x_{t-1}) P(z_t = j | z_{t-1} = i) \\
&\quad P(x_t | z_t = j) P(x_{t+1}, ..., x_T | z_t = j) / L \\
&= \alpha(z_{t-1,i}) A_{ij} P(x_t | z_t = j) \beta(z_{t,j}) / L
\end{aligned}$$

Expected number of transitions from state i to state j that begin at time $t-1$, given the observations

Can be computed with the same $\boldsymbol{\alpha}(z_{t,j})$ and $\boldsymbol{\beta}(z_{t,j})$ recursions

Parameter updates

Initial state distribution: expected counts in state k at time 1

$$\pi_k = \frac{\gamma(z_{1,k})}{\sum_{j=1}^{K} \gamma(z_{1,j})}$$

Estimate transition probabilities:

$$A_{ij} = \frac{\sum_{t=2}^{T} \xi(z_{ij}(t))}{\sum_{t=2}^{T} \sum_k \xi(z_{ik}(t))} = \frac{\sum_{t=2}^{T} \xi(z_{ij}(t))}{\sum_{t=2}^{T} \gamma(z_{t,i})}$$

Emission probabilities are expected number of times observe symbol in particular state:

$$\mu_{i,k} = \frac{\sum_{t=1}^{T} \gamma(z_{t,k}) x_{t,i}}{\sum_{t=1}^{T} \gamma(z_{t,k})}$$

Using HMMs for recognition

Can train an HMM to classify a sequence:

1. train a separate HMM per class
2. evaluate prob. of unlabelled sequence under each HMM
3. classify: HMM with highest likelihood

Assumes can solve two problems:

1. estimate model parameters given some training sequences (we can find local maximum of parameter space near initial position)
2. given model, can evaluate prob. of a sequence

Probability of observed sequence

Want to determine if given observation sequence is likely under the model (for learning, or recognition)

Compute marginals to evaluate prob. of observed seq.: sum across all paths of joint prob. of observed outputs and state

$$P(\mathbf{X}) = \sum_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z})$$

Take advantage of factorization to avoid exp. cost (#paths = K^T)

$$\begin{aligned} P(\mathbf{X}) &= \sum_{z_1} \sum_{z_2} \cdots \sum_{z_T} \prod_{t=1}^{T} P(z_t|z_{t-1}) P(x_t|z_t) \\ &= \sum_{z_1} P(z_1) P(x_1|z_1) \sum_{z_2} P(z_2|z_1) P(x_2|z_2) \\ &\qquad \cdots \sum_{z_T} P(z_T|z_{T-1}) P(x_T|z_T) \end{aligned}$$

Variants on basic HMM

- Input-output HMM
  - Have additional observed variables $\boldsymbol{U}$

- Semi-Markov HMM
  - Improve model of state duration

- Autoregressive HMM
  - Allow observations to depend on some previous observations directly

- Factorial HMM
  - Expand dim. of latent state

State Space Models

Instead of discrete latent state of the HMM, model Z as a continuous latent variable

Standard formulation: linear-Gaussian (LDS), with (hidden state ***Z***, observation ***Y***, other variables ***U***)

- Transition model is linear

$$\mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \epsilon_t$$

- with Gaussian noise

$$\epsilon_t = \mathcal{N}(0, \mathbf{Q}_t)$$

- Observation model is linear

$$\mathbf{y}_t = \mathbf{C}_t \mathbf{z}_t + \mathbf{D}_t \mathbf{u}_t + \delta_t$$

- with Gaussian noise

$$\delta_t = \mathcal{N}(0, \mathbf{R}_t)$$

Model parameters typically independent of time: stationary

Kalman Filter

Algorithm for filtering in linear-Gaussian state space model

Everything is Gaussian, so can compute updates exactly

Dynamics update: predict next belief state

$$p(\mathbf{z}_t|\mathbf{y}_{1:t-1}, \mathbf{u}_{1:t}) = \int \mathcal{N}(\mathbf{z}_t|\mathbf{A}_t\mathbf{z}_{t-1} + \mathbf{B}_t\mathbf{u}_t, \mathbf{Q}_t)\mathcal{N}(\mathbf{z}_{t-1}|\mu_{t-1}, \Sigma_{t-1})d\mathbf{z}_{t-1}$$

$$= \mathcal{N}(\mathbf{z}_t|\mu_{t|t-1}, \Sigma_{t|t-1})$$

$$\mu_{t|t-1} = \mathbf{A}_t\mu_{t-1} + \mathbf{B}_t\mathbf{u}_t$$

$$\Sigma_{t|t-1} = \mathbf{A}_t\Sigma_{t-1}\mathbf{A}_t^T + \mathbf{Q}_t$$

Kalman Filter: Measurement Update

Key step: update hidden state given new measurement:

$$p(\mathbf{z}_t|\mathbf{y}_{1:t},\mathbf{u}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{z}_t,\mathbf{u}_t)p(\mathbf{z}_t|\mathbf{y}_{1:t-1},\mathbf{u}_{1:t})$$

First term a bit complicated, but can apply various identities (such as the matrix inversion lemma, Bayes rule), obtain:

$$p(\mathbf{z}_t|\mathbf{y}_{1:t},\mathbf{u}_{1:t}) = \mathcal{N}(\mathbf{z}_t|\mu_t,\Sigma_t)$$

The mean update depends on Kalman gain matrix **K**, and the residual or innovation **r = y – E[y]**

$$\mu_t = \mu_{t|t-1} + \mathbf{K}_t\mathbf{r}_t$$

$$\mathbf{K}_t = \Sigma_{t|t-1}\mathbf{C}_t^T\mathbf{S}_t^{-1}$$

$$\hat{\mathbf{y}} = \mathbb{E}[\mathbf{y}_t|\mathbf{y}_{1:t-1},\mathbf{u}_t] = \mathbf{C}_t\mu_{t|t-1} + \mathbf{D}_t\mathbf{u}_t$$

$$\mathbf{S}_t = \mathrm{cov}[\mathbf{r}_t|y_{1:t-1},\mathbf{u}_{1:t}] = \mathbf{C}_t\Sigma_{t|t-1}\mathbf{C}_t^T + \mathbf{R}_t$$

Kalman Filter: Extensions

Learning similar to HMM

- Need to solve inference problem – local posterior marginals for latent variables
- Use Kalman smoothing instead of forward-backward in E step, re-derive updates in M step

Many extensions and elaborations

- Non-linear models: extended KF, unscented KF
- Non-Gaussian noise
- More general posteriors (multi-modal, discrete, etc.)
- Large systems with sparse structure (sparse information filter)

Viterbi decoding

How to choose single best path through state space?

Choose state with largest probability at each time t: maximize expected number of correct states

But this may not be the best path, with highest likelihood of generating the data

To find best path – *Viterbi decoding*, form of dynamic programming (forward-backward algorithm)

Same recursions, but replace $\sum$ with **max** ("brace" example)

- **Forward:** retain best path into each node at time t
- **Backward:** retrace path back from state where most probable path ends