CSC2515 Fall 2007
Introduction to Machine Learning

Lecture 10: Sequential Data Models

Example: sequential data

Until now, considered data to be i.i.d.

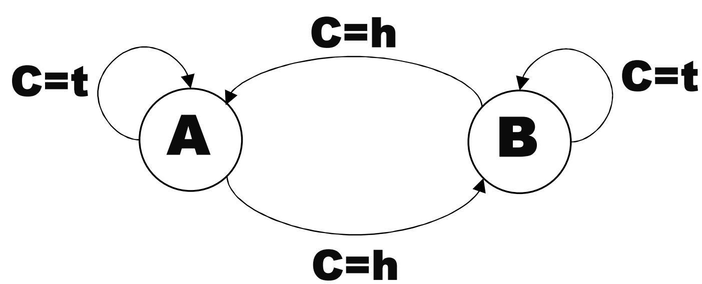
Turn attention to sequential data

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

Simple example: Coins A (p(h) = .6); B (p(h) = .7); C (p(h) = .2)

Process:

1. Let X be coin A or B
2. Loop until tired:
   1. Flip coin X, record result
   2. Flip coin C
   3. If C=heads, switch X



Fully observable formulation: data is sequence of coin 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

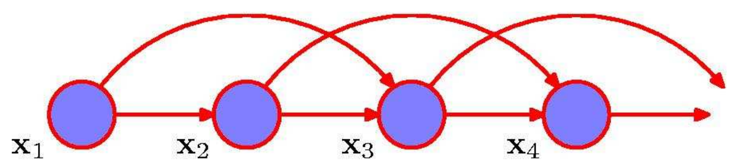
$$P(X_t | X_{t-1}, ..., X_1) = P(X_t | X_{t-1})$$

- 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

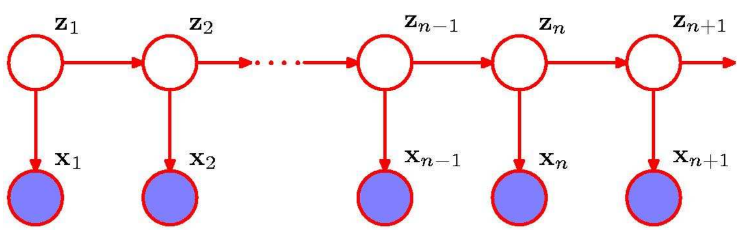
- Consider example of text
- Can capture some regularities with *bigrams* (e.g., **q** nearly always followed by **u**, very rarely by **j**)
- But probability of a letter depends on more than just previous letter
- Can formulate as *second-order* Markov model (*trigram* model)



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

Hidden Markov model (HMM)

- Return to coins example -- now imagine that do not observe ABBAA, but instead sequence of heads/tails
- Generative process:
  1. Let Z be coin A or B
  2. Loop until tired:
     1. Flip coin Z, record result X
     2. Flip coin C
     3. If C=heads, switch Z



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

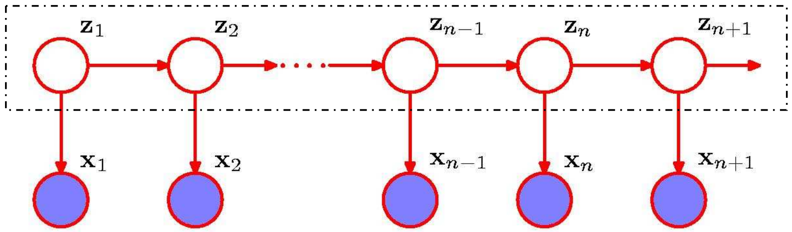
$$P(Z_t = k | Z_{t-1} = j) = A_{jk}$$

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

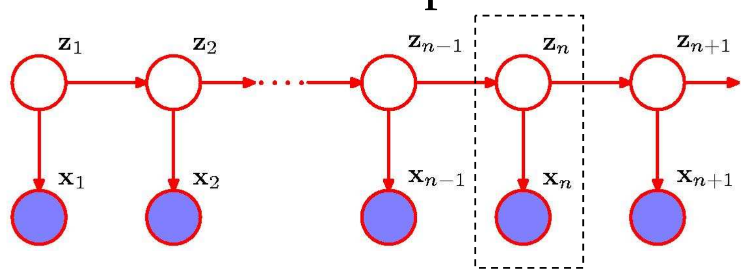
Observations governed by *emission probabilities*, convert state path into sequence of observable symbols or vectors: $P(X_t | Z_t)$

Relationship to other models

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



  - Mixture model with states coupled across time



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

HMM: Main tasks

- Joint probabilities of hidden states and outputs:

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

- Three problems
  1. Computing probability of observed sequence: forward-backward algorithm
  2. Infer most likely hidden state sequence: Viterbi algorithm
  3. Learning parameters: Baum-Welch (EM) algorithm

Probability of observed sequence

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

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

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

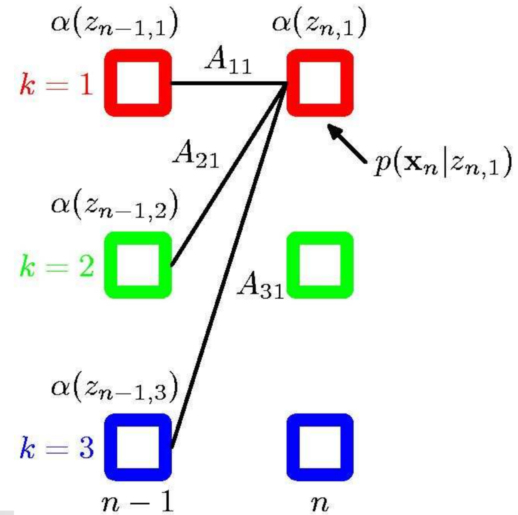
$$\begin{aligned} P(\mathbf{X}) &= \sum_{z_1} \sum_{z_2} \cdots \sum_{z_T} \prod_{t=1}^{T} P(z_t|z_{t-1}) P(x_t|z_t) \\ &= \sum_{z_1} P(z_1) P(x_1|z_1) \sum_{z_2} P(z_2|z_1) P(x_2|z_2) \\ &\quad \cdots \sum_{z_T} P(z_T|z_{T-1}) P(x_T|z_T) \end{aligned}$$

Forward recursion (α)

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

Clever recursion can compute huge sum efficiently

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

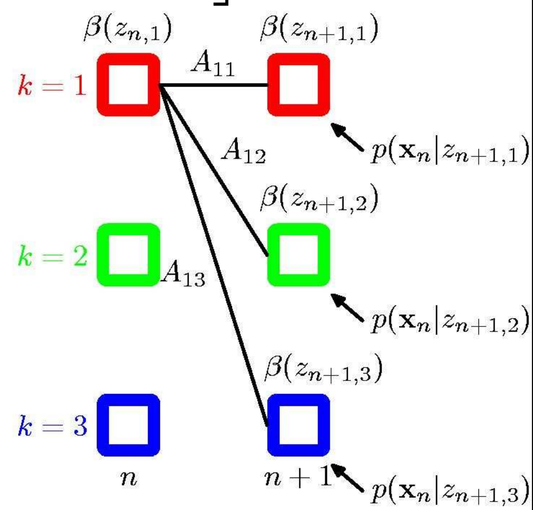


Backward recursion (β)

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

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

$$\beta(z_{T,j}) = 1$$



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

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

Forward-Backward algorithm

Estimate hidden state given observations

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

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

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 $\alpha(z_{t,j})$ and $\beta(z_{t,j})$

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

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. $\rightarrow$ *Baum-Welch* (form of EM)

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

Update parameters (transition matrix **A**, emission parameters ϕ) 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 $\alpha(z_{t,j})$ and $\beta(z_{t,j})$ recursions

Parameter updates

Initial state distribution: expected counts in state *i* 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:

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

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 not single 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** (weather example)

Forward: retain best path into each node at time *t*

Backward: retrace path back from state where most probable path ends

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

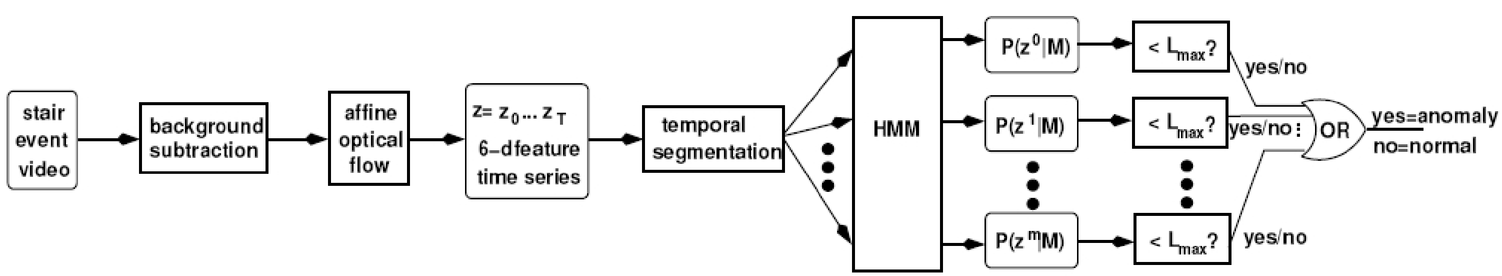
Application example: classifying stair events

Aim: automatically detect unusual events on stairs from video

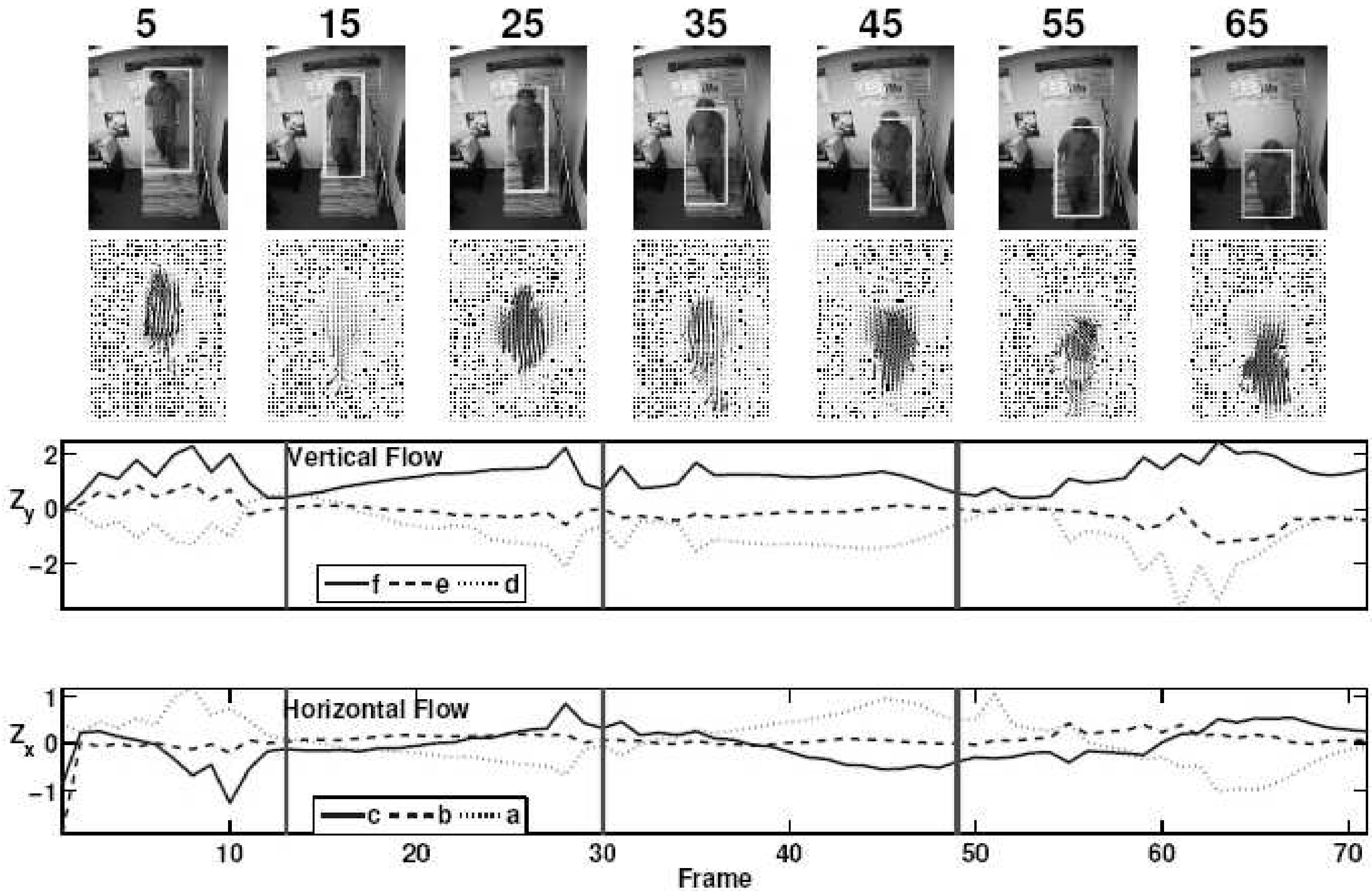
Idea: compute visual features describing person's motion during descent, apply HMM to several sequences of feature values

One-class training:

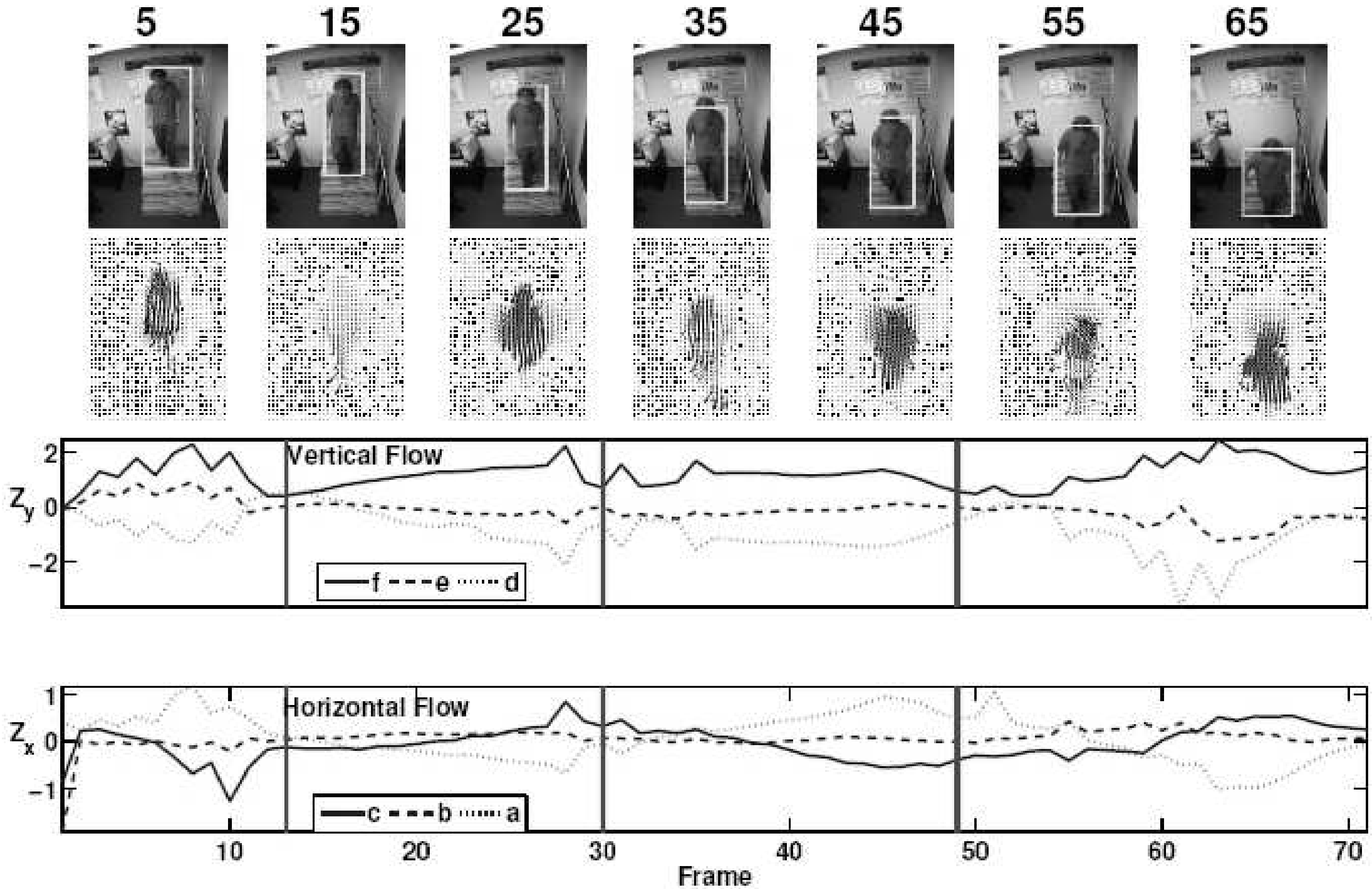
1. train HMM on example sequences from class: *normal* stair descent
2. set likelihood threshold L based on labelled validation set:
$$C(L) = \frac{W}{N_n}\sum_{i=1}^{N_n} g(\log P(\mathbf{X}^i), L) + \frac{(1-W)}{N_a}\sum_{j=1}^{N_a}(1 - g(\log P(\mathbf{X}^j), L))$$
3. classify by thresholding HMM likelihood of test sequence



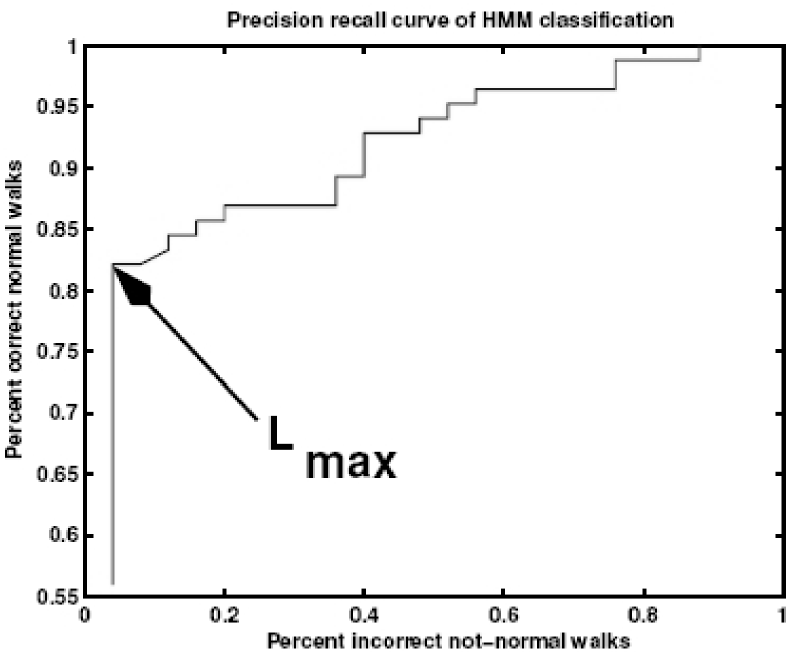
Classifying stair events: Normal event

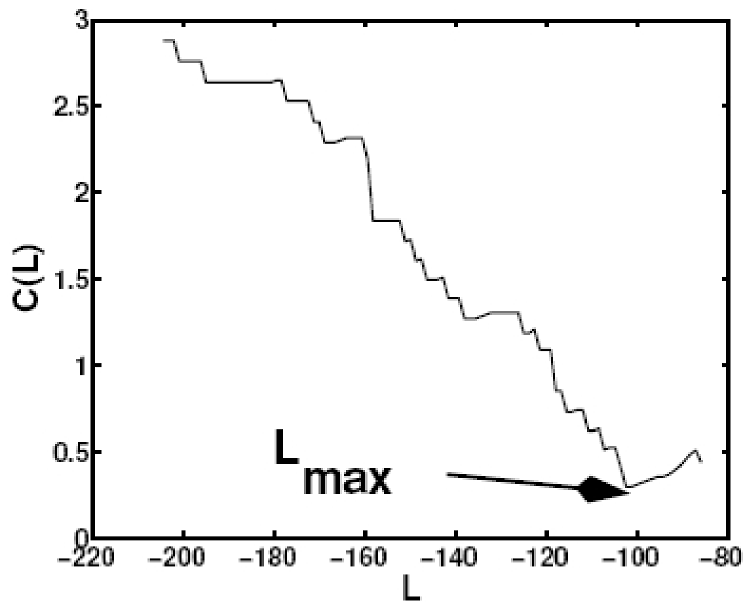


Classifying stair events: Anomalous event



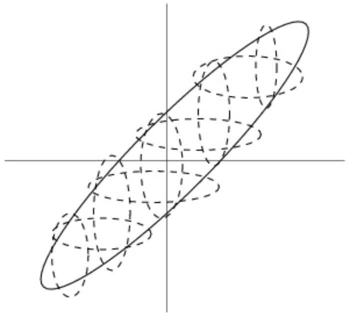
Classifying stair events: Precision-recall





HMM Regularization

1. High dimensional state space:
   - transition matrix has K^2 entries
   - can constrain to be relatively sparse: each state has only a few possible successor states (c)
   - inference now O(cKT), number of parameters O($cK+KM$)
   - can construct state ordering, only allow transitions to later states: “linear”, “chain”, or “left-to-right” HMMs
2. High dimensional observations:
   - in continuous data space, full covariance matrices have many parameters – use mixtures of diagonal covariance Gaussians



HMM Extensions

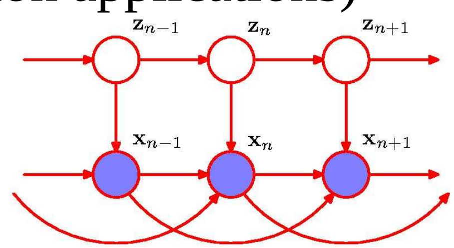
1. Generalize model of state duration:

   - vanilla HMM restricted in model of how long stay in state - prob. that model will spend D steps in state k and then transition out:

$$P(D) = (A_{kk})^D (1 - A_{kk}) \propto \exp(-D \log A_{kk})$$

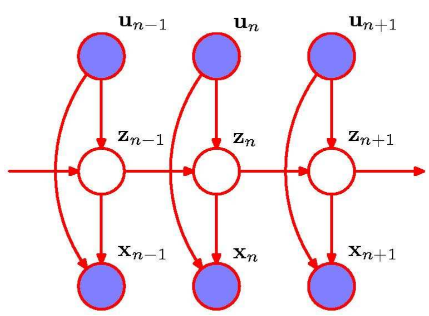
   - instead associate distribution with time spent in state k: P($t|k$) (see *semi-Markov* models for sequence segmentation applications)

2. Combine with auto-regressive Markov model:



   - include long-range relationships
   - directly model relations between observations

3. Supervised setting:

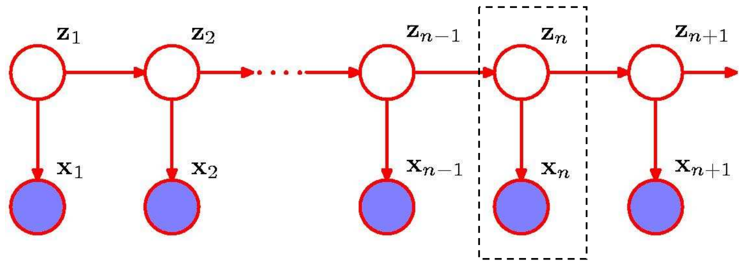


   - include additional observations
   - *input-output* HMM

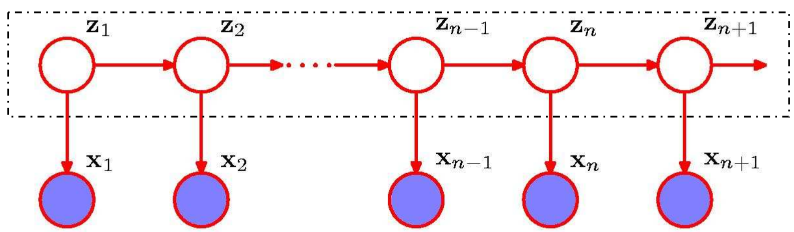
Linear Dynamical Systems

Return to state space model:

- last week's continuous latent variable models, but now not i.i.d.



- view as linear-Gaussian state evolution, continuous-valued, with emissions



LDS generative process

Consider generative process:

$$\begin{aligned}\mathbf{z}_1 &= \mu_0 + \mathbf{u} \\ \mathbf{z}_t &= \mathbf{A}\mathbf{z}_{t-1} + \mathbf{w}_t \\ \mathbf{x}_t &= \mathbf{C}\mathbf{z}_t + \mathbf{v}_t\end{aligned}$$

where **u,** $\mathbf{w}_t$, $\mathbf{v}_t$ are all mean zero Gaussian noise terms

Can express in terms of linear-Gaussian conditional distributions

$$\begin{aligned}p(\mathbf{z}_t|\mathbf{z}_{t-1}) &= \mathcal{N}(\mathbf{z}_t|\mathbf{A}\mathbf{z}_{t-1}, \Gamma) \\ p(\mathbf{x}_t|\mathbf{z}_t) &= \mathcal{N}(\mathbf{x}_t|\mathbf{C}\mathbf{z}_t, \Sigma)\end{aligned}$$