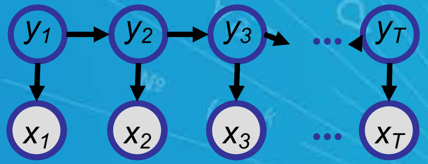
Probabilistic Graphical Models

Case Studies: HMM and CRF

Eric Xing

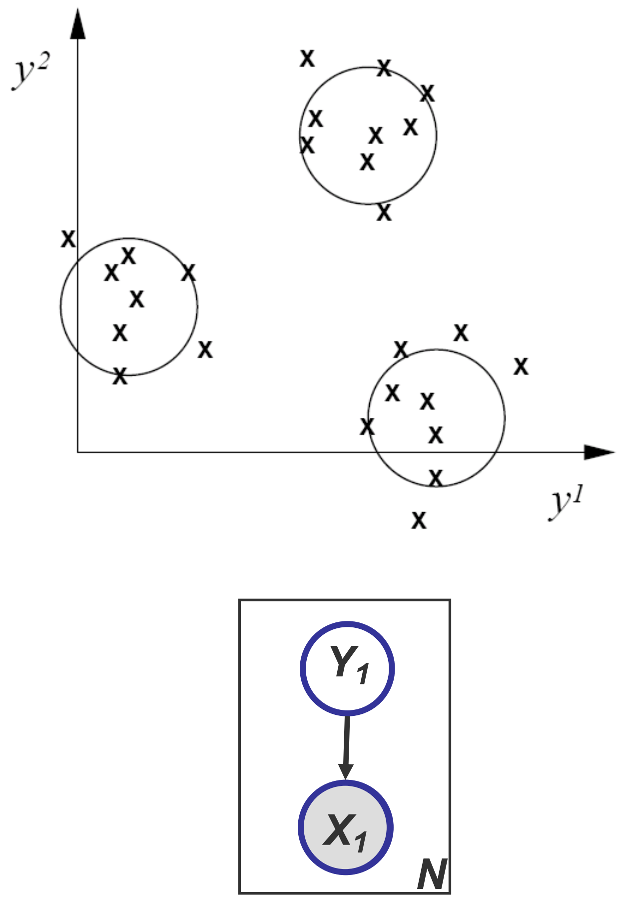
Lecture 6, February 3, 2020

Reading: see class homepage

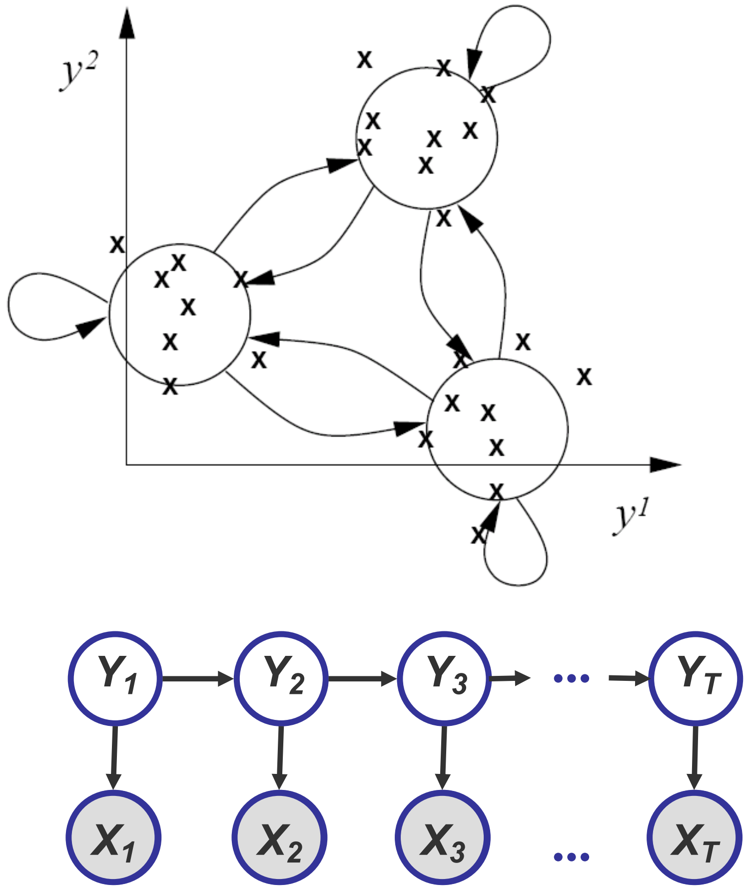


Hidden Markov Model:
from static to dynamic mixture models

Static mixture



Dynamic mixture



Example

- Speech recognition

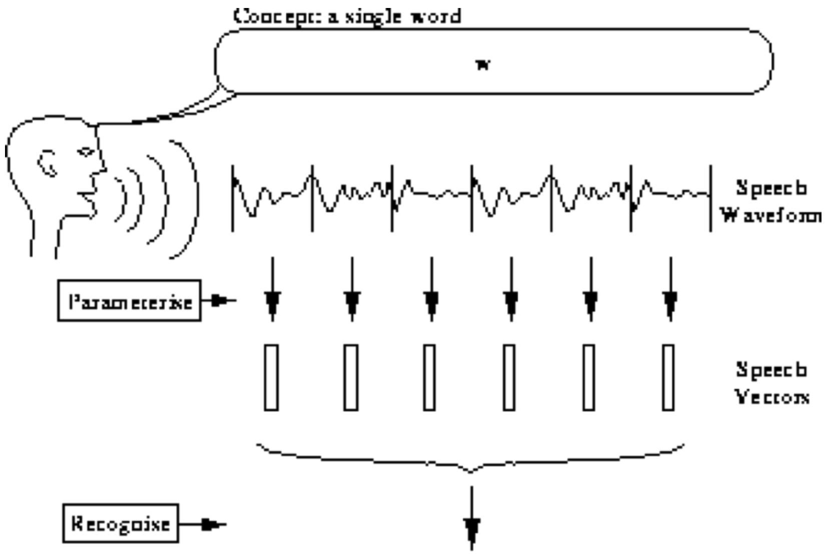
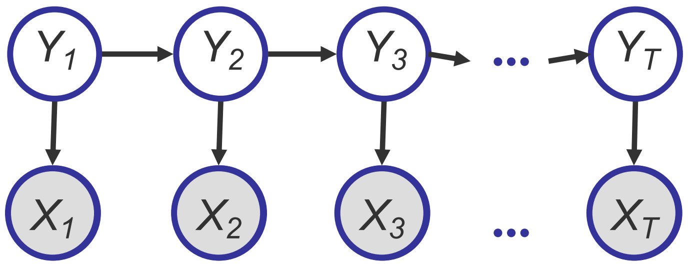


Fig. 1.2 Isolated Word Problem



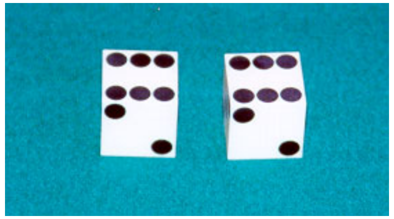
Applications of HMMs

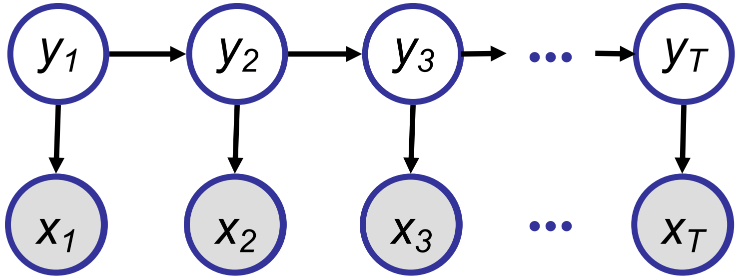
- Some early applications of HMMs
  - finance, but we never saw them
  - speech recognition
  - modelling ion channels

- In the mid-late 1980s HMMs entered genetics and molecular biology, and they are now firmly entrenched.

- Some current applications of HMMs to biology
  - mapping chromosomes
  - aligning biological sequences
  - predicting sequence structure
  - inferring evolutionary relationships
  - finding genes in DNA sequence

Definition (of HMM)





- **Observation space**
  - Alphabetic set: $\mathrm{C} = \{c_1, c_2, \cdots, c_K\}$
  - Euclidean space: R^d
- **Index set of hidden states**

$$\mathrm{I} = \{1, 2, \cdots, M\}$$

- **Transition probabilities** between any two states

$$p(y_t^j = 1 \mid y_{t-1}^i = 1) = a_{i,j},$$

or

$$p(y_t \mid y_{t-1}^i = 1) \sim \mathrm{Multinomial}(a_{i,1}, a_{i,1}, \ldots, a_{i,M}), \forall i \in \mathrm{I}.$$

- **Start probabilities**

$$p(y_1) \sim \mathrm{Multinomial}(\pi_1, \pi_2, \ldots, \pi_M).$$

- **Emission probabilities** associated with each state

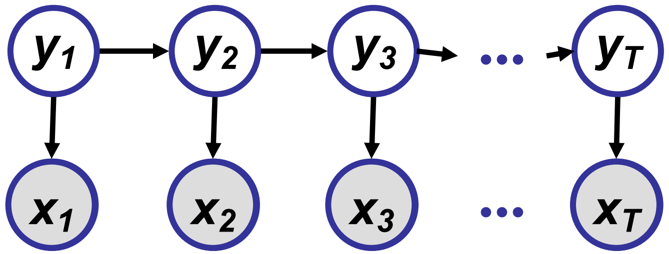
$$p(x_t \mid y_t^i = 1) \sim \mathrm{Multinomial}(b_{i,1}, b_{i,1}, \ldots, b_{i,K}), \forall i \in \mathrm{I}.$$

or in general:

$$p(x_t \mid y_t^i = 1) \sim \mathrm{f}(\cdot \mid \theta_i), \forall i \in \mathrm{I}.$$

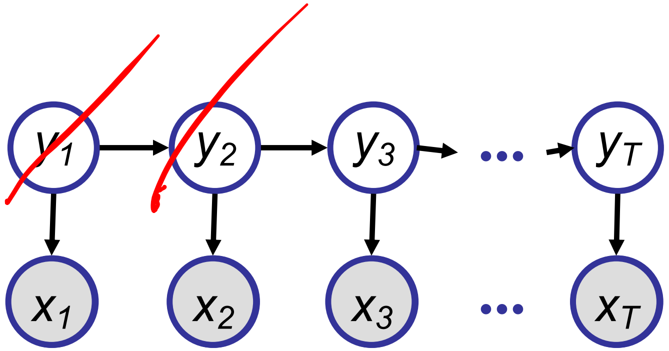
Probability of a parse

- Given a sequence $\mathbf{x} = x_1 \ldots\ldots x_T$
  and a parse $\mathbf{y} = y_1, \ldots\ldots, y_T,$
- To find how likely is the parse:
  (given our HMM and the sequence)



$$p(\mathbf{x}, \mathbf{y}) = p(x_1 \ldots\ldots x_T, y_1, \ldots\ldots, y_T) \quad \text{(Joint probability)}$$
$$= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T)$$
$$= p(y_1)\, P(y_2 \mid y_1) \ldots p(y_T \mid y_{T-1}) \times p(x_1 \mid y_1)\, p(x_2 \mid y_2) \ldots p(x_T \mid y_T)$$
$$= p(y_1, \ldots\ldots, y_T)\, p(x_1 \ldots\ldots x_T \mid y_1, \ldots\ldots, y_T)$$

Variable Elimination on Hidden Markov Model



$$p(\mathbf{x}, \mathbf{y}) = p(x_1 \ldots\ldots x_T, y_1, \ldots\ldots, y_T)$$

$$= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T)$$

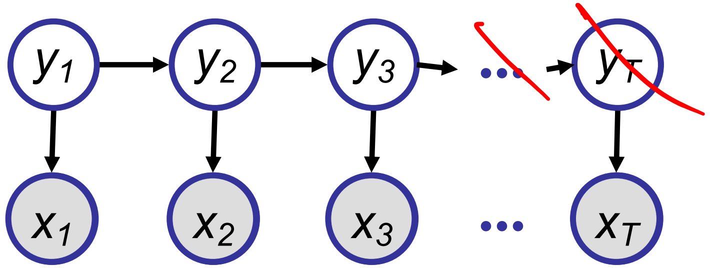
Conditional probability:

$$p(y_i|x_1,\ldots,x_T) = \sum_{y_1} \cdots \sum_{y_{i-1}} \sum_{y_{i+1}} \cdots \sum_{y_T} p(y_i,\ldots,y_T,x_1,\ldots,x_T)$$

$$= \sum_{y_1} \cdots \sum_{y_{i-1}} \sum_{y_{i+1}} \cdots \sum_{y_T} p(y_1)p(x_1|y_1)\ldots p(y_T|y_{T-1})p(x_T|y_T)$$

Variable Elimination on Hidden Markov Model

Conditional probability:



$$p(y_i|x_1,\dots,x_T) = \sum_{y_1}\cdots\sum_{y_{i-1}}\sum_{y_{i+1}}\cdots\sum_{y_T} p(y_i,\dots,y_T,x_1,\dots,x_T)$$

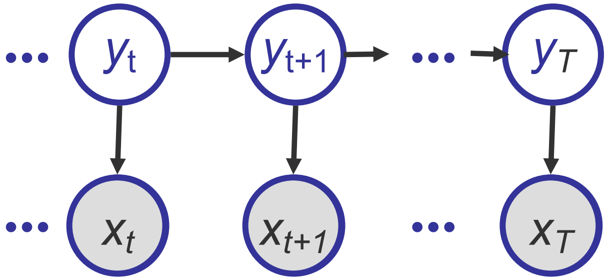
$$= \sum_{y_1}\cdots\sum_{y_{i-1}}\sum_{y_{i+1}}\cdots\sum_{y_T} p(y_1)p(x_1|y_1)\dots p(y_T|y_{T-1})p(x_T|y_T)$$

$$= \sum_{y_1} \cdots \sum_{y_{T-1}} \cdots \sum_{y_T} P(y_T|y_{T-1})P(x_T|y_T)$$

$$m'_{y_T}(x_T, y_{T-1}) \rightarrow P(x_T|y_{T-1})$$

$$= \sum_{y_1} \cdots \sum_{y_{T-2}} \cdots m_{y_T}(\;)\,P(y_{T-1}|y_{T-2})P(x_{T-1}|y_{T-1}) \rightarrow m'_{y_{T-1}}(\;)$$

The Forward Algorithm



- We want to calculate $P(\mathbf{x})$, the likelihood of $\mathbf{x}$, given the HMM
  - Sum over all possible ways of generating $\mathbf{x}$:

$$p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_N} \pi_{y_1} \prod_{t=2}^{T} a_{y_{t-1}, y_t} \prod_{t=1}^{T} p(x_t \mid y_t)$$

  - To avoid summing over an exponential number of paths $\mathbf{y}$, define

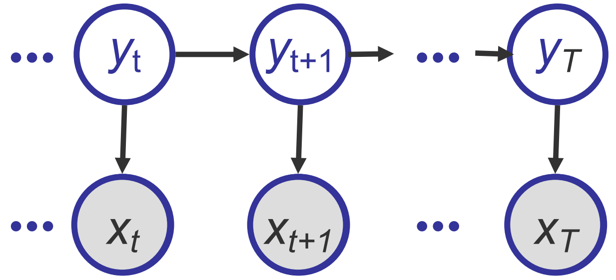
$$\alpha(y_t^k = 1) = \alpha_t^k \overset{\text{def}}{=} P(x_1, \ldots, x_t, y_t^k = 1)$$ (the forward probability)

  - The recursion:

$$\alpha_t^k = p(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Backward Algorithm



- We want to compute $P(y_t^k = 1 \mid \mathbf{x})$,

  the posterior probability distribution on the t th position, given $\mathbf{x}$

  - We start by computing

$$P(y_t^k = 1, \mathbf{x}) = P(x_1, \ldots, x_t, y_t^k = 1, x_{t+1}, \ldots, x_T)$$
$$= P(x_1, \ldots, x_t, y_t^k = 1) P(x_{t+1}, \ldots, x_T \mid x_1, \ldots, x_t, y_t^k = 1)$$
$$= P(x_1 \ldots x_t, y_t^k = 1) P(x_{t+1} \ldots x_T \mid y_t^k = 1)$$

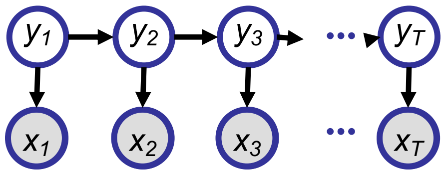
Forward, α_t^k Backward, $\beta_t^k = P(x_{t+1}, \ldots, x_T \mid y_t^k = 1)$

  - The recursion:

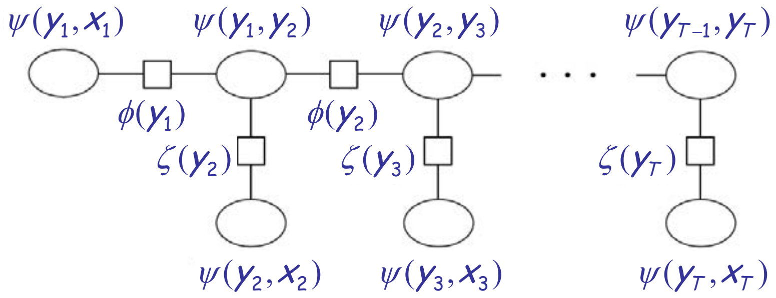
$$\beta_t^k = \sum_i a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

The junction tree algorithm: message passing for HMM

- A junction tree for the HMM



$\Rightarrow$

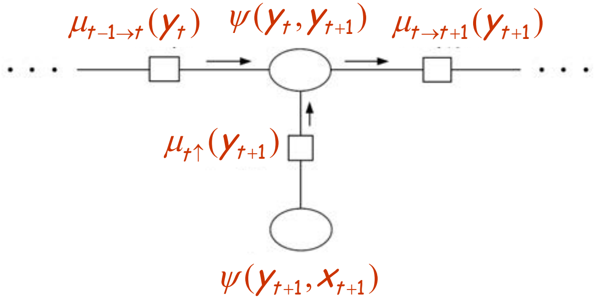


- Rightward pass

$$\mu_{t\to t+1}(y_{t+1}) = \sum_{y_t} \psi(y_t, y_{t+1})\mu_{t-1\to t}(y_t)\mu_{t\uparrow}(y_{t+1})$$

$$= \sum_{y_t} p(y_{t+1} \mid y_t)\mu_{t-1\to t}(y_t)p(x_{t+1} \mid y_{t+1})$$

$$= p(x_{t+1} \mid y_{t+1})\sum_{y_t} a_{y_t, y_{t+1}}\mu_{t-1\to t}(y_t)$$

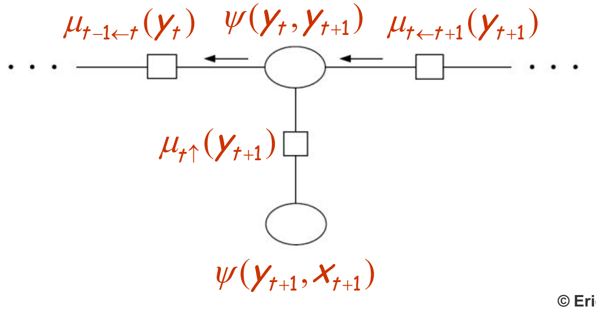


  - This is exactly the *forward algorithm*!

- Leftward pass …

$$\mu_{t-1\leftarrow t}(y_t) = \sum_{y_{t+1}} \psi(y_t, y_{t+1})\mu_{t\leftarrow t+1}(y_{t+1})\mu_{t\uparrow}(y_{t+1})$$

$$= \sum_{y_{t+1}} p(y_{t+1} \mid y_t)\mu_{t\leftarrow t+1}(y_{t+1})p(x_{t+1} \mid y_{t+1})$$



  - This is exactly the *backward algorithm*!

Summary

- Forward algorithm

$$\alpha_t^k \overset{\text{def}}{=} \mu_{t-1\to t}(k) = P(x_1, \ldots, x_{t-1}, x_t, y_t^k = 1)$$

$$\alpha_t^k = p(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

- Backward algorithm

$$\beta_t^k = \sum_i a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

$$\beta_t^k \overset{\text{def}}{=} \mu_{t-1\leftarrow t}(k) = P(x_{t+1}, \ldots, x_T \mid y_t^k = 1)$$

$$\gamma_t^i \overset{\text{def}}{=} p(y_t^i = 1 \mid x_{1:T}) \propto \alpha_t^i \beta_t^i = \sum_j \xi_t^{i,j}$$

$$\xi_t^{i,j} \overset{\text{def}}{=} p(y_t^i = 1, y_{t+1}^j = 1, x_{1:T})$$

$$\propto \mu_{t-1\to t}(y_t^i = 1) \mu_{t\leftarrow t+1}(y_{t+1}^j = 1) p(x_{t+1} \mid y_{t+1}) p(y_{t+1} \mid y_t)$$

$$\xi_t^{i,j} = \alpha_t^i \beta_{t+1}^j a_{i,j} p(x_{t+1} \mid y_{t+1}^i = 1)$$

The matrix-vector form:

$$B_t(i) \overset{\text{def}}{=} p(x_t \mid y_t^i = 1)$$

$$A(i,j) \overset{\text{def}}{=} p(y_{t+1}^j = 1 \mid y_t^i = 1)$$

$$\alpha_t = \left(A^T \alpha_{t-1}\right) .* B_t$$

$$\beta_t = A\left(\beta_{t+1} .* B_{t+1}\right)$$

$$\xi_t = \left(\alpha_t \left(\beta_{t+1} .* B_{t+1}\right)^T\right) .* A$$

$$\gamma_t = \alpha_t .* \beta_t$$

Posterior decoding

- We can now calculate

$$P(y_t^k = 1 \mid \mathbf{x}) = \frac{P(y_t^k = 1, \mathbf{x})}{P(\mathbf{x})} = \frac{\alpha_t^k \beta_t^k}{P(\mathbf{x})}$$

- Then, we can ask
  - What is the most likely state at position t of sequence $\mathbf{x}$:

$$k_t^* = \arg\max_k P(y_t^k = 1 \mid \mathbf{x})$$

  - Note that this is an MPA of a single hidden state, what if we want to a MPA of a whole hidden state sequence?
  - Posterior Decoding: $\left\{ y_t^{k_t^*} = 1 : t = 1 \cdots T \right\}$
  - This is different from MPA of a whole sequence f hidden states
  - This can be understood as *bit error rate* vs. *word error rate*

Example:
MPA of *X* ?
MPA of *(X, Y)* ?

x	y	$P(x,y)$
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

Viterbi decoding

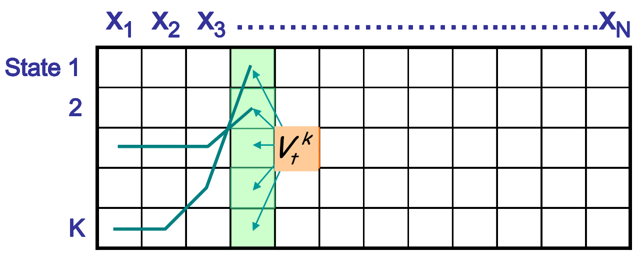
- GIVEN $\mathbf{x} = x_1, \ldots, x_T$, we want to find $\mathbf{y} = y_1, \ldots, y_T$, such that $P(\mathbf{y}|\mathbf{x})$ is maximized:

$$\mathbf{y}^* = \text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\pi} P(\mathbf{y},\mathbf{x})$$

- Let $V_t^k = \max_{\{y_1,\ldots,y_{t-1}\}} P(x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1},x_t,y_t^k=1)$

  = Probability of most likely sequence of states ending at state $y_t = k$

- The recursion: $V_t^k = p(x_t \mid y_t^k = 1) \max_i a_{i,k} V_{t-1}^i$



- Underflows are a significant problem

$$p(x_1,\ldots,x_t,y_1,\ldots,y_t) = \pi_{y_1} a_{y_1,y_2} \cdots a_{y_{t-1},y_t} b_{y_1,x_1} \cdots b_{y_t,x_t}$$

  - These numbers become extremely small – underflow
  - Solution: Take the logs of all values: $V_t^k = \log p(x_t \mid y_t^k = 1) + \max_i \left(\log(a_{i,k}) + V_{t-1}^i\right)$

The Viterbi Algorithm – derivation

- Define the viterbi probability:

$$V_{t+1}^k = \max_{\{y_1,\dots y_t\}} P(x_1,\dots,x_t, y_1,\dots,y_t, x_{t+1}, y_{t+1}^k = 1)$$

$$= \max_{\{y_1,\dots y_t\}} P(x_{t+1}, y_{t+1}^k = 1 \mid x_1,\dots,x_t, y_1,\dots,y_t) P(x_1,\dots,x_t, y_1,\dots,y_t)$$

$$= \max_{\{y_1,\dots y_t\}} P(x_{t+1}, y_{t+1}^k = 1 \mid y_t) P(x_1,\dots,x_{t-1}, y_1,\dots,y_{t-1}, x_t, y_t)$$

$$= \max_i P(x_{t+1}, y_{t+1}^k = 1 \mid y_t^i = 1) \max_{\{y_1,\dots y_{t-1}\}} P(x_1,\dots,x_{t-1}, y_1,\dots,y_{t-1}, x_t, y_t^i = 1)$$

$$= \max_i P(x_{t+1}, \mid y_{t+1}^k = 1) a_{i,k} V_t^i$$

$$= P(x_{t+1}, \mid y_{t+1}^k = 1) \max_i a_{i,k} V_t^i$$

Computational Complexity and implementation details

- What is the running time, and space required, for Forward, and Backward?

$$\alpha_t^k = p(x_t \mid y_t^k = 1)\sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1)\beta_{t+1}^i$$

$$V_t^k = p(x_t \mid y_t^k = 1) \max_i a_{i,k} V_{t-1}^i$$

Time: $O(K^2 N)$; Space: $O(KN)$.

- Useful implementation technique to avoid underflows
  - Viterbi: sum of logs
  - Forward/Backward: rescaling at each position by multiplying by a constant

Learning HMM: two scenarios

- Supervised learning: estimation when the "right answer" is known
  - Examples:

    GIVEN: a genomic region $x = x_1 \ldots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands

    GIVEN: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

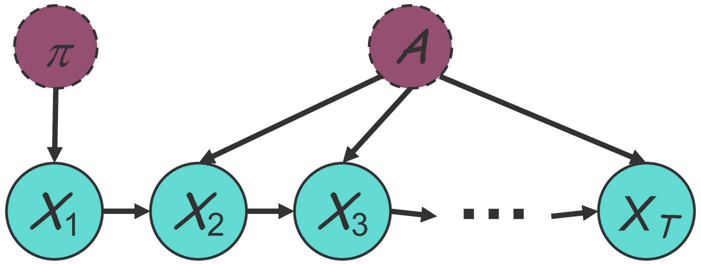
- Unsupervised learning: estimation when the "right answer" is unknown
  - Examples:

    GIVEN: the porcupine genome; we don't know how frequent are the CpG islands there, neither do we know their composition

    GIVEN: 10,000 rolls of the casino player, but we don't see when he changes dice

- QUESTION: Update the parameters θ of the model to maximize $P(x|\theta)$ --- Maximal likelihood (ML) estimation

Parameter sharing



- Consider a time-invariant (stationary) 1st-order Markov model
  - Initial state probability vector: $\pi_k \stackrel{\text{def}}{=} p(X_1^k = 1)$
  - State transition probability matrix: $A_{ij} \stackrel{\text{def}}{=} p(X_t^j = 1 \mid X_{t-1}^i = 1)$
- The joint: $$p(X_{1:T} \mid \theta) = p(x_1 \mid \pi) \prod_{t=2}^{T} \prod_{t=2} p(X_t \mid X_{t-1})$$
- The log-likelihood: $$\ell(\theta; D) = \sum_n \log p(x_{n,1} \mid \pi) + \sum_n \sum_{t=2}^{T} \log p(x_{n,t} \mid x_{n,t-1}, A)$$
- Again, we optimize each parameter separately
  - π is a multinomial frequency vector, and we've seen it before
  - What about A?

Learning a Markov chain transition matrix

- A is a stochastic matrix: $\sum_j A_{ij} = 1$
- Each row of A is multinomial distribution.
- So MLE of A_{ij} is the fraction of transitions from i to j

$$A_{ij}^{ML} = \frac{\#(i \to j)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=2}^{T} x_{n,t-1}^{i} x_{n,t}^{j}}{\sum_n \sum_{t=2}^{T} x_{n,t-1}^{i}}$$

- Application:
  - if the states X_t represent words, this is called a *bigram language model*
- Sparse data problem:
  - If $i \to j$ did not occur in data, we will have A_{ij} =0, then any future sequence with word pair $i \to j$ will have zero probability.
  - A standard hack: *backoff smoothing* or *deleted interpolation*

$$\tilde{A}_{i\to\bullet} = \lambda \eta_t + (1-\lambda) A_{i\to\bullet}^{ML}$$

Supervised ML estimation for "Hidden" MM

- Given $x = x_1 \ldots x_N$ for which the true state path $y = y_1 \ldots y_N$ is known,
  - Define:

    A_{ij} = # times state transition $i \to j$ occurs in $\mathbf{y}$

    B_{ik} = # times state i in $\mathbf{y}$ emits k in $\mathbf{x}$

  - We can show that the maximum likelihood parameters θ are:

$$a_{ij}^{ML} = \frac{\#(i \to j)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=2}^{T} y_{n,t-1}^i y_{n,t}^j}{\sum_n \sum_{t=2}^{T} y_{n,t-1}^i} = \frac{A_{ij}}{\sum_{j'} A_{ij'}}$$

$$b_{ik}^{ML} = \frac{\#(i \to k)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=1}^{T} y_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T} y_{n,t}^i} = \frac{B_{ik}}{\sum_{k'} B_{ik'}}$$

  - What if x is continuous? We can treat $\{(x_{n,t}, y_{n,t}) : t = 1:T, n = 1:N\}$ as $N \times T$ observations of, e.g., a Gaussian, and apply learning rules for Gaussian …

Supervised ML estimation, ctd.

- Intuition:
  - When we know the underlying states, the best estimate of θ is the average frequency of transitions & emissions that occur in the training data
- Drawback:
  - Given little data, there may be overfitting:
    - $P(x|\theta)$ is maximized, but θ is unreasonable: 0 probabilities – VERY BAD
- Example:
  - Given 10 casino rolls, we observe

    x = 2, 1, 5, 6, 1, 2, 3, 6, 2, 3

    y = F, F, F, F, F, F, F, F, F, F
  - Then: $a_{FF} = 1$; $a_{FL} = 0$

    $b_{F1} = b_{F3} = .2$;

    $b_{F2} = .3$; $b_{F4} = 0$; $b_{F5} = b_{F6} = .1$

Pseudocounts

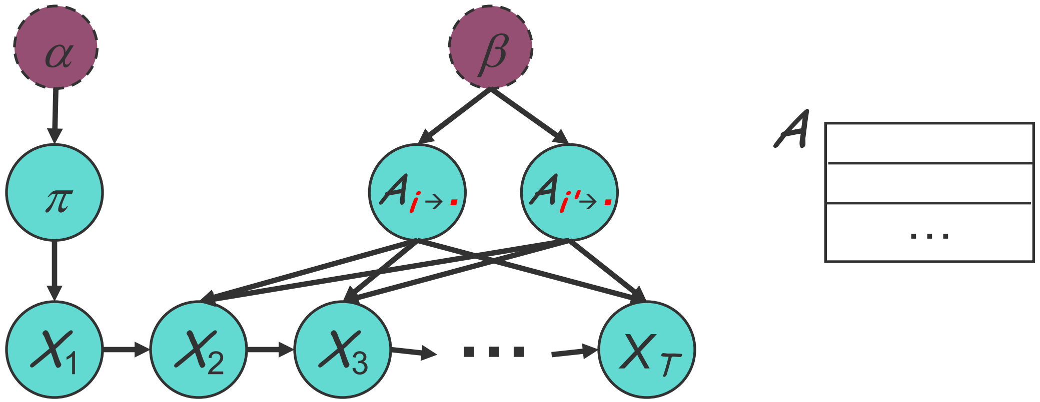
- Solution for small training sets:
  - Add pseudocounts

    A_{ij} = # times state transition $i \rightarrow j$ occurs in $\mathbf{y}$ + R_{ij}
    B_{ik} = # times state i in $\mathbf{y}$ emits k in $\mathbf{x}$ + S_{ik}

  - R_{ij}, S_{ij} are pseudocounts representing our prior belief
  - Total pseudocounts: $R_i = \Sigma_j R_{ij}$, $S_i = \Sigma_k S_{ik}$,
    - --- "strength" of prior belief,
    - --- total number of imaginary instances in the prior
- Larger total pseudocounts $\Rightarrow$ strong prior belief
- Small total pseudocounts: just to avoid 0 probabilities --- smoothing
- This is equivalent to Bayesian est. under a uniform prior with "parameter strength" equals to the pseudocounts

Bayesian language model

- Global and local parameter independence



- The posterior of $A_{i\to\cdot}$ and $A_{i'\to\cdot}$ is factorized despite v-structure on X_t, because X_{t-1} acts like a multiplexer
- Assign a Dirichlet prior β_i to each row of the transition matrix:

$$A_{ij}^{Bayes} \overset{\text{def}}{=} p(j \mid i, D, \beta_i) = \frac{\#(i \to j) + \beta_{i,k}}{\#(i \to \bullet) + |\beta_i|} = \lambda_i \beta'_{i,k} + (1 - \lambda_i) A_{ij}^{ML}, \text{ where } \lambda_i = \frac{|\beta_i|}{|\beta_i| + \#(i \to \bullet)}$$

  - We could consider more realistic priors, e.g., mixtures of Dirichlets to account for types of words (adjectives, verbs, etc.)

Example: HMM

- Supervised learning: estimation when the "right answer" is known
  - Examples:

    GIVEN: a genomic region $x = x_1 \dots x_{1,000,000}$ where we have good (experimental) annotations of the CpG islands

    GIVEN: the casino player allows us to observe him one evening, as he changes dice and produces 10,000 rolls

- Unsupervised learning: estimation when the "right answer" is unknown
  - Examples:

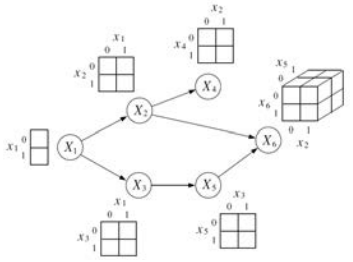
    GIVEN: the porcupine genome; we don't know how frequent are the CpG islands there, neither do we know their composition

    GIVEN: 10,000 rolls of the casino player, but we don't see when he changes dice

- QUESTION: Update the parameters θ of the model to maximize $P(x|\theta)$ --- Maximal likelihood (ML) estimation

Learning HMM: two scenarios

- Supervised learning: if only we knew the true state path then ML parameter estimation would be trivial
  - E.g., recall that for complete observed tabular BN:



$$\theta_{ijk}^{ML} = \frac{n_{ijk}}{\sum_{i,j',k} n_{ij'k}}$$

$$a_{ij}^{ML} = \frac{\#(i \to j)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=2}^T y_{n,t-1}^i y_{n,t}^j}{\sum_n \sum_{t=2}^T y_{n,t-1}^i}$$

$$b_{ik}^{ML} = \frac{\#(i \to k)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=1}^T y_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^T y_{n,t}^i}$$

  - What if y is continuous? We can treat $\{(x_{n,t}, y_{n,t}) : t = 1:T, n = 1:N\}$ as $N \times T$ observations of, e.g., a GLIM, and apply learning rules for GLIM …

- Unsupervised learning: when the true state path is unknown, we can fill in the missing values using inference recursions.
  - The Baum Welch algorithm (i.e., EM)
    - Guaranteed to increase the log likelihood of the model after each iteration
    - Converges to local optimum, depending on initial conditions

The Baum Welch algorithm

- The complete log likelihood

$$\ell_c(\theta; \mathbf{x}, \mathbf{y}) = \log p(\mathbf{x}, \mathbf{y}) = \log \prod_n \left(p(y_{n,1}) \prod_{t=2}^{T} p(y_{n,t} \mid y_{n,t-1}) \prod_{t=1}^{T} p(x_{n,t} \mid x_{n,t}) \right)$$

- The expected complete log likelihood

$$\langle \ell_c(\theta; \mathbf{x}, \mathbf{y}) \rangle = \sum_n \left(\langle y_{n,1}^i \rangle_{p(y_{n,1}|\mathbf{x}_n)} \log \pi_i \right) + \sum_n \sum_{t=2}^{T} \left(\langle y_{n,t-1}^i y_{n,t}^j \rangle_{p(y_{n,t-1}, y_{n,t}|\mathbf{x}_n)} \log a_{i,j} \right) + \sum_n \sum_{t=1}^{T} \left(x_{n,t}^k \langle y_{n,t}^i \rangle_{p(y_{n,t}|\mathbf{x}_n)} \log b_{i,k} \right)$$

- EM
  - The **E** step

$$\gamma_{n,t}^i = \langle y_{n,t}^i \rangle = p(y_{n,t}^i = 1 \mid \mathbf{x}_n)$$

$$\xi_{n,t}^{i,j} = \langle y_{n,t-1}^i y_{n,t}^j \rangle = p(y_{n,t-1}^i = 1, y_{n,t}^j = 1 \mid \mathbf{x}_n)$$

  - The **M** step ("symbolically" identical to MLE)

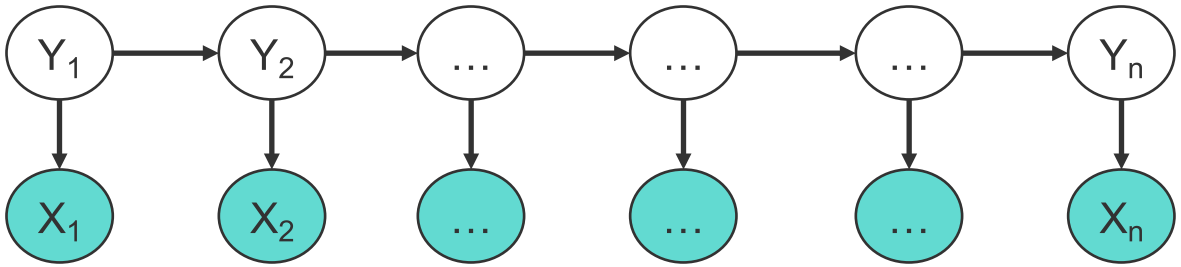
$$\pi_i^{ML} = \frac{\sum_n \gamma_{n,1}^i}{N} \qquad a_{ij}^{ML} = \frac{\sum_n \sum_{t=2}^{T} \xi_{n,t}^{i,j}}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i} \qquad b_{ik}^{ML} = \frac{\sum_n \sum_{t=1}^{T} \gamma_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i}$$

$$a_{ij}^{ML} = \frac{\#(i \to j)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=2}^{T} y_{n,t-1}^i y_{n,t}^j}{\sum_n \sum_{t=2}^{T} y_{n,t-1}^i}$$

$$b_{ik}^{ML} = \frac{\#(i \to k)}{\#(i \to \bullet)} = \frac{\sum_n \sum_{t=1}^{T} y_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T} y_{n,t}^i}$$

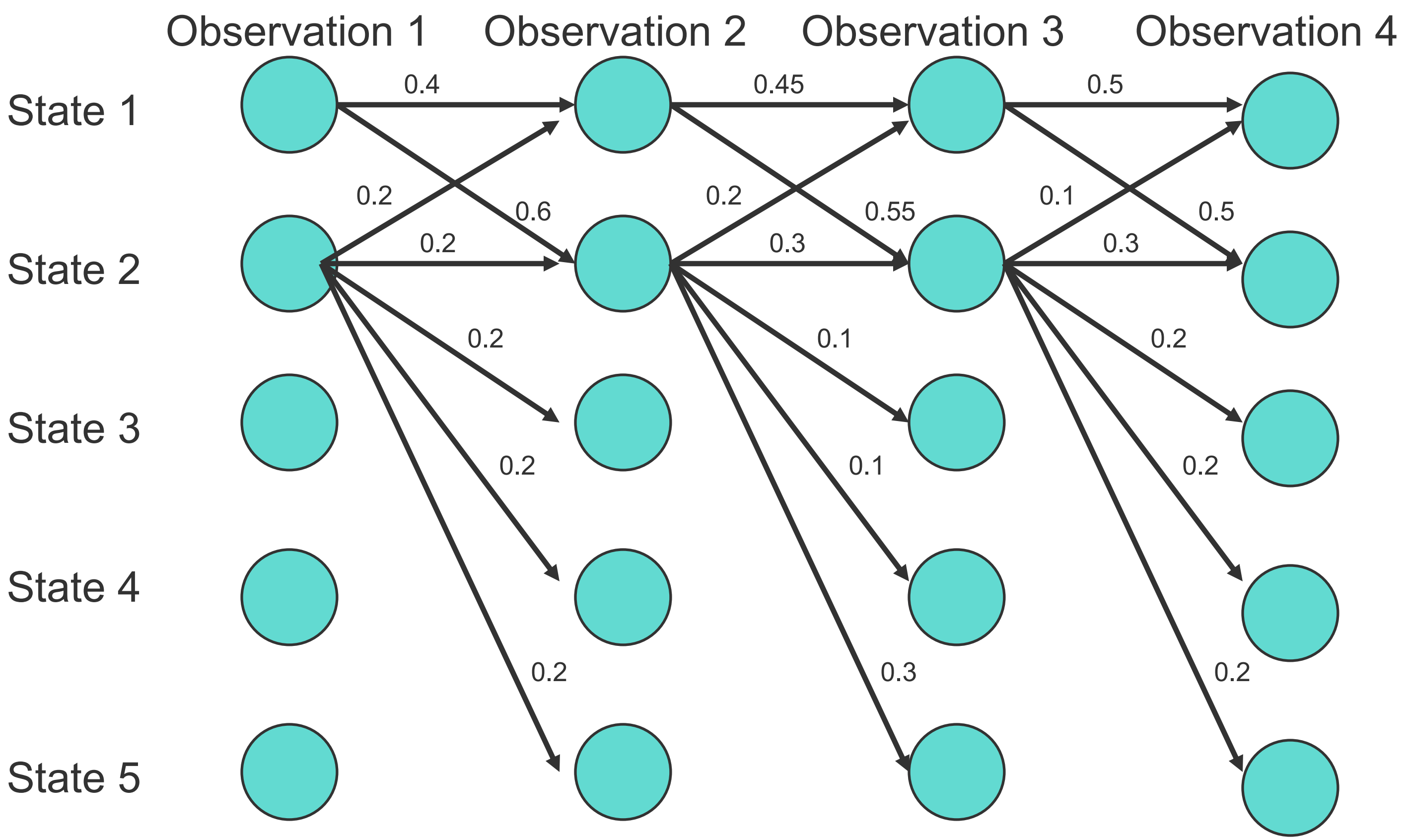
Conditional Random Fields

Shortcomings of Hidden Markov Model (1): locality of features



- HMM models capture dependences between each state and only its corresponding observation
  - NLP example: In a sentence segmentation task, each segmental state may depend not just on a single word (and the adjacent segmental stages), but also on the (non-local) features of the whole line such as line length, indentation, amount of white space, etc.
- Mismatch between learning objective function and prediction objective function
  - HMM learns a joint distribution of states and observations P(Y, X), but in a prediction task, we need the conditional probability P(Y|X)

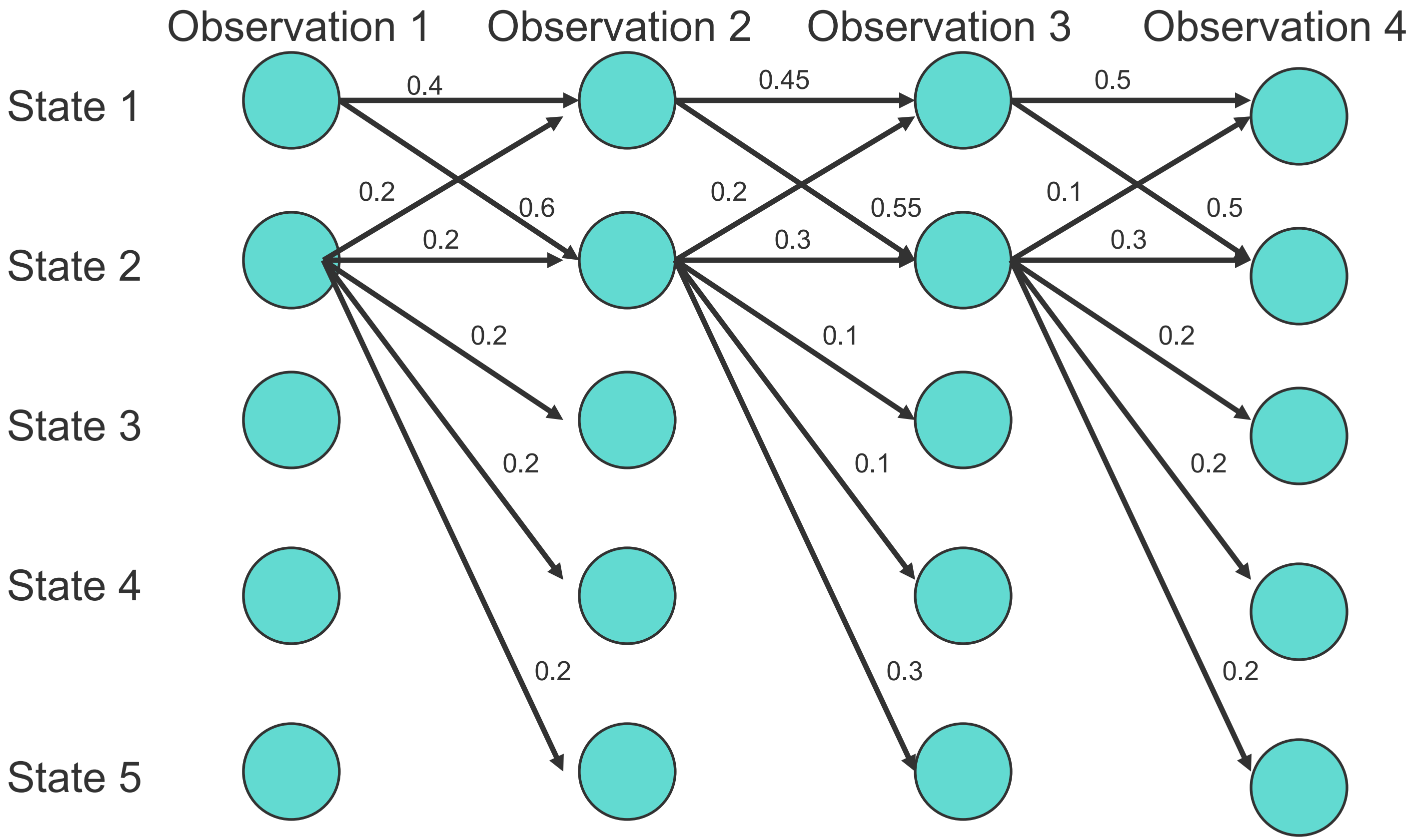
Shortcomings of HMM (2): the Label bias problem



What the local transition probabilities say:

- State 1 almost always prefers to go to state 2
- State 2 almost always prefer to stay in state 2

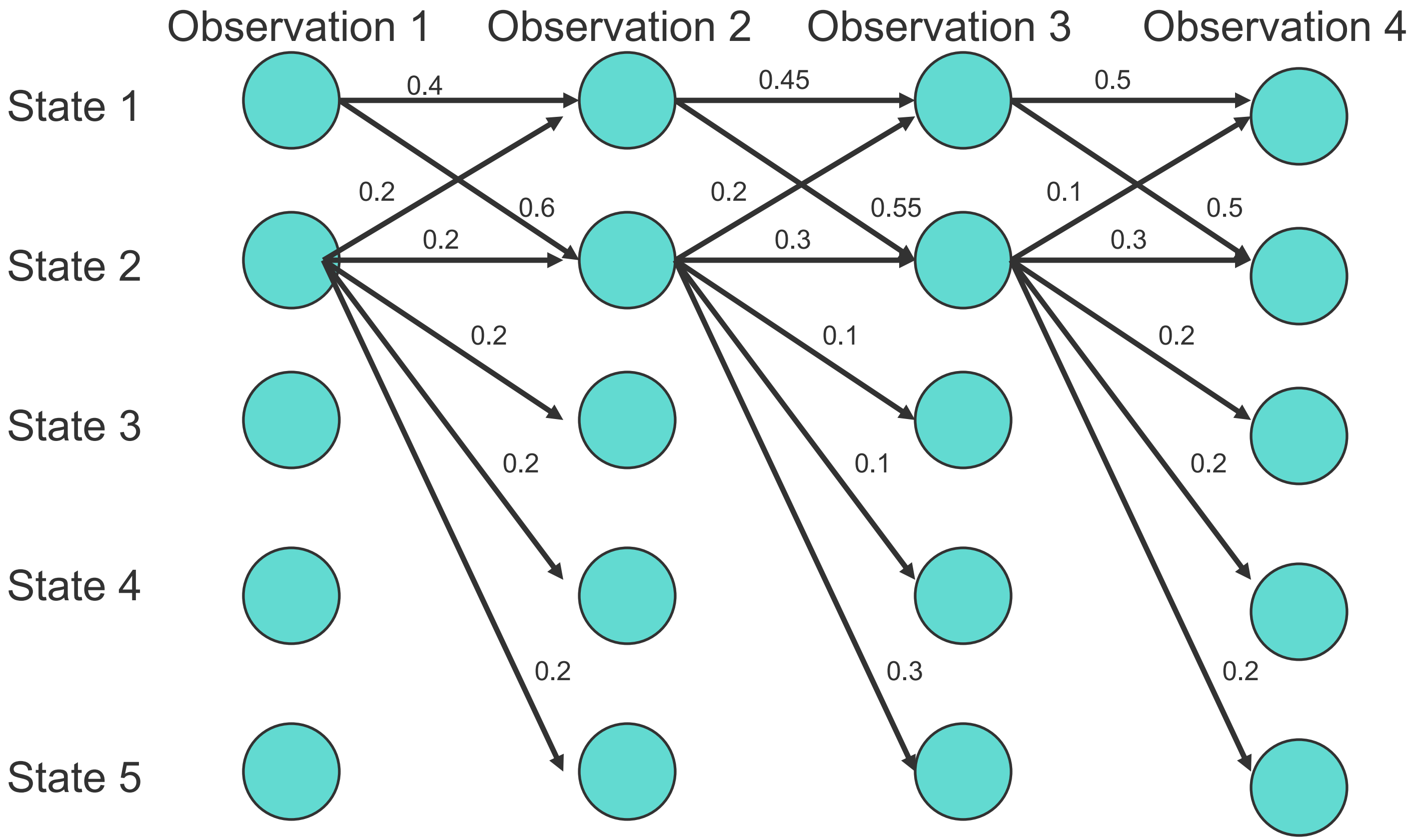
HMM: the Label bias problem



Probability of path 1-> 1-> 1-> 1:

- 0.4 x 0.45 x 0.5 = 0.09

HMM: the Label bias problem

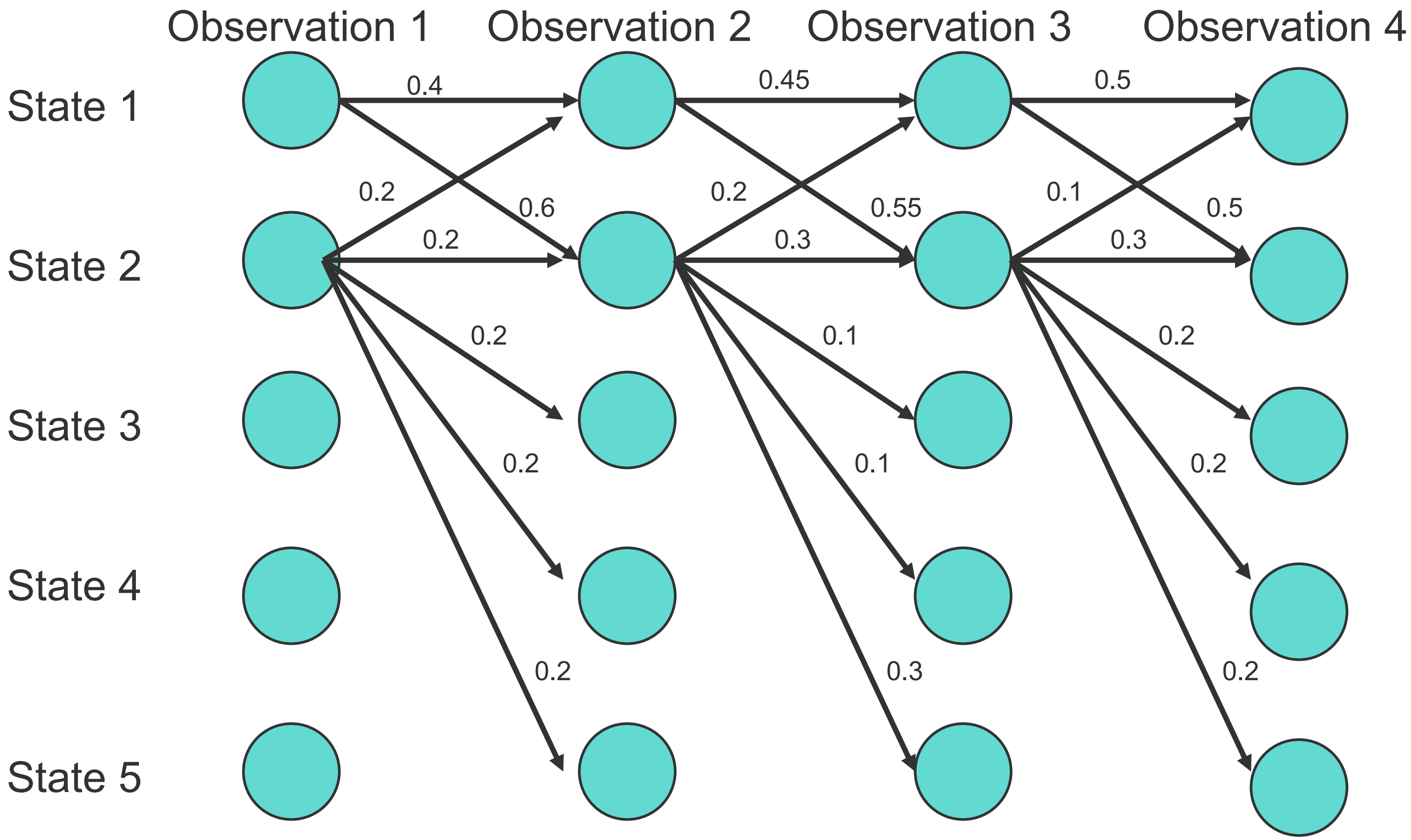


Probability of path 2->2->2->2 :

- 0.2 X 0.3 X 0.3 = 0.018

Other paths:
1-> 1-> 1-> 1: 0.09

HMM: the Label bias problem

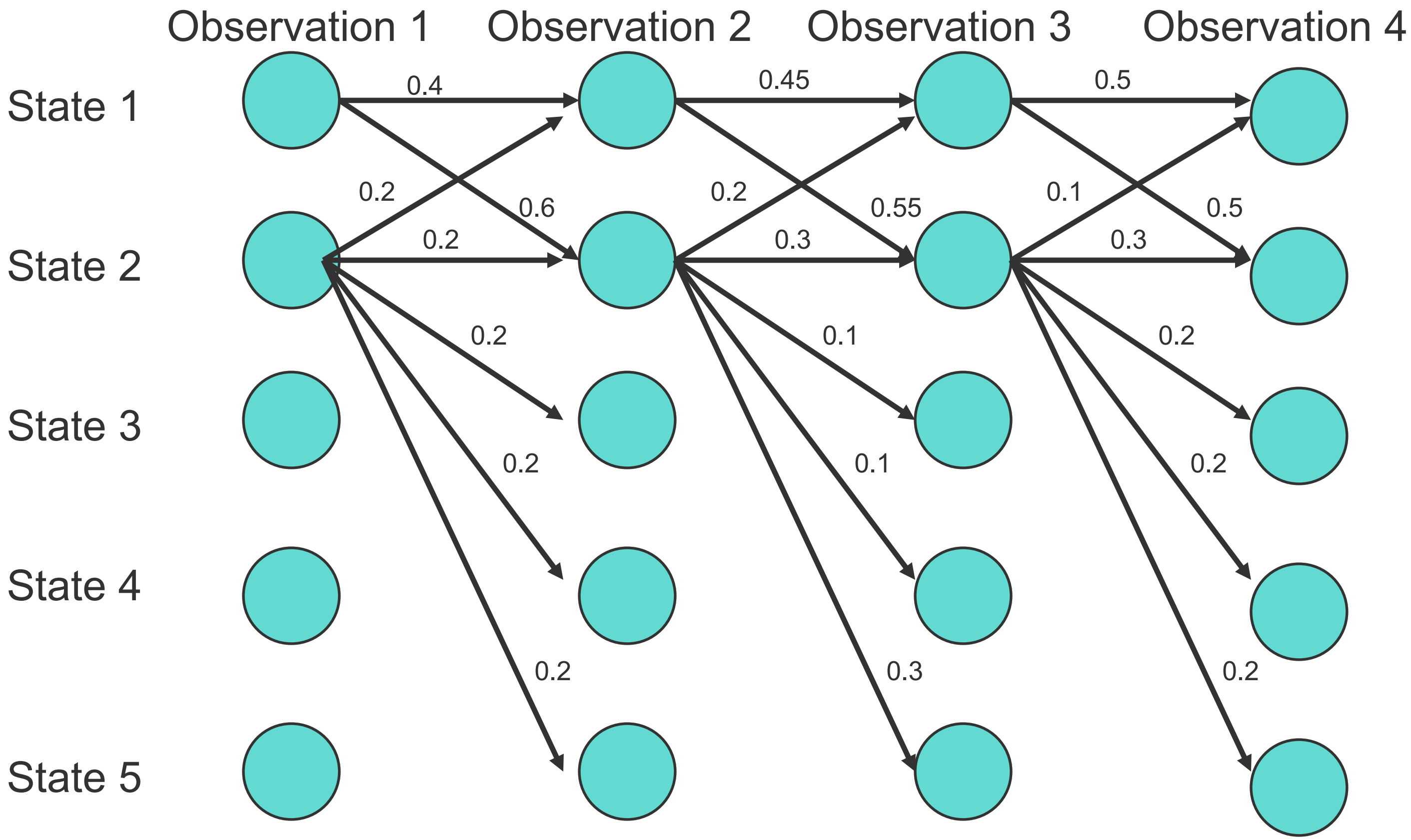


Probability of path 1->2->1->2:

- 0.6 X 0.2 X 0.5 = 0.06

Other paths:
1->1->1->1: 0.09
2->2->2->2: 0.018

HMM: the Label bias problem

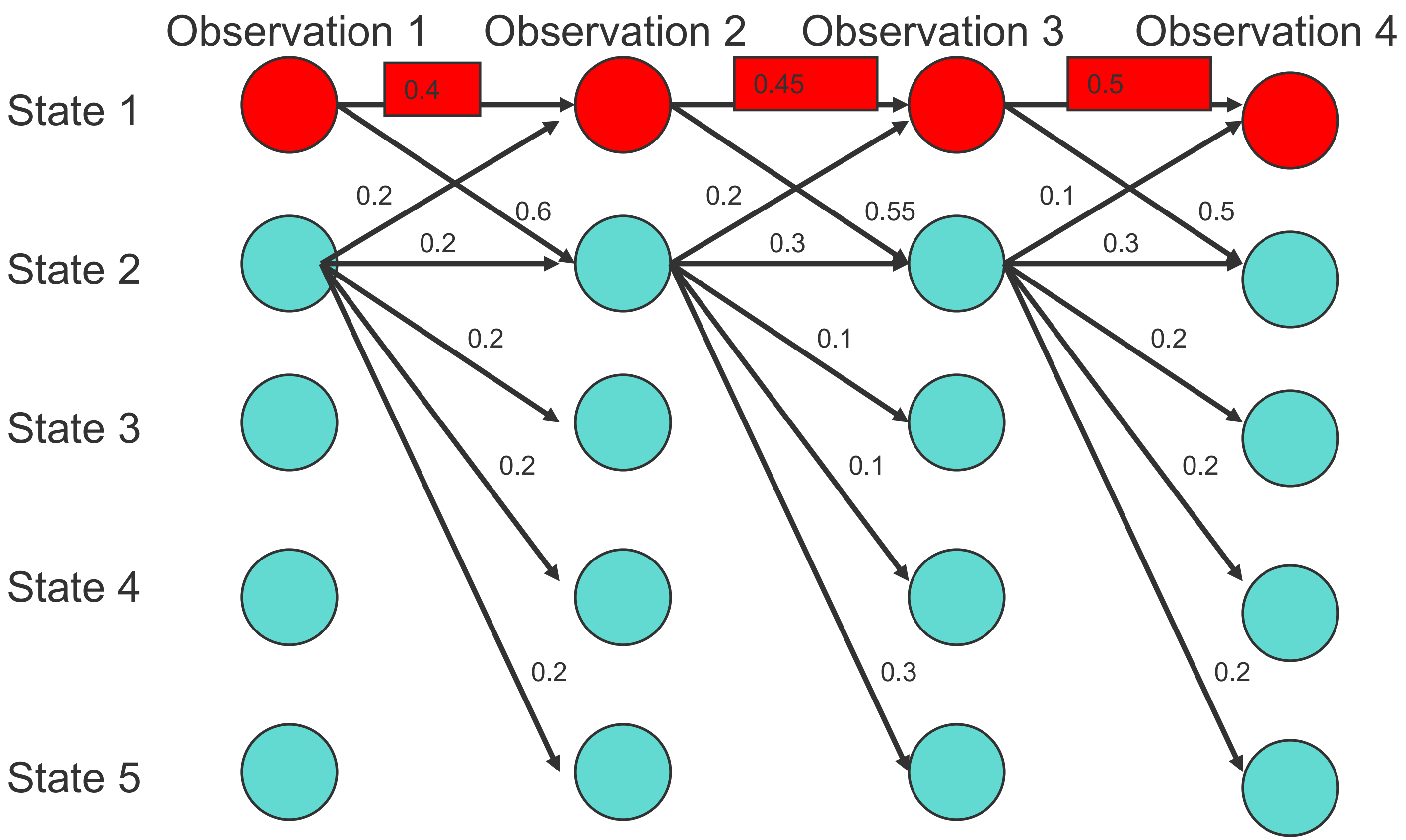


Probability of path 1->1->2->2:

- 0.4 X 0.55 X 0.3 = 0.066

Other paths:
1->1->1->1: 0.09
2->2->2->2: 0.018
1->2->1->2: 0.06

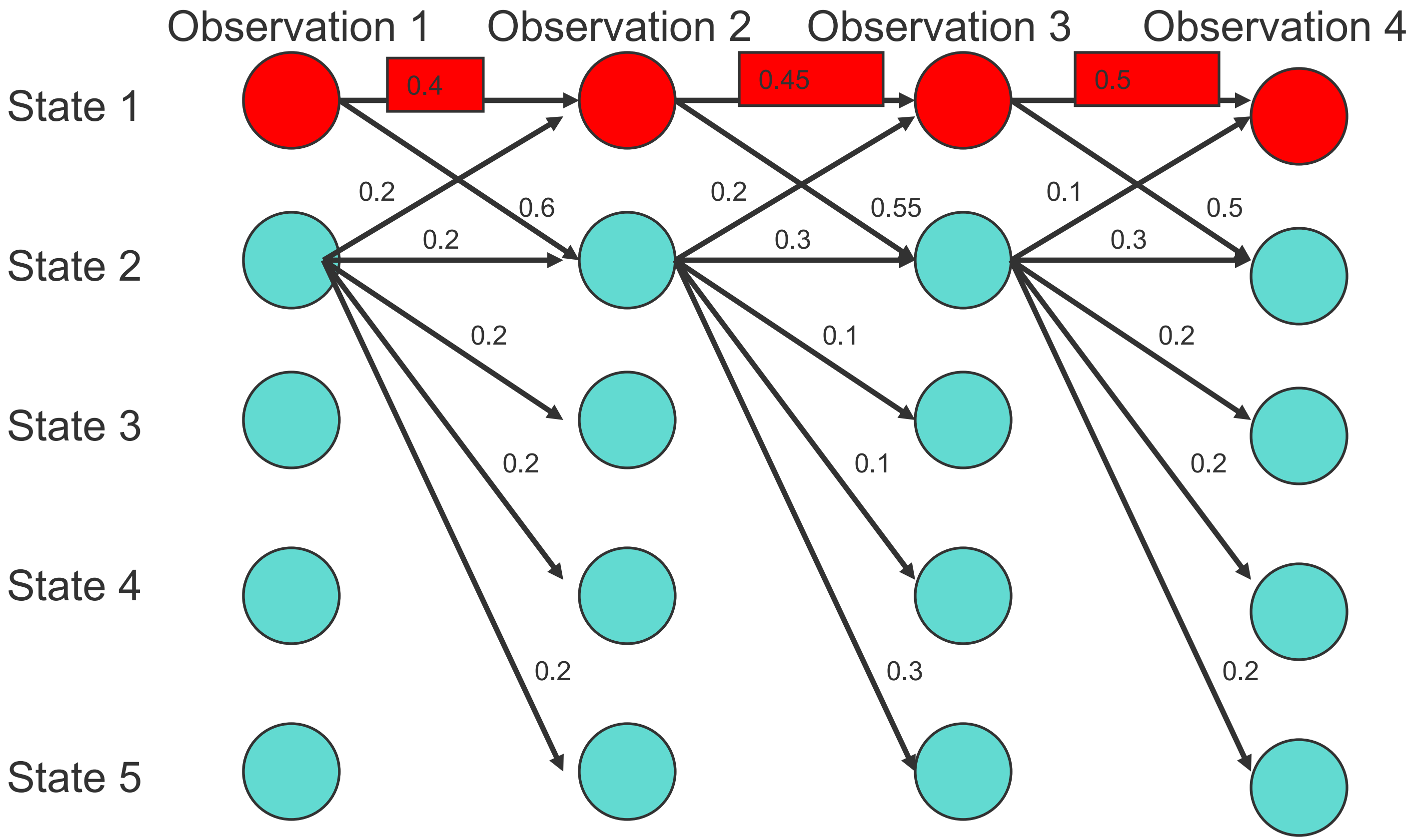
HMM: the Label bias problem



Most Likely Path: 1-> 1-> 1-> 1

- Although **locally** it seems state 1 wants to go to state 2 and state 2 wants to remain in state 2.
- **why?**

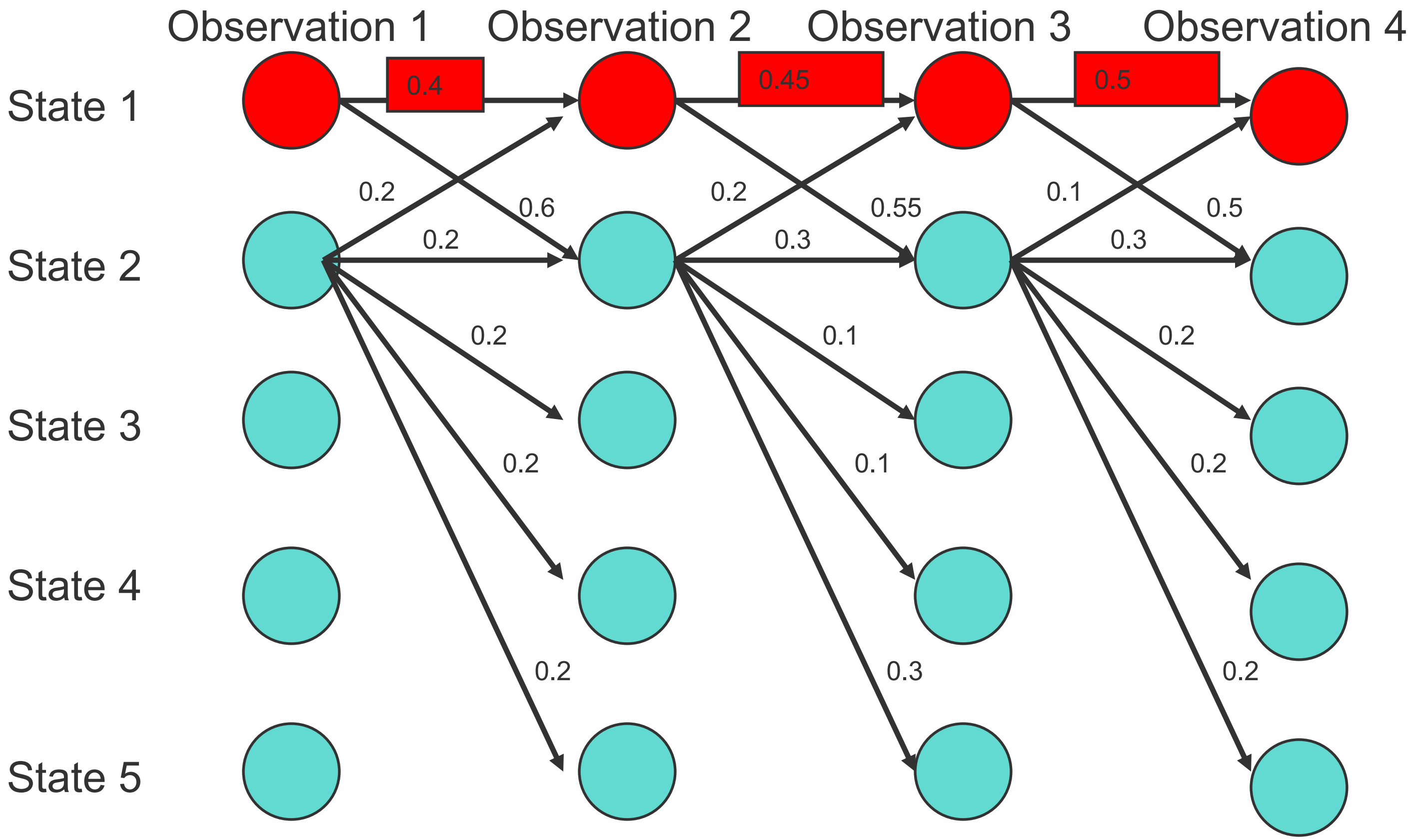
HMM: the Label bias problem



Most Likely Path: 1-> 1-> 1-> 1

- State 1 has only two transitions but state 2 has 5:
  - Average transition probability from state 2 is lower

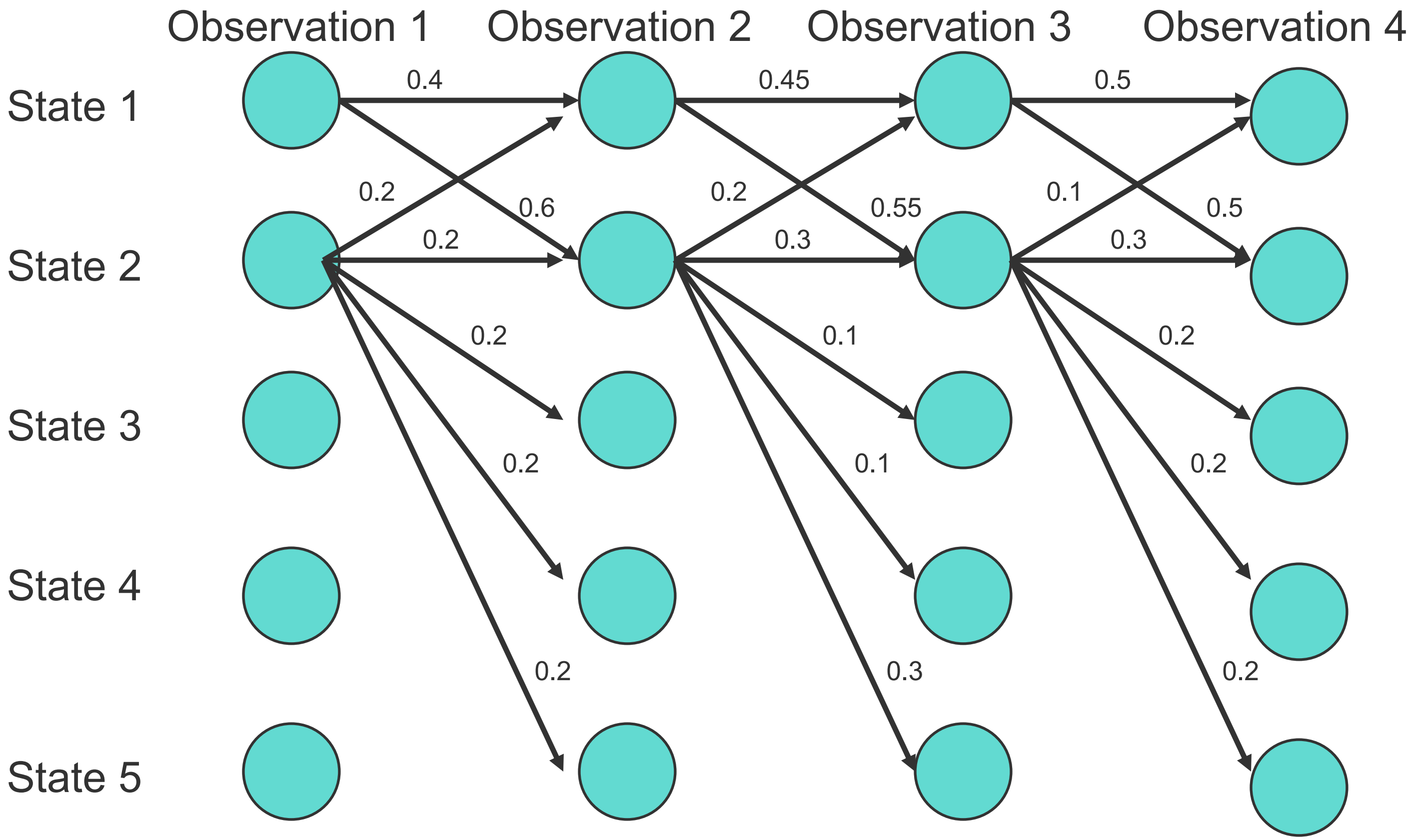
HMM: the Label bias problem



Label bias problem in HMM:

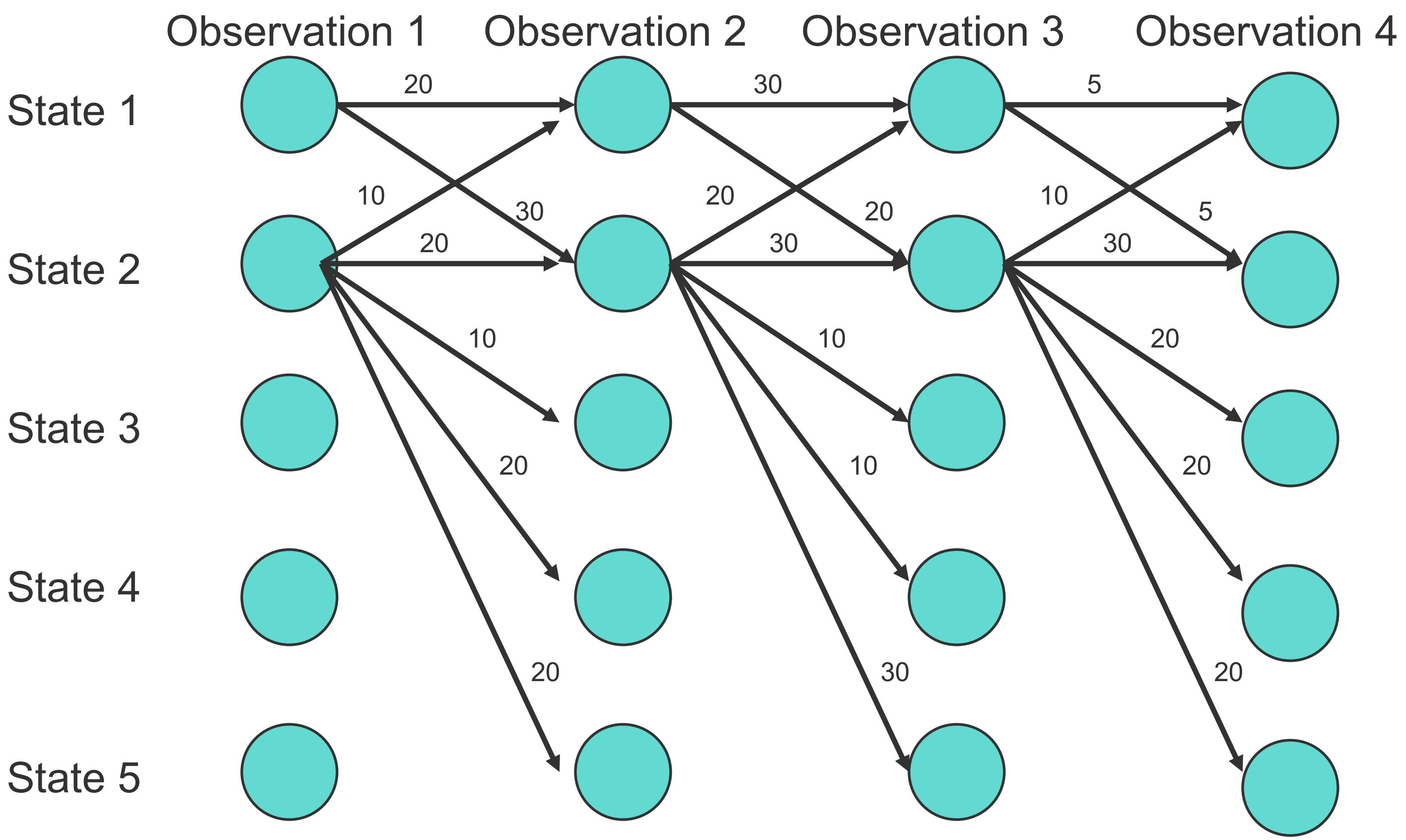
- Preference of states with lower number of transitions over others

Solution:
Do not normalize probabilities locally



From local probabilities ….

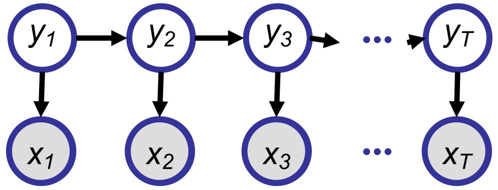
Solution:
Do not normalize probabilities locally



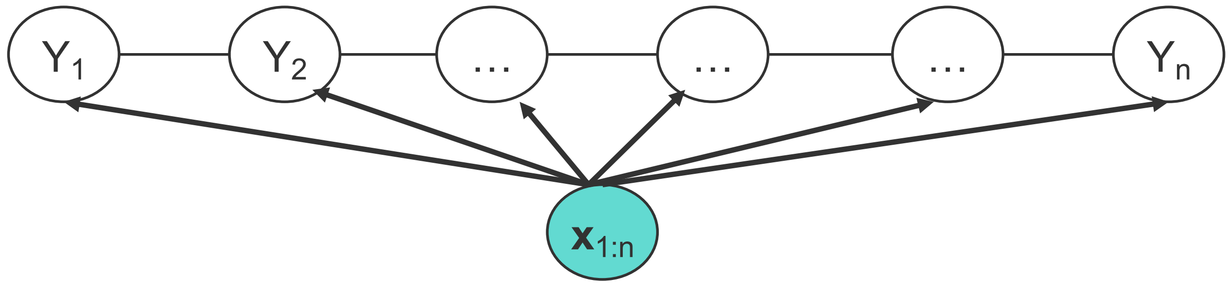
From local probabilities to local potentials

- States with lower transitions do not have an unfair advantage!

From HMM to CRF



$$P(X,Y) = \pi_{y_1} \prod_{t=2}^{T} a_{y_{t-1},y_t} \prod_{t=1}^{T} p(x_t \mid y_t)$$

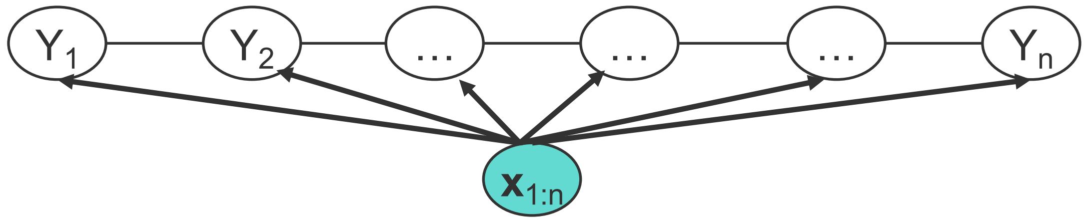


$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n})}\prod_{i=1}^{n}\phi(y_i, y_{i-1}, \mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n}, \mathbf{w})}\prod_{i=1}^{n}\exp(\mathbf{w}^T\mathbf{f}(y_i, y_{i-1}, \mathbf{x}_{1:n}))$$

- CRF is a partially directed model
  - Discriminative model, unlike HMM
  - Usage of global normalizer Z(**x**) overcomes the label bias problem of HMM
  - Models the dependence between each state and the entire observation sequence

Conditional Random Fields

- General parametric form:



$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x},\lambda,\mu)} \exp\Big(\sum_{i=1}^{n}\Big(\sum_{k}\lambda_k f_k(y_i, y_{i-1}, \mathbf{x}) + \sum_{l}\mu_l g_l(y_i, \mathbf{x})\Big)\Big)$$

$$= \frac{1}{Z(\mathbf{x},\lambda,\mu)} \exp\Big(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x}))\Big)$$

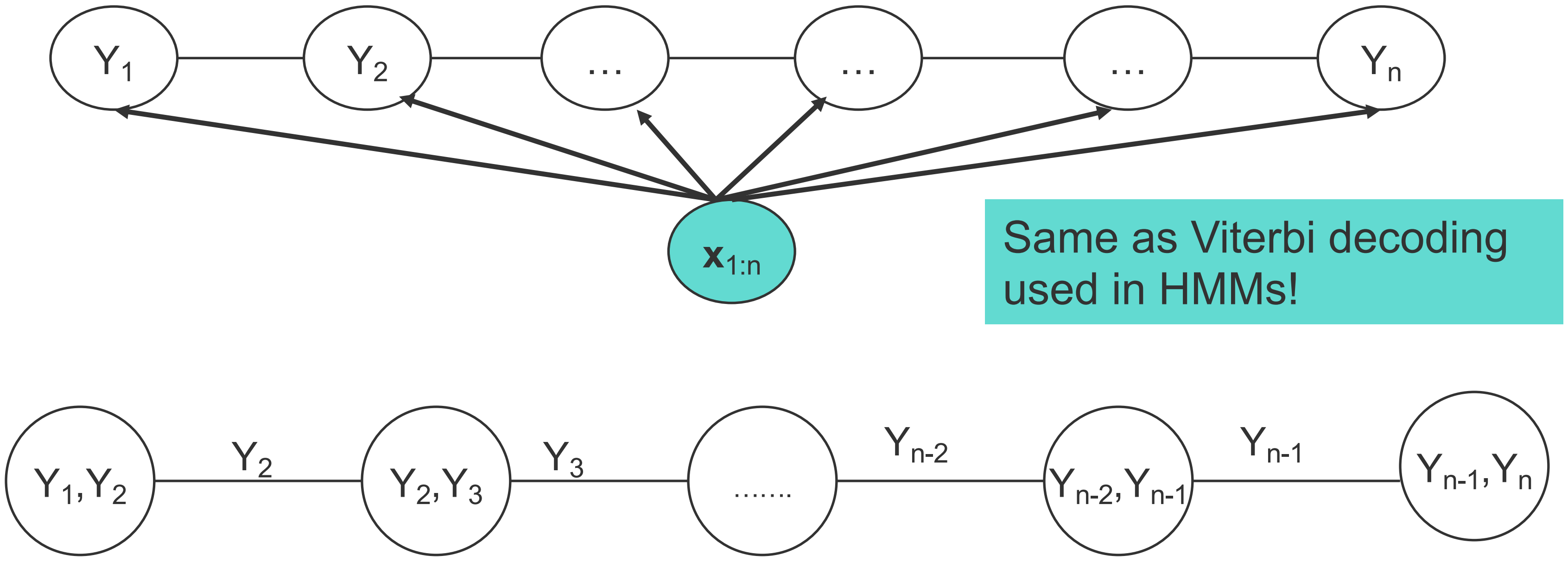
$$\text{where } Z(\mathbf{x},\lambda,\mu) = \sum_{\mathbf{y}} \exp\Big(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x}))\Big)$$

CRFs: Inference

- Given CRF parameters λ and μ, find the $\mathbf{y}^*$ that maximizes P(y|x)

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \exp\left(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x}))\right)$$

  - Can ignore Z(x) because it is not a function of y
- Run the max-product algorithm on the junction-tree of CRF:



Same as Viterbi decoding used in HMMs!

CRF learning

- Given $\{(\mathbf{x}_d, \mathbf{y}_d)\}_{d=1}^N$, find λ^*, μ^* such that

$$\begin{aligned}
\lambda*, \mu* &= \arg\max_{\lambda,\mu} L(\lambda,\mu) = \arg\max_{\lambda,\mu} \prod_{d=1}^{N} P(\mathbf{y}_d|\mathbf{x}_d,\lambda,\mu) \\
&= \arg\max_{\lambda,\mu} \prod_{d=1}^{N} \frac{1}{Z(\mathbf{x}_d,\lambda,\mu)} \exp(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d))) \\
&= \arg\max_{\lambda,\mu} \sum_{d=1}^{N}(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d)) - \log Z(\mathbf{x}_d,\lambda,\mu))
\end{aligned}$$

- Computing the gradient w.r.t λ:

Gradient of the log-partition function in an exponential family is the expectation of the sufficient statistics.

$$\nabla_\lambda L(\lambda,\mu) = \sum_{d=1}^{N}(\sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{\mathbf{y}}(P(\mathbf{y}|\mathbf{x_d}) \sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d)))$$

CRF learning

$$\nabla_\lambda L(\lambda,\mu) = \sum_{d=1}^{N}\left(\sum_{i=1}^{n}\mathbf{f}(y_{d,i},y_{d,i-1},\mathbf{x}_d) - \sum_{\mathbf{y}}\left(P(\mathbf{y}|\mathbf{x_d})\sum_{i=1}^{n}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)\right)\right)$$

- Computing the model expectations:
  - Requires exponentially large number of summations: Is it intractable?

$$\sum_{\mathbf{y}}\left(P(\mathbf{y}|\mathbf{x}_d)\sum_{i=1}^{n}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)\right) = \sum_{i=1}^{n}\left(\sum_{\mathbf{y}}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)P(\mathbf{y}|\mathbf{x}_d)\right)$$

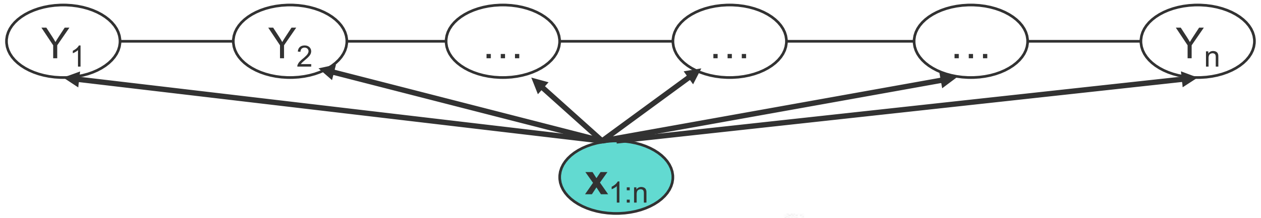
$$= \sum_{i=1}^{n}\sum_{y_i,y_{i-1}}\mathbf{f}(y_i,y_{i-1},\mathbf{x}_d)P(y_i,y_{i-1}|\mathbf{x}_d)$$

Expectation of **f** over the corresponding marginal probability of neighboring nodes!!

- Tractable!
  - Can compute marginals using the sum-product algorithm on the chain

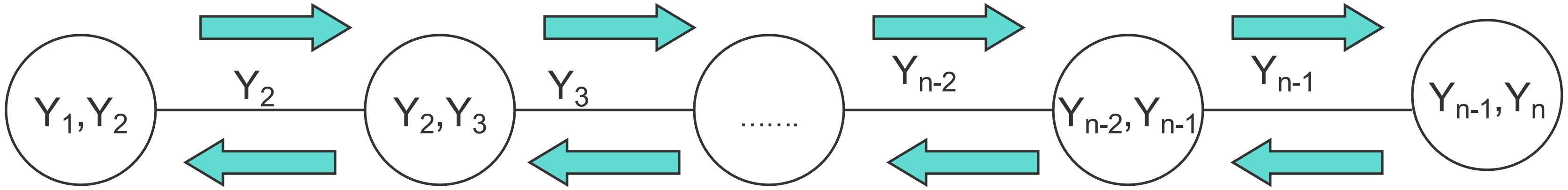
CRF learning

- Computing marginals using junction-tree calibration:



- Junction Tree Initialization:

$$\alpha^0(y_i, y_{i-1}) = \exp(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_i, \mathbf{x}_d))$$



- After calibration:

$$P(y_i, y_{i-1} | \mathbf{x}_d) \propto \alpha(y_i, y_{i-1})$$

$$\Rightarrow P(y_i, y_{i-1} | \mathbf{x}_d) = \frac{\alpha(y_i, y_{i-1})}{\sum_{y_i, y_{i-1}} \alpha(y_i, y_{i-1})} = \alpha'(y_i, y_{i-1})$$

Also called
forward-backward algorithm

CRF learning

- Computing feature expectations using calibrated potentials:

$$\sum_{y_i, y_{i-1}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) P(y_i, y_{i-1} | \mathbf{x}_d) = \sum_{y_i, y_{i-1}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) \alpha'(y_i, y_{i-1})$$

- Now we know how to compute $r_{\lambda}L(\lambda,\mu)$:

$$\begin{aligned}
\nabla_{\lambda} L(\lambda, \mu) &= \sum_{d=1}^{N} (\sum_{i=1}^{n} \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{\mathbf{y}} (P(\mathbf{y}|\mathbf{x_d}) \sum_{i=1}^{n} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d))) \\
&= \sum_{d=1}^{N} (\sum_{i=1}^{n} (\mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) - \sum_{y_i, y_{i-1}} \alpha'(y_i, y_{i-1}) \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d)))
\end{aligned}$$

- Learning can now be done using gradient ascent:

$$\begin{aligned}
\lambda^{(t+1)} &= \lambda^{(t)} + \eta \nabla_{\lambda} L(\lambda^{(t)}, \mu^{(t)}) \\
\mu^{(t+1)} &= \mu^{(t)} + \eta \nabla_{\mu} L(\lambda^{(t)}, \mu^{(t)})
\end{aligned}$$

CRF learning

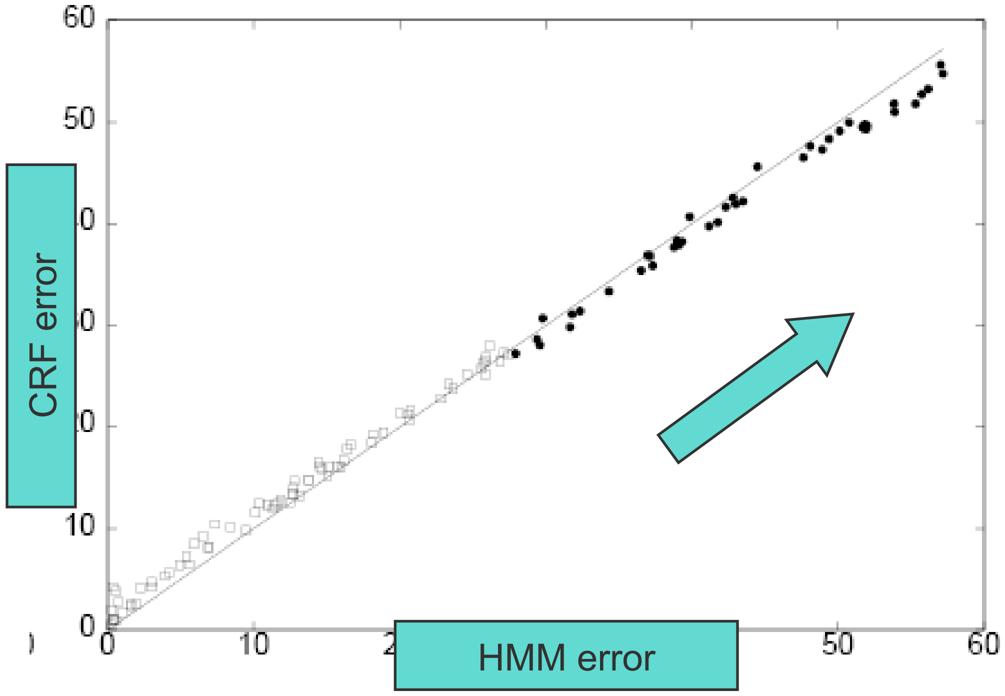
- In practice, we use a Gaussian Regularizer for the parameter vector to improve generalizability

$$\lambda*, \mu* \quad = \quad \arg\max_{\lambda,\mu} \sum_{d=1}^{N} \log P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu) - \frac{1}{2\sigma^2}(\lambda^T\lambda + \mu^T\mu)$$

- In practice, gradient ascent has very slow convergence
  - Alternatives:
    - Conjugate Gradient method
    - Limited Memory Quasi-Newton Methods

CRFs: some empirical results

- Comparison of error rates on synthetic data



Data is increasingly higher order in the direction of arrow

CRFs achieve the lowest error rate for higher order data

CRFs: some empirical results

- Parts of Speech tagging

model	error	oov error
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM$^+$	4.81%	26.99%
CRF$^+$	4.27%	23.76%

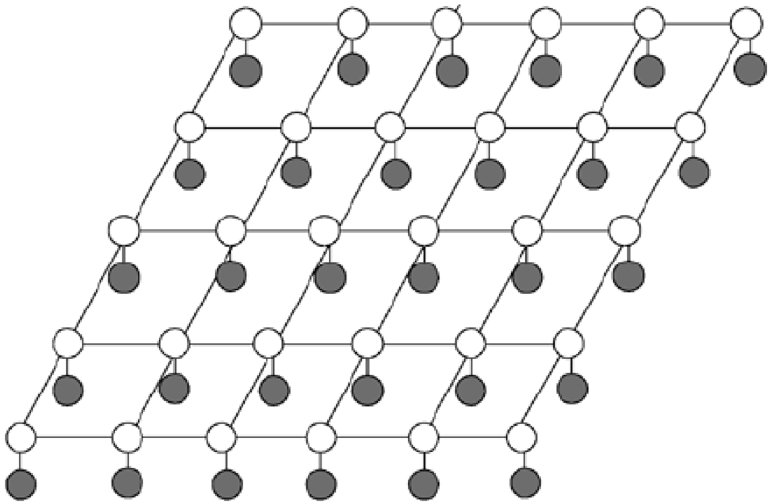
$^+$Using spelling features

- Using same set of features: HMM >=< CRF > MEMM
- Using additional overlapping features: CRF$^+$ > MEMM$^+$ >> HMM

Supplementary

Other CRFs

- So far we have discussed only 1-dimensional chain CRFs
  - Inference and learning: exact
- We could also have CRFs for arbitrary graph structure
  - E.g: Grid CRFs
  - Inference and learning no longer tractable
  - Approximate techniques used
    - MCMC Sampling
    - Variational Inference
    - Loopy Belief Propagation
  - We will discuss these techniques soon



Applications of CRF in Vision

Stereo Matching

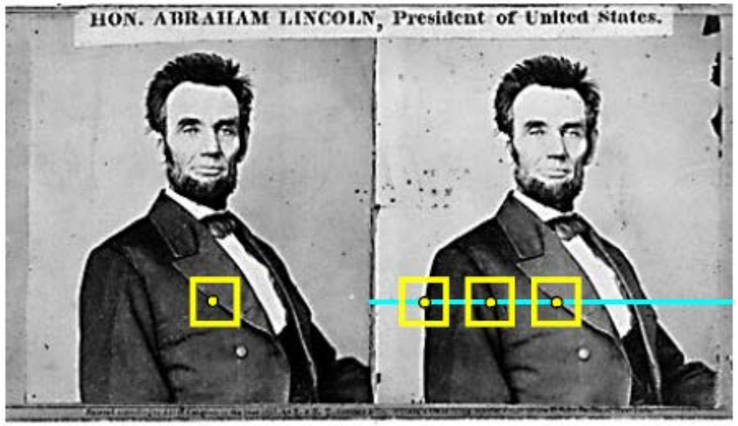


Image Segmentation

Image Restoration

Application: Image Segmentation

$\phi_i(y_i, x) \in \mathbb{R}^{\approx 1000}$: local image features, e.g. bag-of-words

$\rightarrow \quad \langle w_i, \phi_i(y_i, x) \rangle$: local classifier (like logistic-regression)

$\phi_{i,j}(y_i, y_j) = [\![y_i = y_j]\!] \in \mathbb{R}^1$: test for same label

$\rightarrow \quad \langle w_{ij}, \phi_{ij}(y_i, y_j) \rangle$: penalizer for label changes (if $w_{ij} > 0$)

combined: $\operatorname{argmax}_y p(y|x)$ is smoothed version of local cues

original

local classification

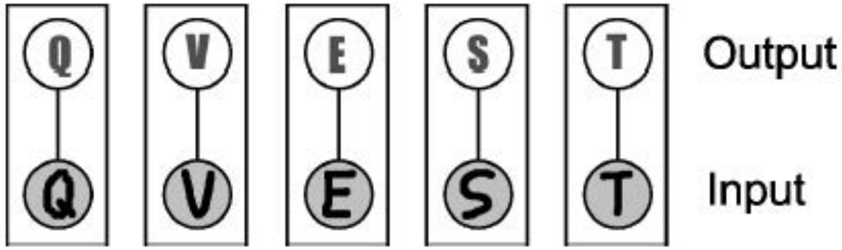
local + smoothness

Application: Handwriting Recognition

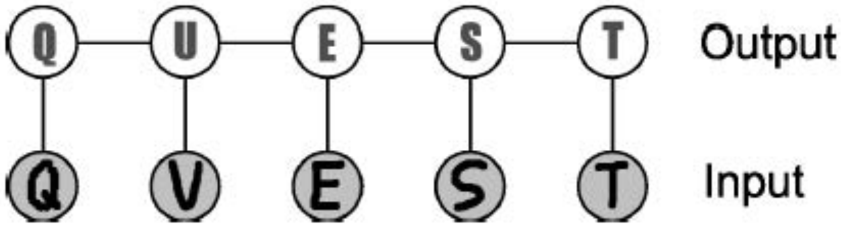
$\phi_i(y_i, x) \in \mathbb{R}^{\approx 1000}$: image representation (pixels, gradients)
$\rightarrow \quad \langle w_i, \phi_i(y_i, x)\rangle$: local classifier if x_i is letter y_i

$\phi_{i,j}(y_i, y_j) = e_{y_i} \otimes e_{y_j} \in \mathbb{R}^{26\cdot 26}$: letter/letter indicator
$\rightarrow \quad \langle w_{ij}, \phi_{ij}(y_i, y_j)\rangle$: encourage/suppress letter combinations

combined: $\operatorname{argmax}_y p(y|x)$ is "corrected" version of local cues



local classification



local + "correction"

Application: Pose Estimation

$\phi_i(y_i, x) \in \mathbb{R}^{\approx 1000}$: local image representation, e.g. HoG

$\rightarrow \quad \langle w_i, \phi_i(y_i, x) \rangle$: local confidence map

$\phi_{i,j}(y_i, y_j) = \textit{good\_fit}(y_i, y_j) \in \mathbb{R}^1$: test for geometric fit

$\rightarrow \quad \langle w_{ij}, \phi_{ij}(y_i, y_j) \rangle$: penalizer for unrealistic poses

together: $\operatorname{argmax}_y p(y|x)$ is sanitized version of local cues

original   local classification   local + geometry

Feature Functions for CRF in Vision

$\phi_i(y_i, x)$: local representation, high-dimensional

$\rightarrow \quad \langle w_i, \phi_i(y_i, x) \rangle$: local classifier

$\phi_{i,j}(y_i, y_j)$: prior knowledge, low-dimensional

$\rightarrow \quad \langle w_{ij}, \phi_{ij}(y_i, y_j) \rangle$: penalize outliers

learning adjusts parameters:

- unary w_i: learn local classifiers and their importance
- binary w_{ij}: learn importance of smoothing/penalization

$\operatorname{argmax}_y p(y|x)$ is cleaned up version of local prediction

Case Study: Image Segmentation

- Image segmentation (FG/BG) by modeling of interactions btw RVs
  - Images are noisy.
  - Objects occupy continuous regions in an image.

[Nowozin,Lampert 2012]

Input image

Pixel-wise separate optimal labeling

Locally-consistent joint optimal labeling

$$Y^* = \arg\max_{y \in \{0,1\}^n} \Bigg[\underbrace{\sum_{i \in S} V_i(y_i, X)}_{\text{Unary Term}} + \underbrace{\sum_{i \in S} \sum_{j \in N_i} V_{i,j}(y_i, y_j)}_{\text{Pairwise Term}} \Bigg].$$

Y: labels
X: data (features)
S: pixels
N_i: neighbors of pixel i

Discriminative Random Fields

- A special type of CRF
  - The unary and pairwise potentials are designed using local discriminative classifiers.
  - Posterior

$$P(Y \mid X) = \frac{1}{Z}\exp\Big(\sum_{i\in S} \underbrace{A_i(y_i, X)}_{\text{Association}} + \sum_{i\in S}\sum_{j\in N_i} \underbrace{I_{ij}(y_i, y_j, X)}_{\text{Interaction}}\Big)$$

- Association Potential
  - Local discriminative model for site i: using logistic link with GLM.

$$A_i(y_i, X) = \log P(y_i \mid f_i(X)) \qquad P(y_i = 1 \mid f_i(X)) = \frac{1}{1+\exp(-(w^T f_i(X)))} = \sigma(w^T f_i(X))$$

- Interaction Potential
  - Measure of how likely site *i* and *j* have the same label given *X*

$$I_{ij}(y_i, y_j, X) = \underbrace{k y_i y_j}_{(1)} + \underbrace{(1-k)(2\sigma(y_i y_j \mu_{ij}(X)) - 1)}_{(2)}$$

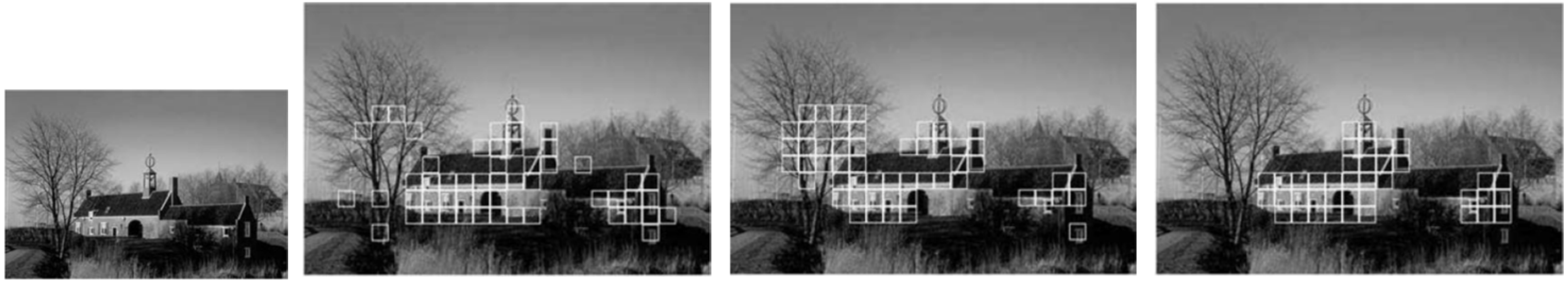
(1) Data-independent smoothing term (2) Data-dependent pairwise logistic function

S. Kumar and M. Hebert. Discriminative Random Fields. IJCV, 2006.

DRF Results

- Task: Detecting man-made structure in natural scenes.
  - Each image is divided in non-overlapping 16x16 tile blocks.

- An example



Input image Logistic MRF DRF

- Logistic: No smoothness in the labels
- MRF: Smoothed False positive. Lack of neighborhood interaction of the data

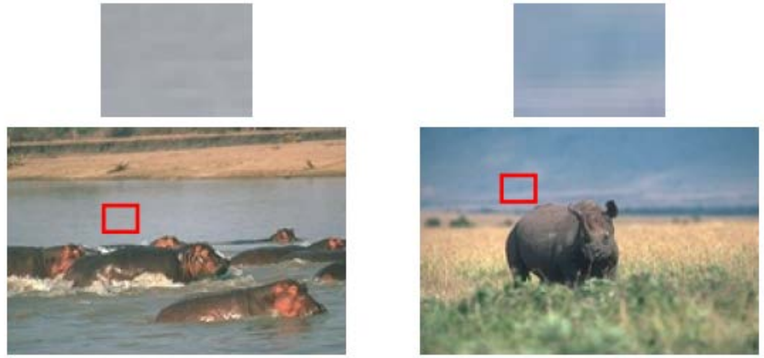
S. Kumar and M. Hebert. Discriminative Random Fields. IJCV, 2006.

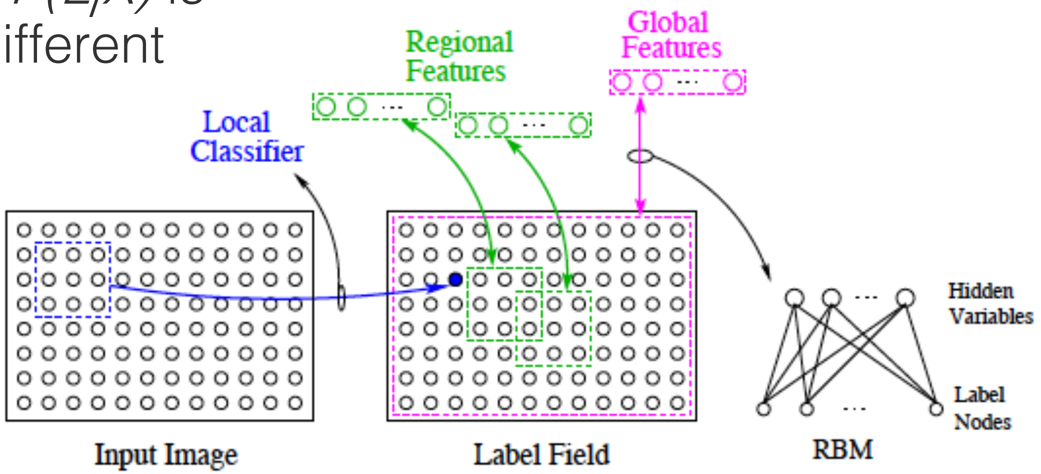
Multiscale Conditional Random Fields

- Considering features in different scales
  - Local Features (site)
  - Regional Label Features (small patch)
  - Global Label Features (big patch or the whole image)
- The conditional probability *P(L|X)* is formulated by features in different scales

$$P(L \mid X) = \frac{1}{Z} \prod_{s} P_s(L \mid X)$$

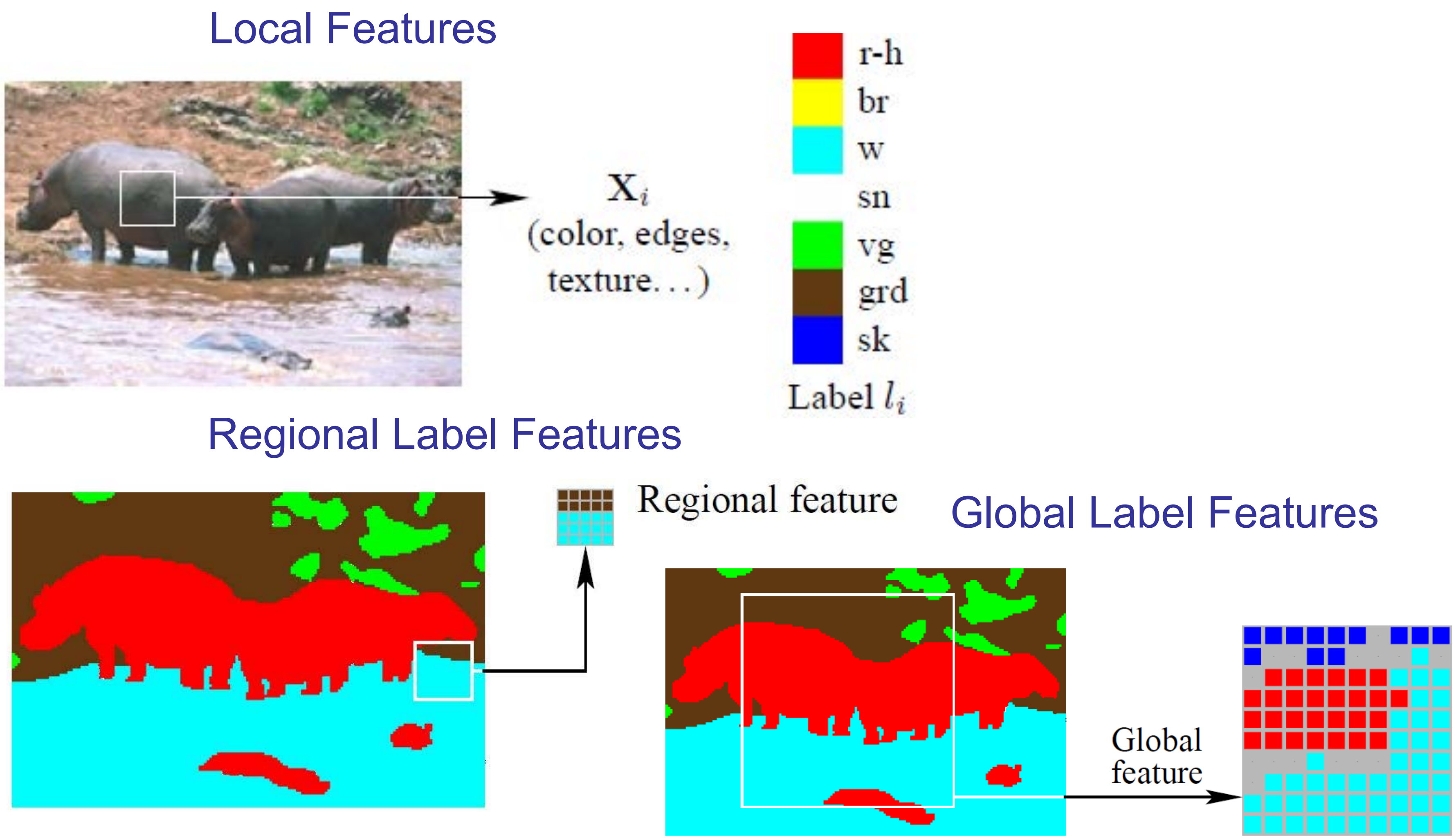
$$Z = \sum_{L} \prod_{s} P_s(L \mid X)$$





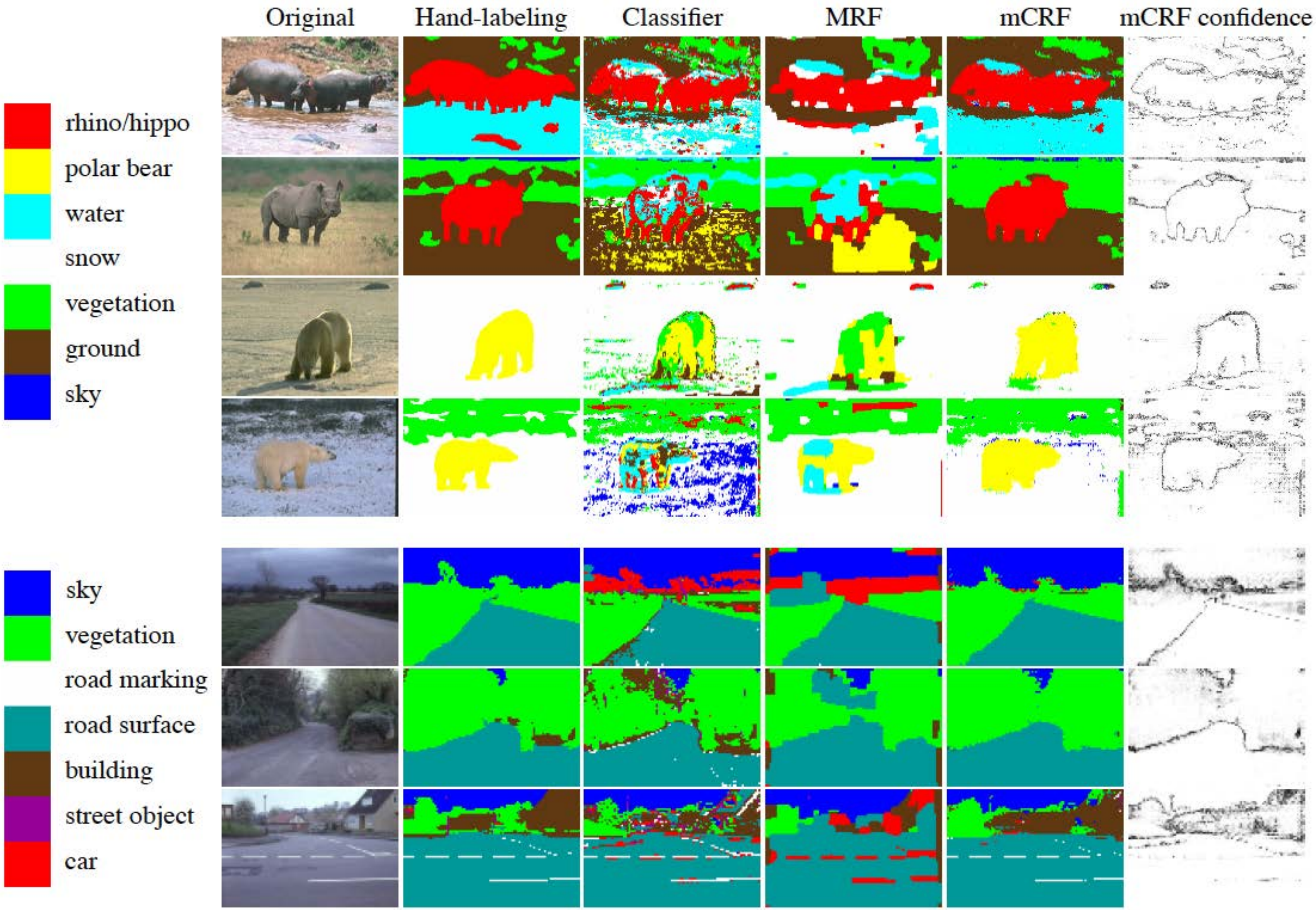
He, X. et. al.: Multiscale conditional random fields for image labeling. CVPR 2004

Multiscale Conditional Random Fields



He, X. et. al.: Multiscale conditional random fields for image labeling. CVPR 2004

mC

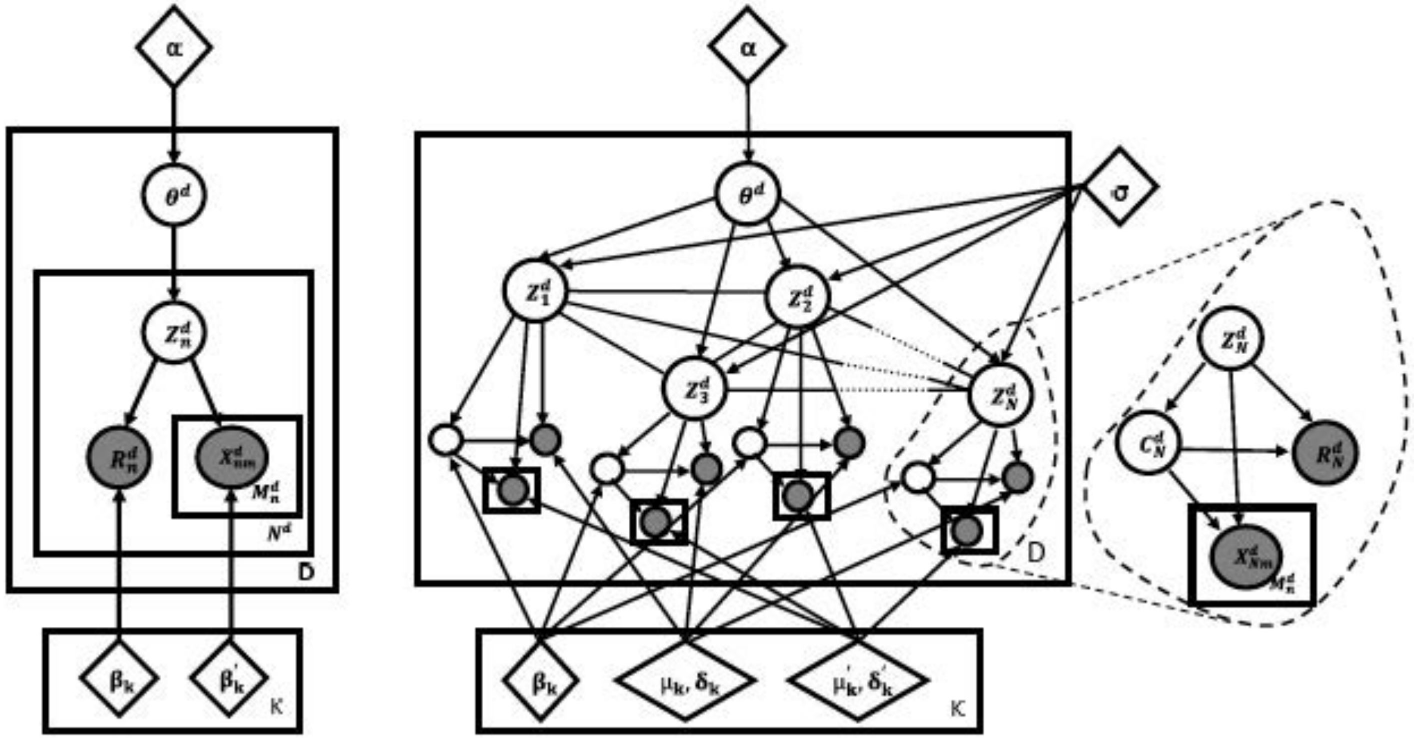


He, X. et. al.: Multiscale conditional random fields for image labeling. CVPR 2004

Topic Random Fields

- Spatial MRF over topic assignments

$$p(\mathbf{z}^d|\boldsymbol{\theta}^d,\sigma) = \frac{1}{A(\boldsymbol{\theta}^d,\sigma)} \exp\left[\sum_n \sum_k z^d_{nk} \log \theta^d_k + \sum_{n\sim m} \sigma I(z^d_n = z^d_m)\right]$$

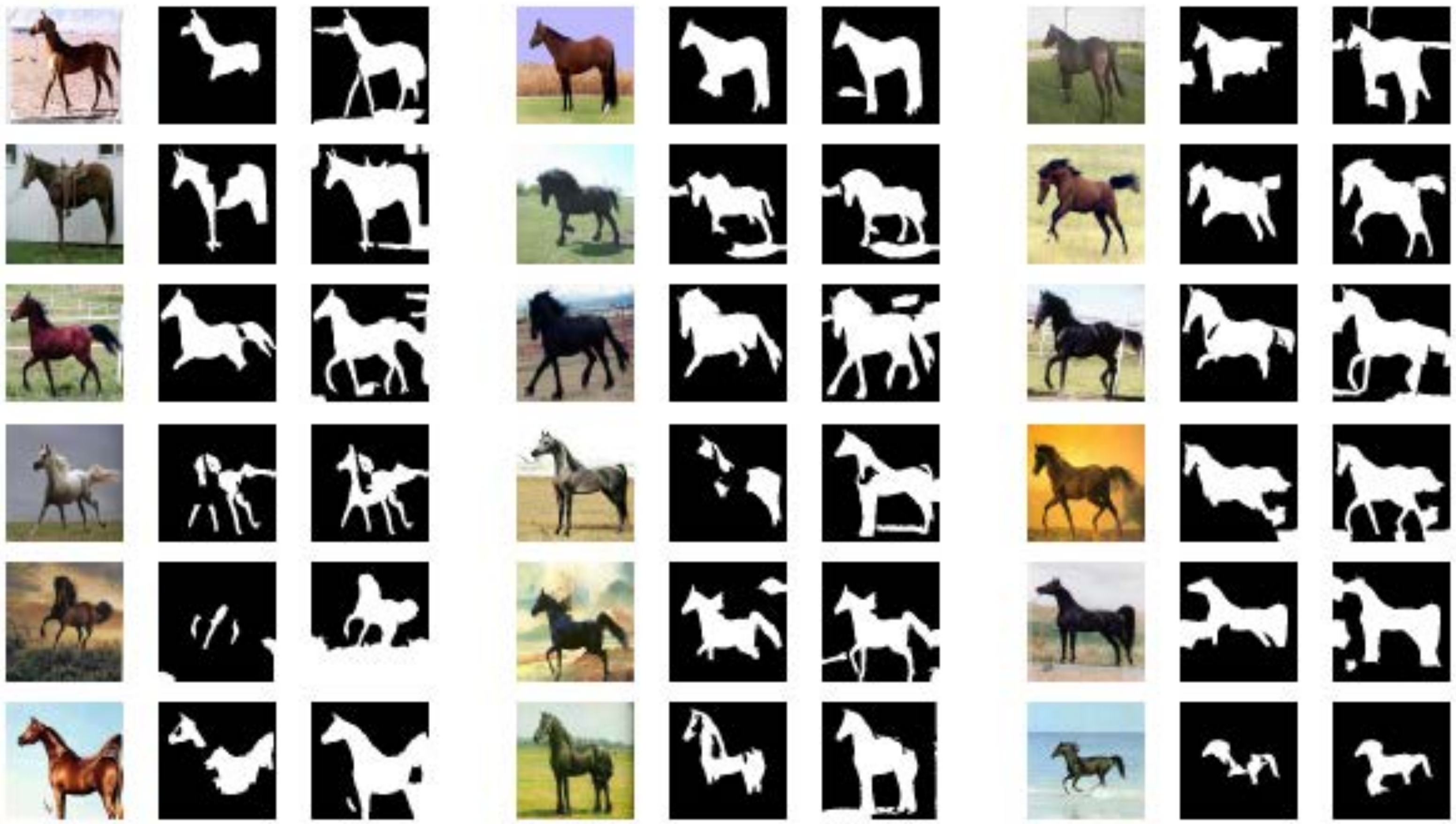


(a) Spatial LDA (b) TRF

Zhao, B. et. al.: Topic random fields for image segmentation. ECCV 2010

TRF Results

Spatial LDA vs. Topic Random Fields



Zhao, B. et. al.: Topic random fields for image segmentation. ECCV 2010

Summary

- Conditional Random Fields are partially directed discriminative models
- They overcome the label bias problem of HMM by using a global normalizer
- Inference for 1-D chain CRFs is exact
  - Same as Max-product or Viterbi decoding
- Learning also is exact
  - globally optimum parameters can be learned
  - Requires using sum-product or forward-backward algorithm
- CRFs involving arbitrary graph structure are intractable in general
  - E.g.: Grid CRFs
  - Inference and learning require approximation techniques
    - MCMC sampling
    - Variational methods
    - Loopy BP