

Conditional Random Fields and Structured Perceptron

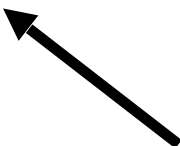
Instructor: Wei Xu
Ohio State University
CSE 5525

Some slides from Eric Xing and Michael Collins

Previously, we saw MEMMs...

$$\begin{aligned} P(t_1 \dots t_n | w_1 \dots w_n) &= \prod_{i=1}^n q(t_i | t_{i-1}, w_1 \dots w_n, i) \\ &= \prod_{i=1}^n \frac{e^{v \cdot f(t_i, t_{i-1}, w_1 \dots w_n, i)}}{\sum_{t'} e^{v \cdot f(t', t_{i-1}, w_1 \dots w_n, i)}} \end{aligned}$$

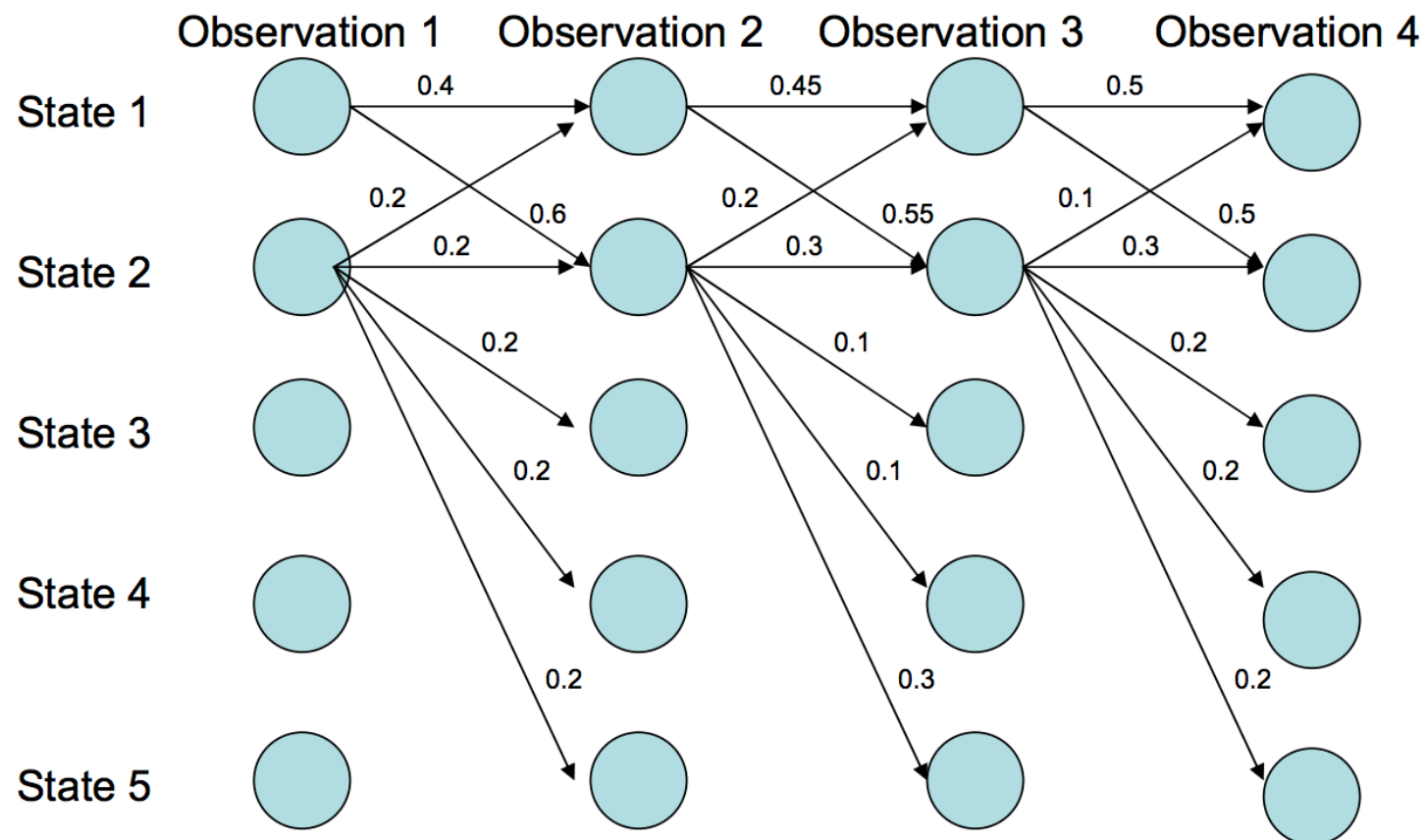
MEMMs: The Label Bias Problem

$$P(t_1, \dots, t_n | w_1 \dots w_n) = \prod_{i=1}^n \frac{e^{v \cdot f(t_i, t_{i-1}, w_1 \dots w_n, i)}}{\sum_{t'} e^{v \cdot f(t', t_{i-1}, w_1 \dots w_n, i)}}$$


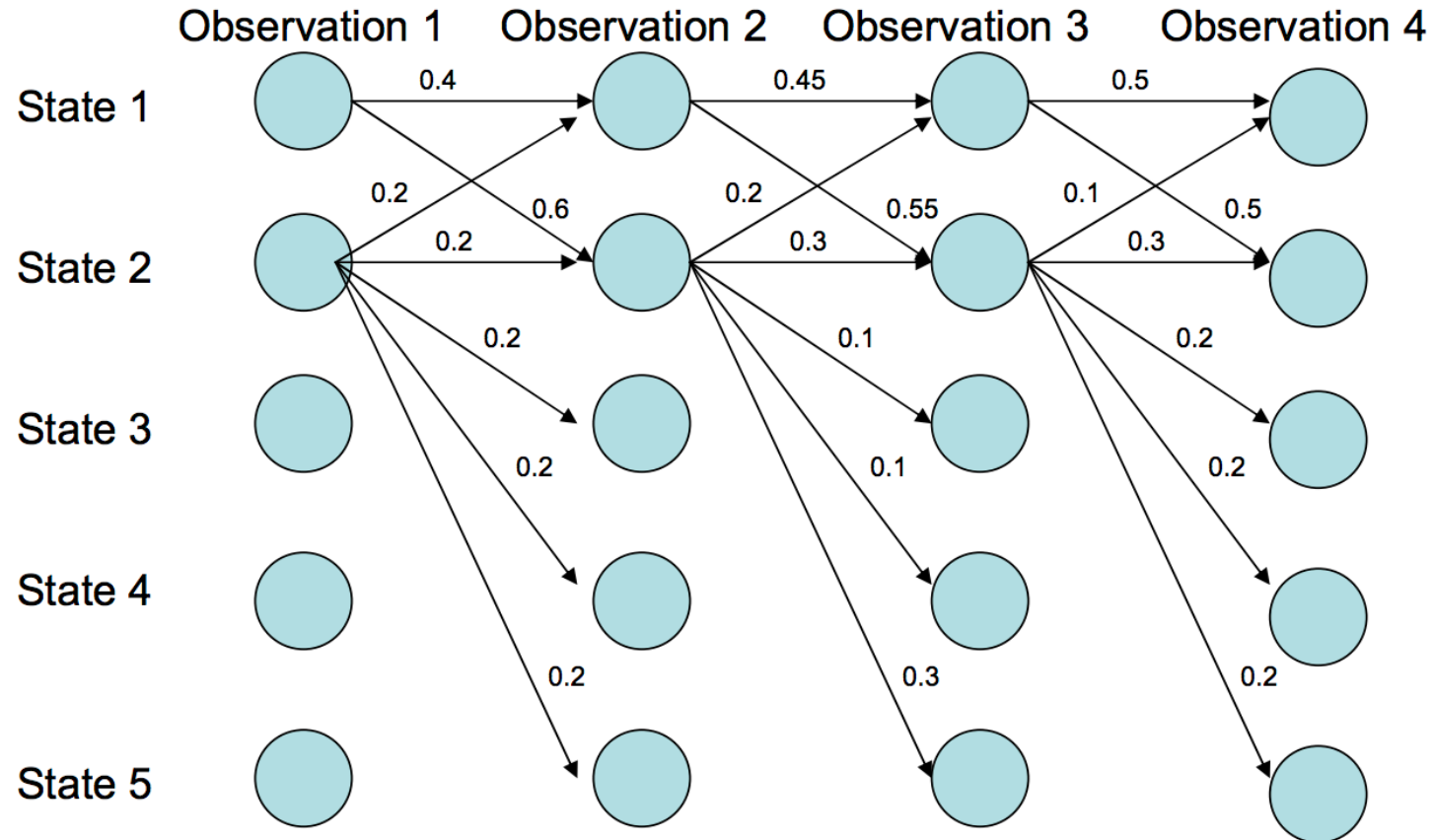
These are forced to sum to 1 Locally

Q: Is that really necessary?

MEMMs: The Label Bias Problem



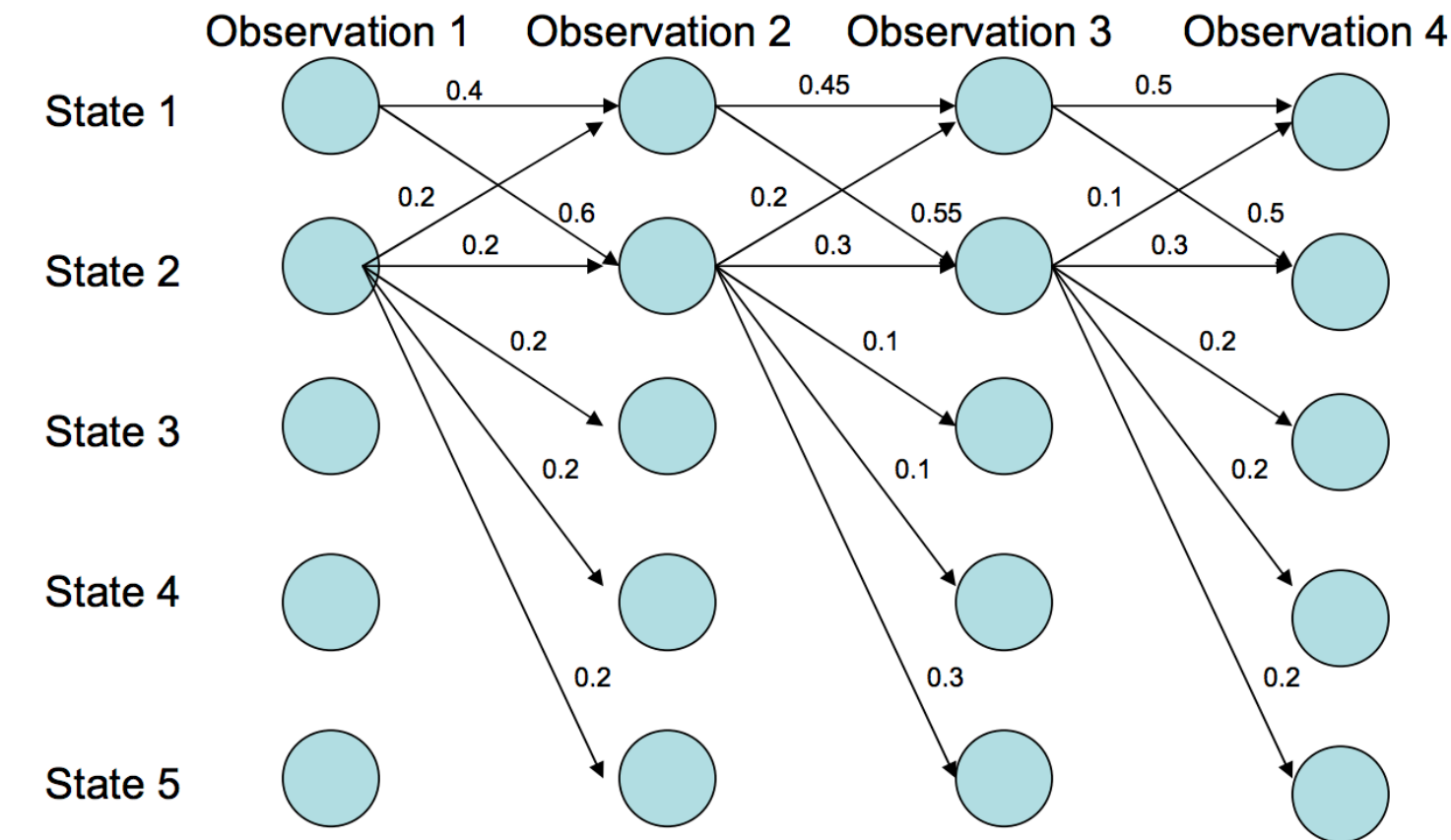
MEMMs: The Label Bias Problem



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

MEMMs: The Label Bias Problem



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

- $0.4 \times 0.55 \times 0.3 = 0.066$

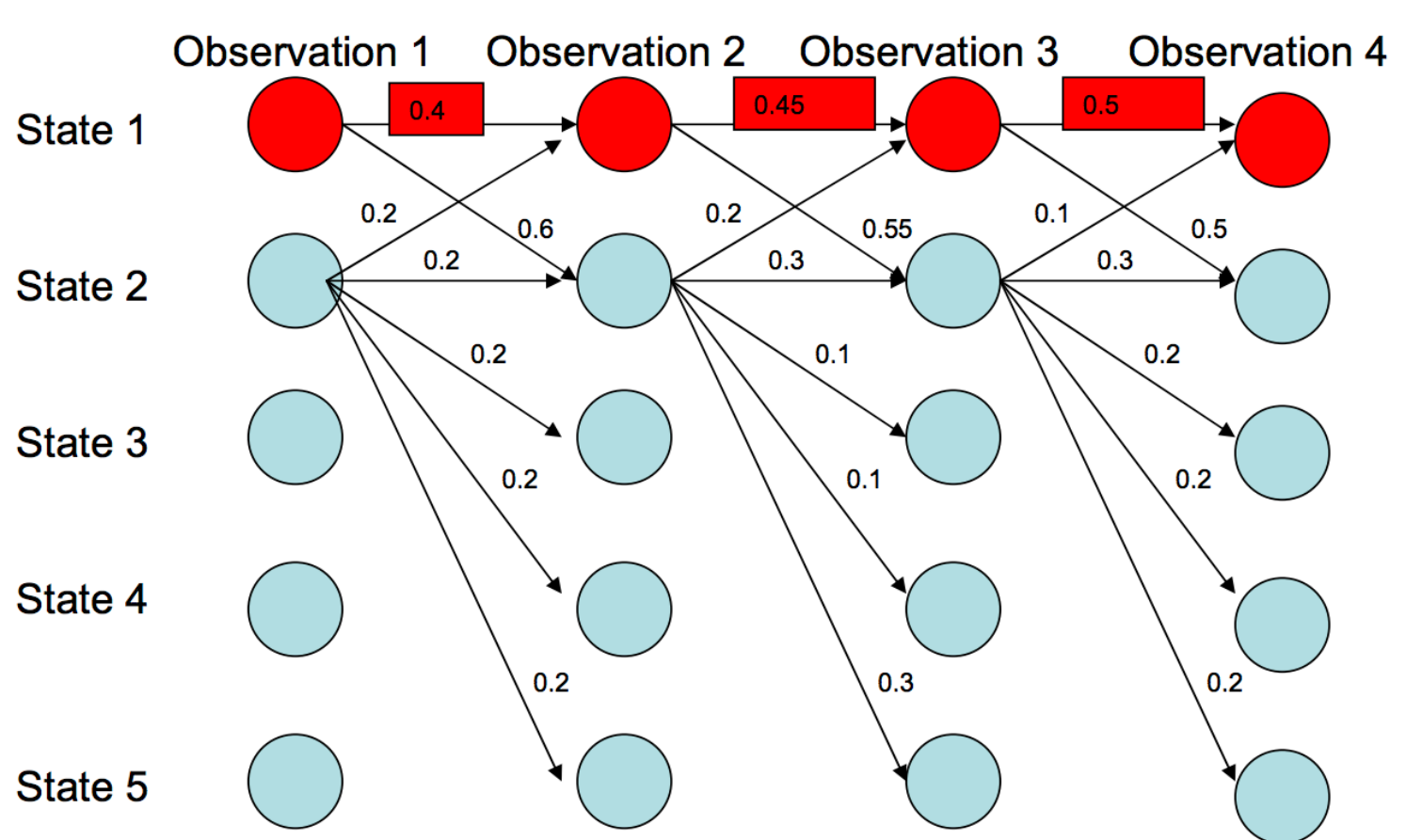
Other paths:

1->1->1->1: 0.09

2->2->2->2: 0.018

1->2->1->2: 0.06

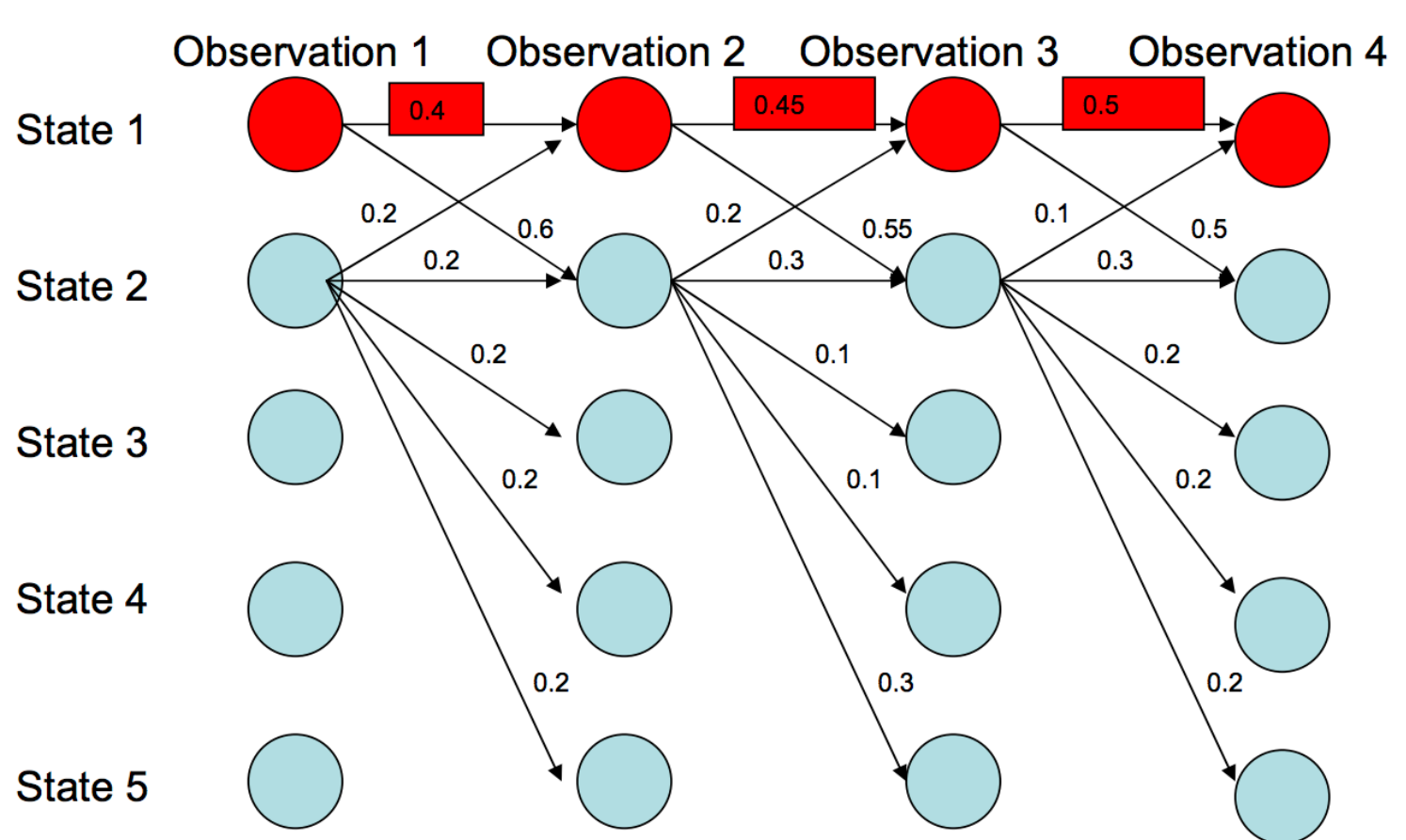
MEMMs: The Label Bias Problem



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?**

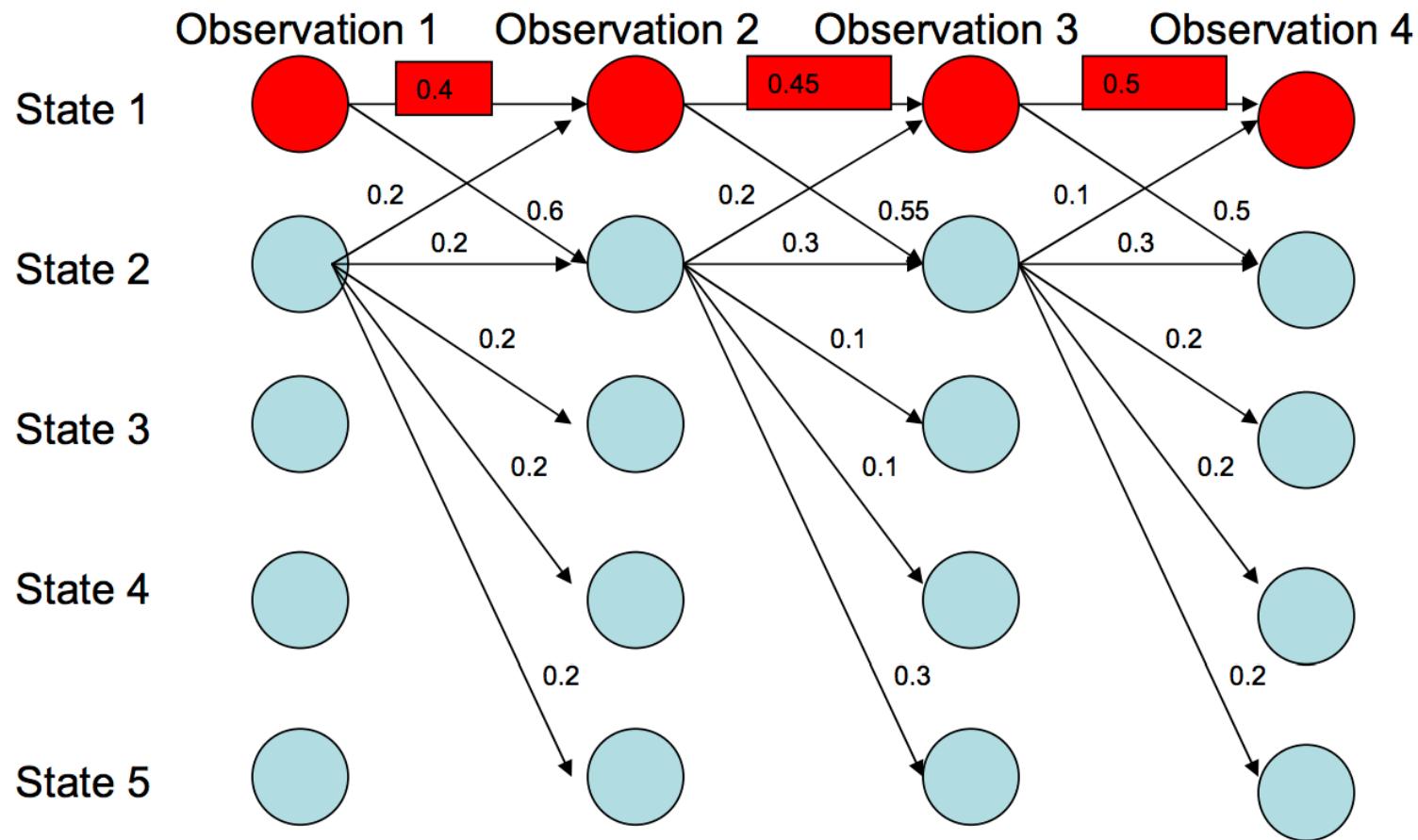
MEMMs: The Label Bias Problem



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

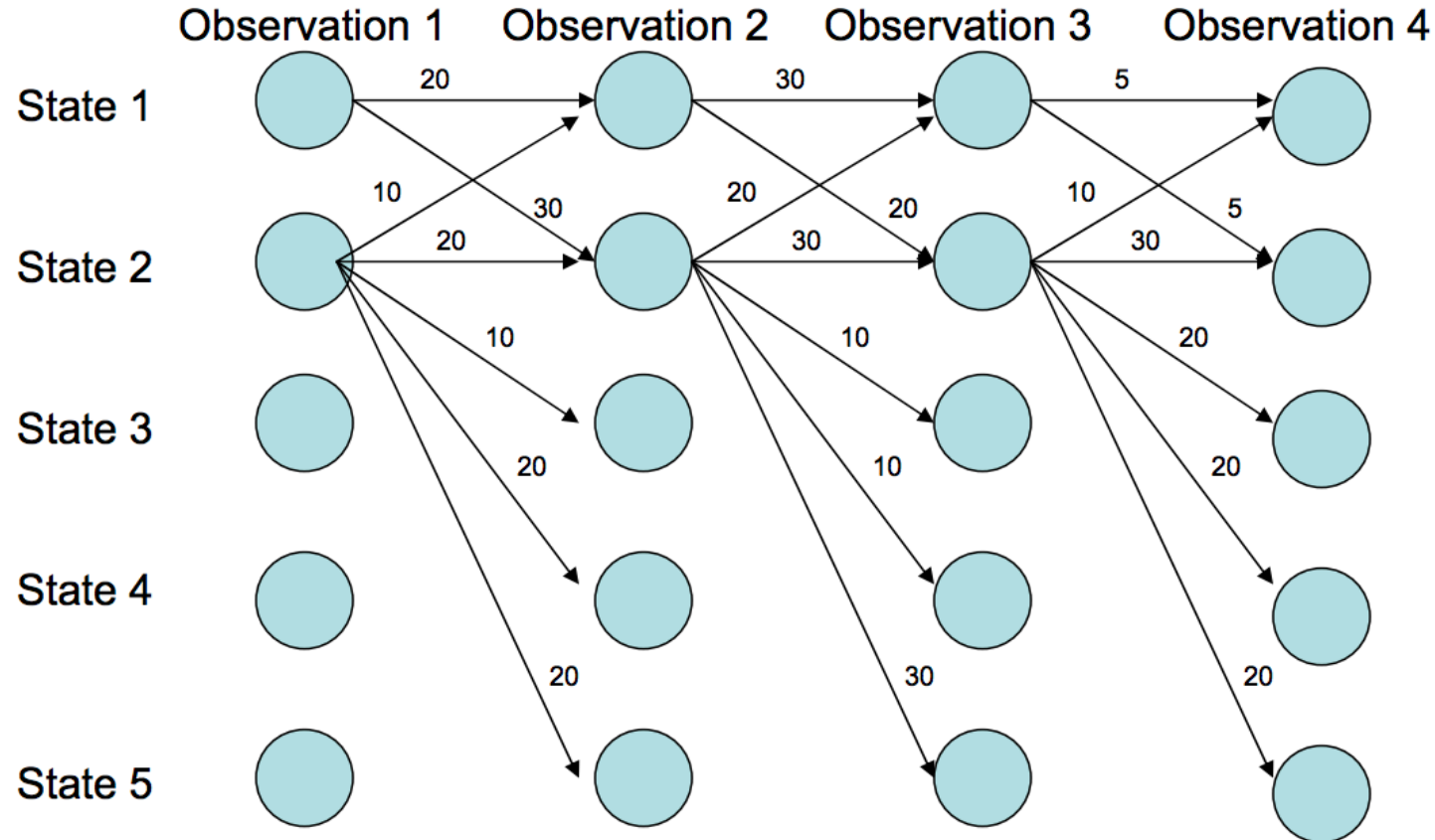
MEMMs: The Label Bias Problem



Label bias problem in MEMM:

- Preference of states with lower number of transitions over others

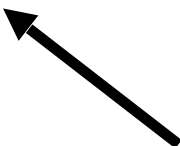
Solution: Do not normalize probabilities locally



From local probabilities to local potentials

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

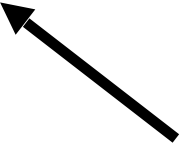
MEMMs: The Label Bias Problem

$$P(t_1, \dots, t_n | w_1 \dots w_n) = \prod_{i=1}^n \frac{e^{v \cdot f(t_i, t_{i-1}, w_1 \dots w_n, i)}}{\sum_{t'} e^{v \cdot f(t', t_{i-1}, w_1 \dots w_n, i)}}$$


These are forced to sum to 1 Locally

Q: Is that really necessary?

From MEMMs to Conditional Random Fields

$$P(t_1, \dots, t_n | w_1 \dots w_n) \propto \prod_{i=1}^n e^{v \cdot f(t_i, t_{i-1}, w_1 \dots w_n, i)}$$


Q: how can we make the distribution over tag sequences sum to 1?

From MEMMs to Conditional Random Fields

$$P(t_1, \dots, t_n | w_1 \dots w_n) = \frac{1}{Z(v, w_1, \dots, w_n)} \prod_{i=1}^n e^{v \cdot f(t_i, t_{i-1}, w_1 \dots w_n, i)}$$

$$Z(v, w_1, \dots, w_n) = \sum_{t_1, \dots, t_n} \prod_{i=1}^n e^{v \cdot f(t_i, t_{i-1}, w_1 \dots w_n, i)}$$

CRF uses global normalizer to overcome the label bias problem of MEMM

Conditional Random Fields

- Learning:
 - similar to MEMM (gradient descent or MAP perceptron)
- Inference:
 - similar to HMM (dynamic programming)
 - 1) given model parameters, find best tag sequence
 - 2) during learning, compute marginal probabilities and the Z

Gradient ascent

Loop While not converged:

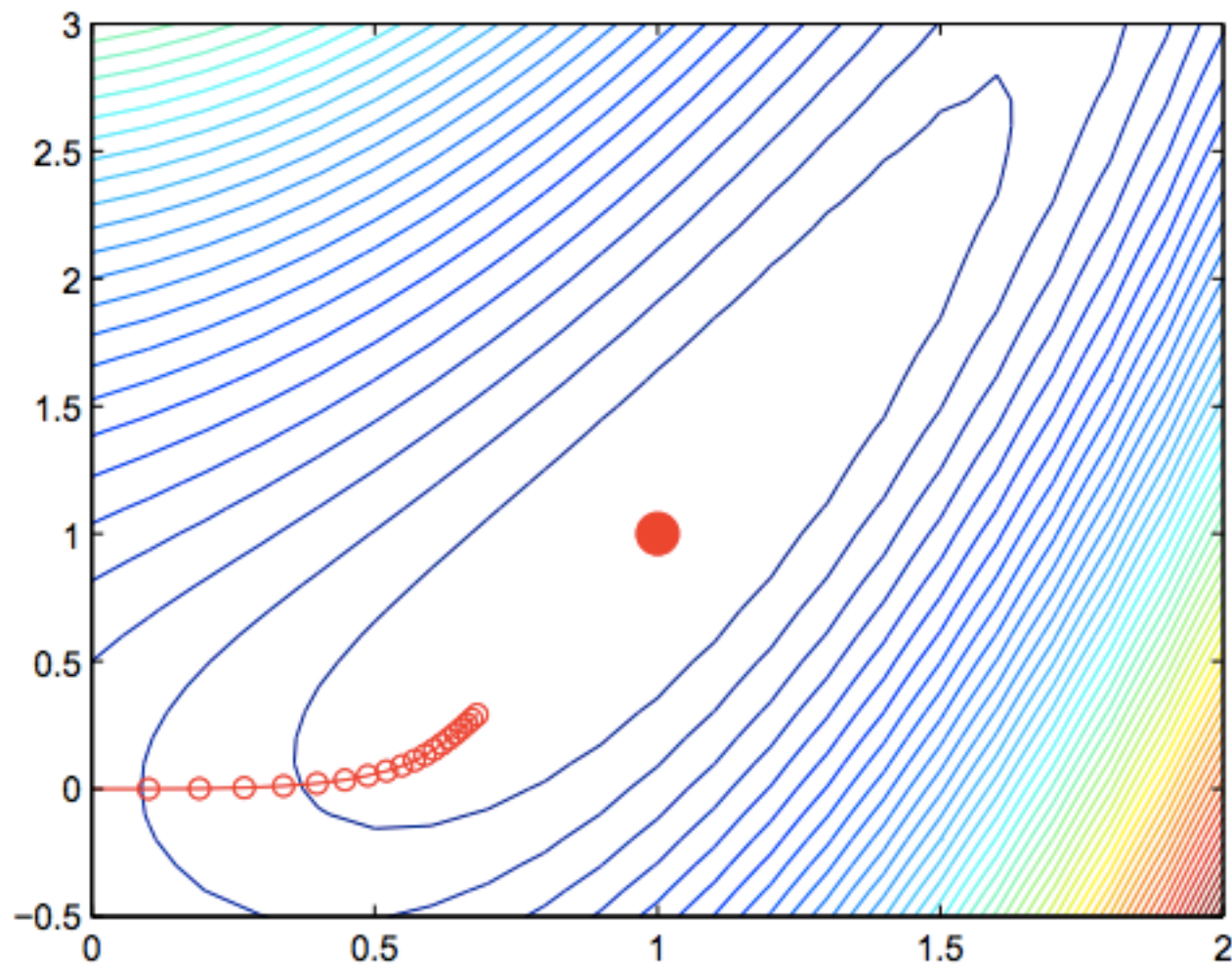
For all features j , compute and add derivatives

$$w_j^{\text{new}} = w_j^{\text{old}} + \eta \frac{\partial}{\partial w_j} \mathcal{L}(w)$$

$\mathcal{L}(w)$: Training set log-likelihood

$$\left(\frac{\partial \mathcal{L}}{\partial w_1}, \frac{\partial \mathcal{L}}{\partial w_2}, \dots, \frac{\partial \mathcal{L}}{\partial w_n} \right)$$

Gradient ascent



Gradient of Log-Linear Models

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^D f_j(y_i, d_i) - \sum_{i=1}^D \sum_{y \in Y} f_j(y, d_i) P(y|d_i)$$

MAP-based Learning (perceptron)

$$\frac{\partial \mathcal{L}}{\partial w_j} \approx \sum_{i=1}^D f_j(y_i, d_i) - \sum_{i=1}^D f_j(\arg \max_{y \in Y} P(y|d_i), d_i)$$

Conditional Random Field Gradient

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^D \sum_k f_j(t_k, t_{k-1}, w_1, \dots, w_n, k) - \sum_{i=1}^D \sum_{t_1, \dots, t_n} \sum_k f_j(t_k, t_{k-1}, w_1, \dots, w_n, k) P(t_1, \dots, t_n | w_1, \dots, w_n)$$

Tractable! Can be computed with the dynamic programming (Forward-Backward) algorithm

MAP-based learning (perceptron)

$$\frac{\partial \mathcal{L}}{\partial w_j} \approx \sum_{i=1}^D \sum_k f_j(t_k, t_{k-1}, w_1, \dots, w_n, k) - \sum_{i=1}^D \sum_k f_j(\arg \max_{t_1, \dots, t_n} P(t_1, \dots, t_n | w_1, \dots, w_n), w_1, \dots, w_n, k)$$

Training a Tagger using the Perceptron Algorithm

can be computed with
the dynamic
programming (Viterbi)
algorithm

Algorithm 40 **STRUCTUREDPERCEPTRONTRAIN**(**D**, *MaxIter*)

```
1:  $w \leftarrow 0$  // initialize weights
2: for  $iter = 1 \dots MaxIter$  do
3:   for all  $(x, y) \in D$  do
4:      $\hat{y} \leftarrow \operatorname{argmax}_{\hat{y} \in \mathcal{Y}(x)} w \cdot \phi(x, \hat{y})$  // compute prediction
5:     if  $\hat{y} \neq y$  then
6:        $w \leftarrow w + \phi(x, y) - \phi(x, \hat{y})$  // update weights
7:     end if
8:   end for
9: end for
10: return  $w$  // return learned weights
```

An Example

Say the correct tags for i 'th sentence are

the/~~DT~~ man/~~NN~~ bit/~~VBD~~ the/~~DT~~ dog/~~NN~~

Under current parameters, output is

the/~~DT~~ man/~~NN~~ bit/~~NN~~ the/~~DT~~ dog/~~NN~~

Assume also that features track: (1) all bigrams; (2) word/tag pairs

Parameters incremented:

$\langle \text{NN}, \text{VBD} \rangle, \langle \text{VBD}, \text{DT} \rangle, \langle \text{VBD} \rightarrow \text{bit} \rangle$

Parameters decremented:

$\langle \text{NN}, \text{NN} \rangle, \langle \text{NN}, \text{DT} \rangle, \langle \text{NN} \rightarrow \text{bit} \rangle$

Experiments

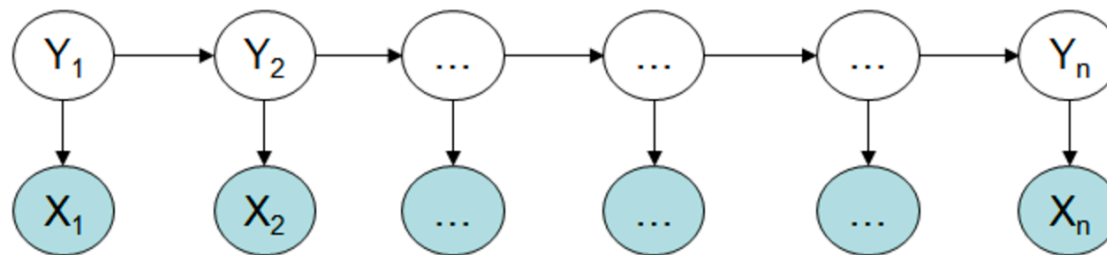
- ▶ Wall Street Journal part-of-speech tagging data

Perceptron = 2.89% error, Log-linear tagger = 3.28% error

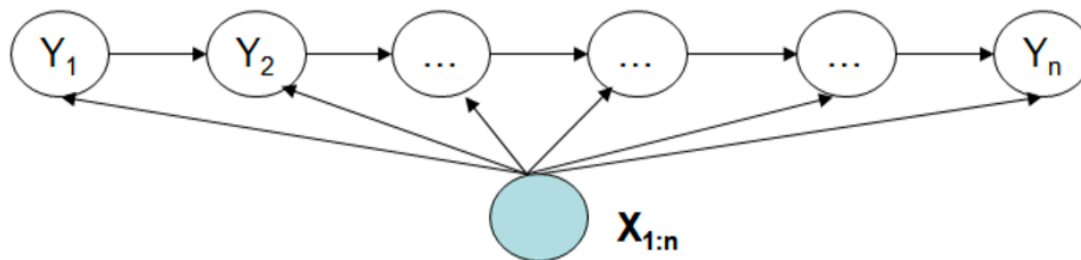
- ▶ [Ramshaw and Marcus, 1995] NP chunking data

Perceptron = 93.63% accuracy, Log-linear tagger = 93.29% accuracy

Summary

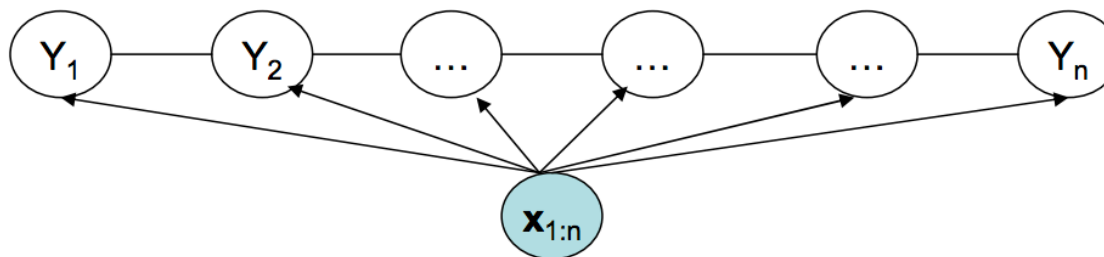


HMM



MEMM

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



CRF

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