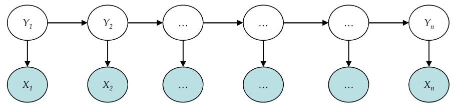
Maximum Entropy Markov Models and Conditional Random Fields

Ko, Youngjoong

Dept. of Computer Engineering,
Dong-A University

Motivation: Shortcomings of Hidden Markov Model



- **HMM models direct dependence between each state and only its corresponding observation**
  - NLP example: In a sentence segmentation task, segmentation may depend not just on a single word, but also on the features of the whole line such as line length, indentation, amount of white space, etc. (eg. *P(capitalization|tag), P(hyphen|tag), P(suffix|tag)*)
- **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**)

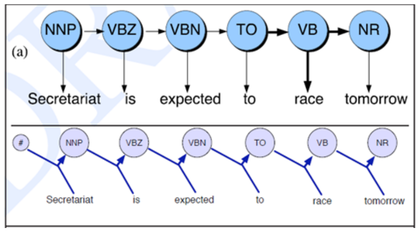
Solution: Maximum Entropy Markov Model (MEMM)

- **MEMM uses the Viterbi algorithm with MaxEnt**
  - POS tagging
  - HMM

$$\hat{T} = \operatorname*{argmax}_T P(T|W) = \operatorname*{argmax}_T P(W|T)P(T) = \operatorname*{argmax}_T \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})$$

vs. MEMM

$$\hat{T} = \operatorname*{argmax}_T P(T|W) = \operatorname*{argmax}_T \prod_i P(tag_i|word_i, tag_{i-1})$$



Solution: Maximum Entropy Markov Model (MEMM)

- **MEMM can condition on any useful feature of the input observation.**
  - HMM *vs.* MEMM

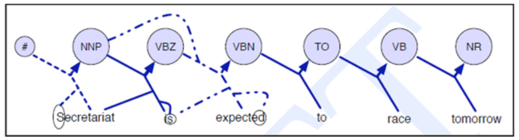
HMM:

$$P(Q|O) = \prod_{i=1}^{n} P(o_i|q_i) \times \prod_{i=1}^{n} P(q_i|q_{i-1})$$

MEMM:

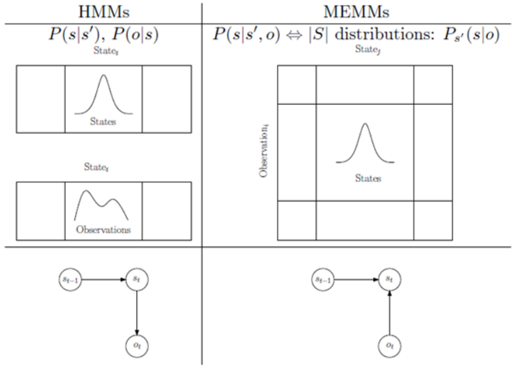
$$P(Q|O) = \prod_{i}^{n} P(q_i|q_{i-1}, o_i)$$

$$P(q|q', o) = \frac{1}{Z(o, q')} \exp\left(\sum_i w_i f_i(o, q)\right)$$



Solution: Maximum Entropy Markov Model (MEMM)

❖ **Summary of HMMs vs. MEMMs**



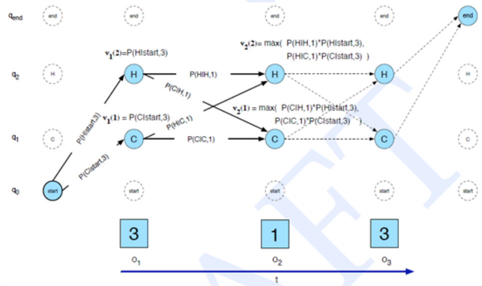
Solution: Maximum Entropy Markov Model (MEMM)

❖ **Decoding in MEMMs**

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t);\quad 1 \le j \le N, 1 < t \le T$$

➢ HMM $\quad v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i)\, P(o_t|s_j) \quad 1 \le j \le N, 1 < t \le T$

➢ MEMM $\quad v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i, o_t) \quad 1 \le j \le N, 1 < t \le T$

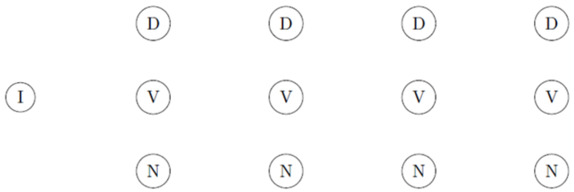


Solution: Maximum Entropy Markov Model (MEMM)

❖ **An Example of Viterbi in MEMMs**

"Matt saw the cat"

	I or N	V	D
Matt	$p_N = .9, p_V = .1$	$p_N = .8, p_V = .2$	$p_N = .9, p_V = .1$
saw	$p_N = .2, p_V = .8$	$p_N = .7, p_V = .3$	$p_N = 1$
the	$p_D = 1$	$p_D = 1$	$p_D = 1$
cat	$p_N = .9, p_V = .1$	$p_N = .95, p_V = .05$	$p_N = 1$

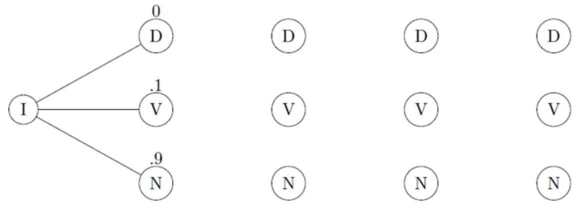


Solution: Maximum Entropy Markov Model (MEMM)

❖ **An Example of Viterbi in MEMMs**

"*Matt* saw the cat"

	I or N	V	D
Matt	$p_N = .9, p_V = .1$	$p_N = .8, p_V = .2$	$p_N = .9, p_V = .1$
saw	$p_N = .2, p_V = .8$	$p_N = .7, p_V = .3$	$p_N = 1$
the	$p_D = 1$	$p_D = 1$	$p_D = 1$
cat	$p_N = .9, p_V = .1$	$p_N = .95, p_V = .05$	$p_N = 1$

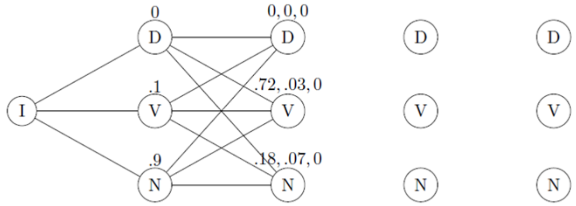


Solution: Maximum Entropy Markov Model (MEMM)

❖ An Example of Viterbi in MEMMs

"Matt *saw* the cat"

	I or *N*	*V*	*D*
Matt	$p_N = .9, p_V = .1$	$p_N = .8, p_V = .2$	$p_N = .9, p_V = .1$
saw	$p_N = .2, p_V = .8$	$p_N = .7, p_V = .3$	$p_N = 1$
the	$p_D = 1$	$p_D = 1$	$p_D = 1$
cat	$p_N = .9, p_V = .1$	$p_N = .95, p_V = .05$	$p_N = 1$

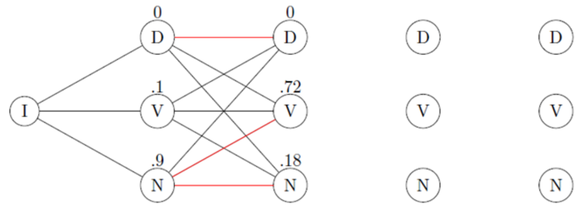


Solution: Maximum Entropy Markov Model (MEMM)

❖ An Example of Viterbi in MEMMs

"Matt *saw* the cat"

	I or *N*	*V*	*D*
Matt	$p_N = .9, p_V = .1$	$p_N = .8, p_V = .2$	$p_N = .9, p_V = .1$
saw	$p_N = .2, p_V = .8$	$p_N = .7, p_V = .3$	$p_N = 1$
the	$p_D = 1$	$p_D = 1$	$p_D = 1$
cat	$p_N = .9, p_V = .1$	$p_N = .95, p_V = .05$	$p_N = 1$

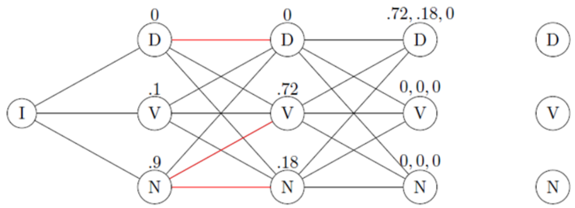


Solution: Maximum Entropy Markov Model (MEMM)

❖ An Example of Viterbi in MEMMs

"Matt saw *the* cat"

	I or *N*	*V*	*D*
Matt	$p_N = .9, p_V = .1$	$p_N = .8, p_V = .2$	$p_N = .9, p_V = .1$
saw	$p_N = .2, p_V = .8$	$p_N = .7, p_V = .3$	$p_N = 1$
the	$p_D = 1$	$p_D = 1$	$p_D = 1$
cat	$p_N = .9, p_V = .1$	$p_N = .95, p_V = .05$	$p_N = 1$

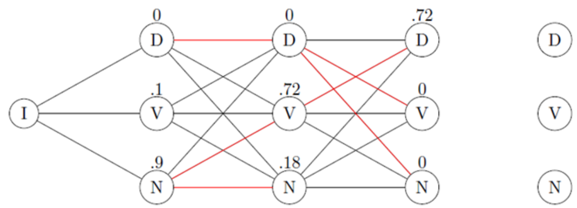


Solution: Maximum Entropy Markov Model (MEMM)

❖ An Example of Viterbi in MEMMs

"Matt saw *the* cat"

	I or *N*	*V*	*D*
Matt	$p_N = .9, p_V = .1$	$p_N = .8, p_V = .2$	$p_N = .9, p_V = .1$
saw	$p_N = .2, p_V = .8$	$p_N = .7, p_V = .3$	$p_N = 1$
the	$p_D = 1$	$p_D = 1$	$p_D = 1$
cat	$p_N = .9, p_V = .1$	$p_N = .95, p_V = .05$	$p_N = 1$

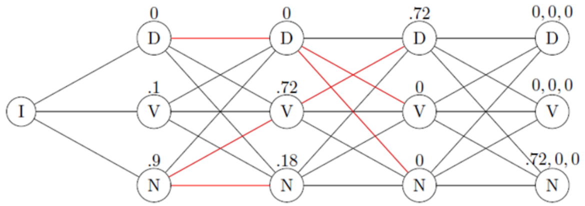


Solution: Maximum Entropy Markov Model (MEMM)

❖ An Example of Viterbi in MEMMs

"Matt saw the *cat*"

	I or *N*	*V*	*D*
Matt	$p_N = .9, p_V = .1$	$p_N = .8, p_V = .2$	$p_N = .9, p_V = .1$
saw	$p_N = .2, p_V = .8$	$p_N = .7, p_V = .3$	$p_N = 1$
the	$p_D = 1$	$p_D = 1$	$p_D = 1$
cat	$p_N = .9, p_V = .1$	$p_N = .95, p_V = .05$	$p_N = 1$

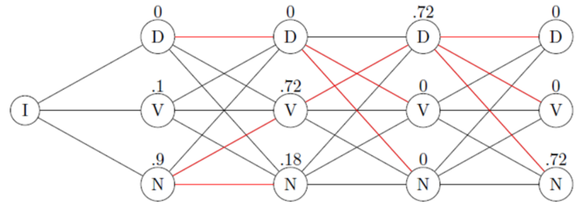


Solution: Maximum Entropy Markov Model (MEMM)

❖ An Example of Viterbi in MEMMs

"Matt saw the *cat*"

	I or *N*	*V*	*D*
Matt	$p_N = .9, p_V = .1$	$p_N = .8, p_V = .2$	$p_N = .9, p_V = .1$
saw	$p_N = .2, p_V = .8$	$p_N = .7, p_V = .3$	$p_N = 1$
the	$p_D = 1$	$p_D = 1$	$p_D = 1$
cat	$p_N = .9, p_V = .1$	$p_N = .95, p_V = .05$	$p_N = 1$

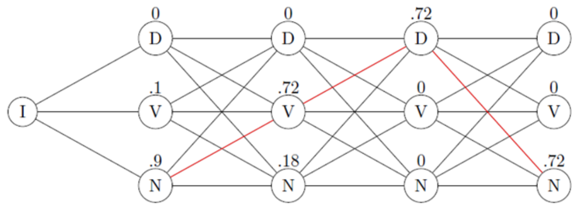


Solution: Maximum Entropy Markov Model (MEMM)

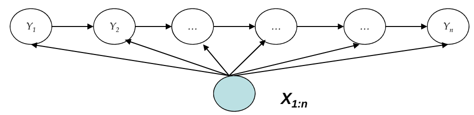
❖ An Example of Viterbi in MEMMs

"Matt saw the cat"

	I or *N*	*V*	*D*
Matt	$p_N = .9, p_V = .1$	$p_N = .8, p_V = .2$	$p_N = .9, p_V = .1$
saw	$p_N = .2, p_V = .8$	$p_N = .7, p_V = .3$	$p_N = 1$
the	$p_D = 1$	$p_D = 1$	$p_D = 1$
cat	$p_N = .9, p_V = .1$	$p_N = .95, p_V = .05$	$p_N = 1$



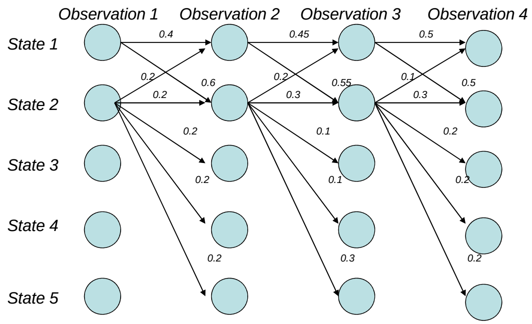
Solution: Maximum Entropy Markov Model (MEMM)



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

- ❖ **Models dependence between each state and the full observation sequence explicitly**
  - ➢ More expressive than HMMs
- ❖ **Discriminative model**
  - ➢ Completely ignores modeling P(**X**): saves modeling effort
  - ➢ Learning objective function consistent with predictive function: P(**Y**|**X**)

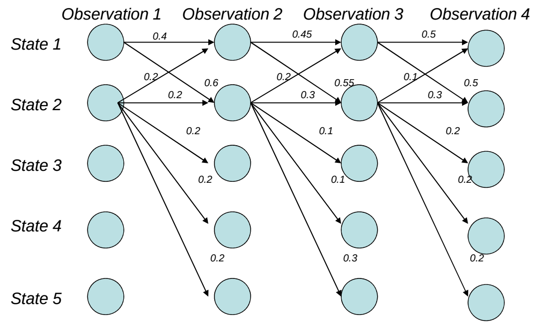
MEMM: Label bias problem



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*

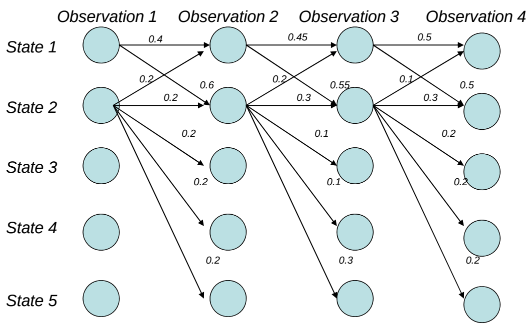
MEMM: Label bias problem



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

- *0.4 x 0.45 x 0.5 = 0.09*

MEMM: Label bias problem



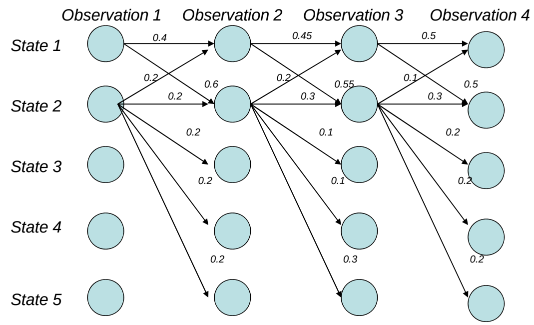
Probability of path 2->2->2->2 :

- *0.2 X 0.3 X 0.3 = 0.018*

Other paths:

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

MEMM: Label bias problem



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

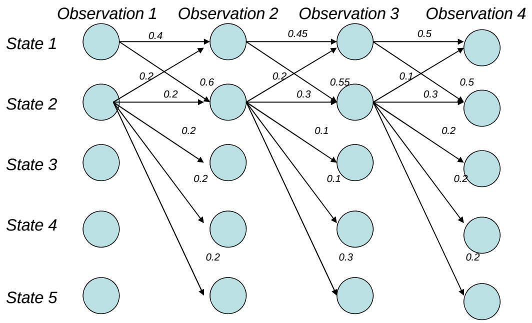
- *0.6 X 0.2 X 0.5 = 0.06*

Other paths:

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

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

MEMM: Label bias problem



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

- 0.4 X 0.55 X 0.3 = 0.066

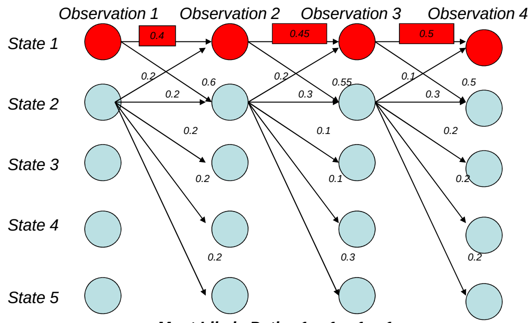
Other paths:

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

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

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

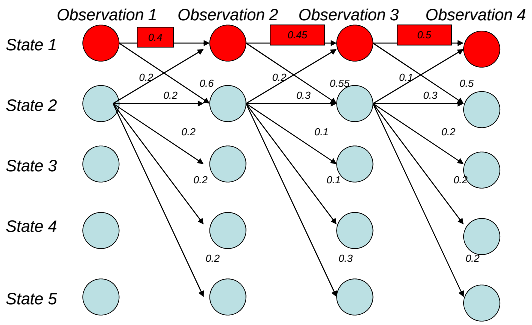
MEMM: Label bias problem



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?***

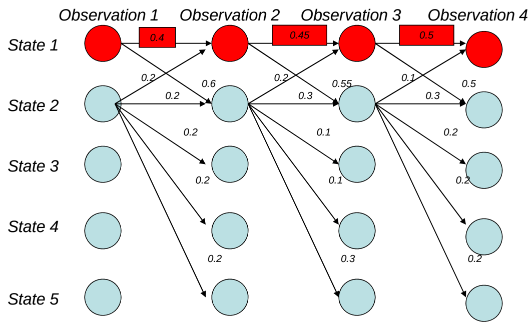
MEMM: Label bias problem



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*

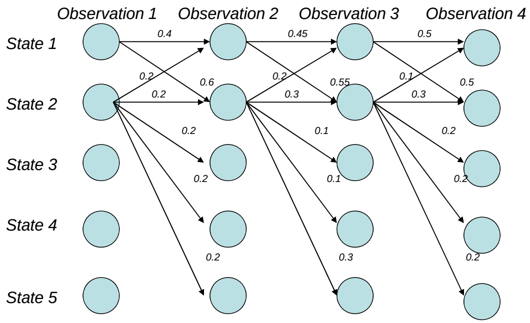
MEMM: Label bias problem



Label bias problem in MEMM:

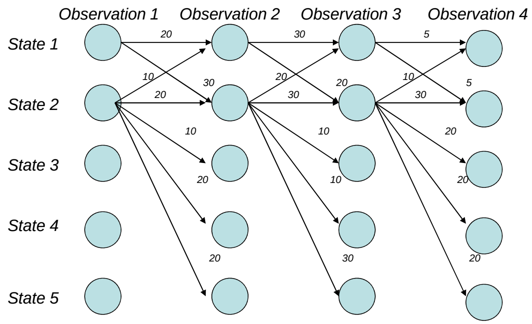
- *Preference of states with lower number of transitions over others*

Solution: Do not normalize probabilities locally



From local probabilities ….

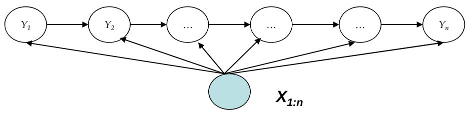
Solution: Do not normalize probabilities locally



From local probabilities to local potentials

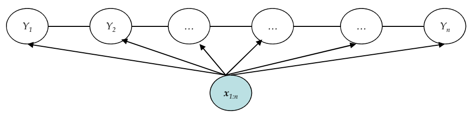
- *States with lower transitions do not have an unfair advantage!*

From MEMM ….



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

From MEMM to CRF

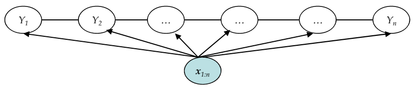


$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n})}\prod_{i=1}^{n}\phi(y_i,y_{i-1},\mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n})}\prod_{i=1}^{n}\exp(\mathbf{w}^T\mathbf{f}(y_i,y_{i-1},\mathbf{x}_{1:n}))$$

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

Conditional Random Fields

❖ **General parametric form:**



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

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

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

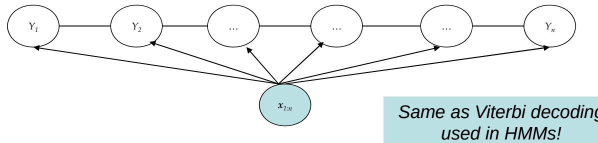
CRFs: Inference

❖ **Given CRF parameters λ and μ, find the y* that maximizes P(y|x)**

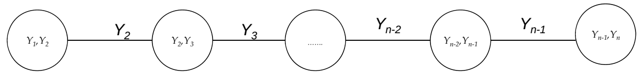
$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \exp(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x})))$$

➢ 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 $\{(x_d, y_d)\}_{d=1}^N$, find λ*, μ* such that**

$$\lambda*, \mu* = \arg\max_{\lambda,\mu} L(\lambda,\mu) = \arg\max_{\lambda,\mu}\prod_{d=1}^{N} P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu)$$

$$= \arg\max_{\lambda,\mu}\prod_{d=1}^{N}\frac{1}{Z(\mathbf{x}_d)}\exp(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d)))$$

$$= \arg\max_{\lambda,\mu}\sum_{d=1}^{N}(\sum_{i=1}^{n}(\lambda^T \mathbf{f}(y_{d,i}, y_{d,i-1}, \mathbf{x}_d) + \mu^T \mathbf{g}(y_{d,i}, \mathbf{x}_d)) - \log Z(\mathbf{x}_d))$$

❖ **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

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

❖ **Computing the model expectations:**

➢ Requires exponentially large number of summations: Is it intractable?

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

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

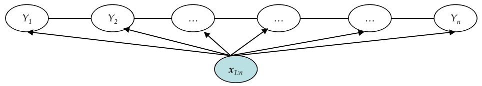
Expectation of f over the corresponding marginal probability of neighboring nodes!!

❖ **Tractable!**

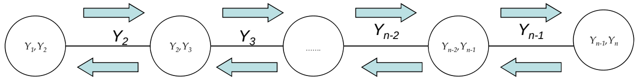
➢ Can compute marginals using the sum-product algorithm on the chain

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:**

Also called forward-backward algorithm

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

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(λ,μ):**

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

❖ **Learning can now be done using gradient ascent:**

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

CRF learning

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

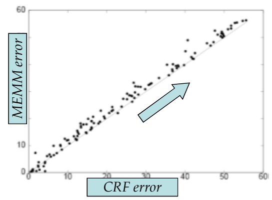
$$\lambda*, \mu* = \arg\max_{\lambda,\mu} \sum_{d=1}^{N} \log P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu) - \frac{1}{2\sigma^2}(\lambda^T\lambda + \mu^T\mu)$$

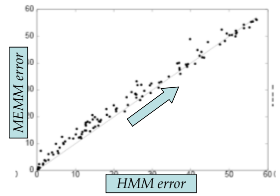
❖ **In practice, gradient ascent has very slow convergence**

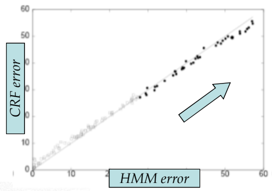
- Alternatives:
  - Conjugate Gradient method
  - Limited Memory Quasi-Newton Methods

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**

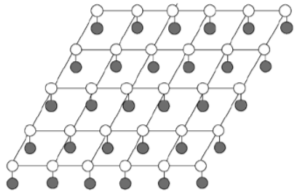
model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM+	4.81%	26.99%
CRF+	4.27%	23.76%

+Using spelling features

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

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 in the future



Summary

- **Conditional Random Fields are partially directed discriminative models**
- **They overcome the label bias problem of MEMMs 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