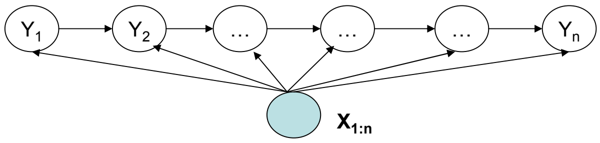
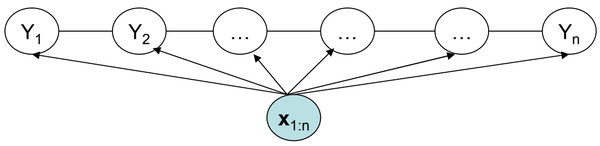
From MEMM ....



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

From MEMM to CRF



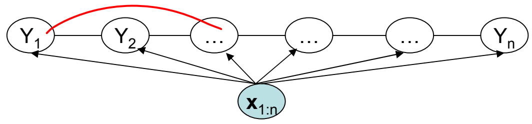
$$P(\mathbf{y}_{1:n}|\mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n})} \prod_{i=1}^{n} \phi(y_i, y_{i-1}, \mathbf{x}_{1:n}) = \frac{1}{Z(\mathbf{x}_{1:n}, \mathbf{w})} \prod_{i=1}^{n} \exp(\mathbf{w}^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_{1:n}))$$

- CRF is a partially directed model
  - Discriminative model 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:

$\psi(y_i, y_{i-1}, x)$



$$
\begin{aligned}
P(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x},\lambda,\mu)} \exp\Big(\sum_{i=1}^{n}\big(\sum_k \lambda_k f_k(y_i, y_{i-1}, \mathbf{x}) + \sum_l \mu_l g_l(y_i, \mathbf{x})\big)\Big) \\
&= \frac{1}{Z(\mathbf{x},\lambda,\mu)} \exp\Big(\sum_{i=1}^{n}\big(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x})\big)\Big)
\end{aligned}
$$

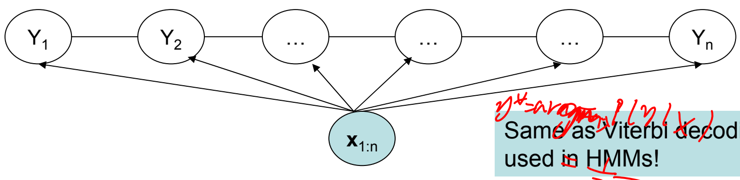
$$
\text{where } Z(\mathbf{x},\lambda,\mu) = \sum_{\mathbf{y}} \exp\Big(\sum_{i=1}^{n}\big(\lambda^T \mathbf{f}(y_i, y_{i-1}, \mathbf{x}) + \mu^T \mathbf{g}(y_i, \mathbf{x})\big)\Big)
$$

CRFs: Inference

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

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

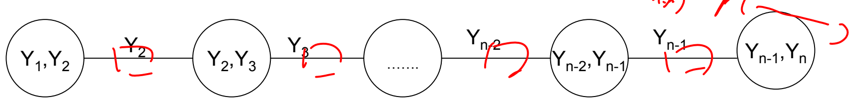
  - Can ignore Z($\mathbf{x}$) because it is not a function of $\mathbf{y}$
- Run the max-product algorithm on the junction-tree of CRF:



$y^* = \arg\max P(y|x)$

Same as Viterbi decoding used in HMMs!

$Z(x,\lambda,\mu)$ exp(



CRF learning

$\log Z = \log \sum_y (\tilde{p}(y|x))$

$\frac{d}{d\lambda} \log Z = \frac{1}{Z} \frac{dZ}{d\lambda} = \frac{1}{Z} \tilde{p} \frac{d}{d\lambda}$

$= \frac{1}{Z} \tilde{p} \sum f(y_i, y_{i-1})$

$= P(y|x) \sum f(\_)$

$\frac{d}{d\lambda} \exp x = \exp x \frac{dx}{d\lambda}$

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

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

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

log $\Rightarrow$

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

- 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

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

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

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

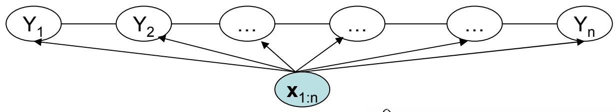
$$= \sum_{i=1}^{n} \sum_{y_i, y_{i-1}} \mathbf{f}(y_i, y_{i-1}, \mathbf{x}_d) P(y_i, y_{i-1}|\mathbf{x}_d)$$

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

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

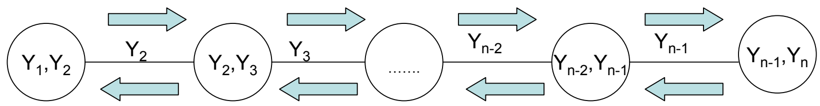
CRF learning

- Computing marginals using junction-tree calibration:



- Junction Tree Initialization:

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



- After calibration:

Also called forward-backward algorithm

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

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 $\nabla_\lambda L(\lambda,\mu)$:

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

- 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

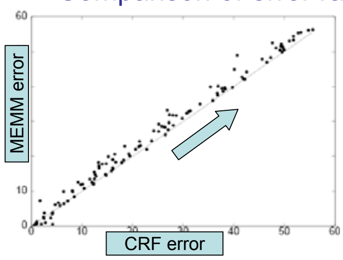
- In practice, we use a Gaussian Regularizer for the parameter vector to improve generalizability

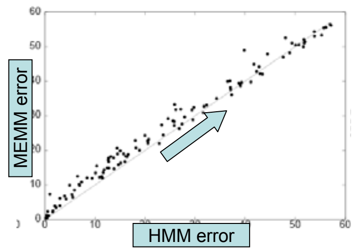
$$\lambda*, \mu* \quad = \quad \arg\max_{\lambda,\mu} \sum_{d=1}^{N} \log P(\mathbf{y}_d|\mathbf{x}_d, \lambda, \mu) - \frac{1}{2\sigma^2}(\lambda^T\lambda + \mu^T\mu)$$

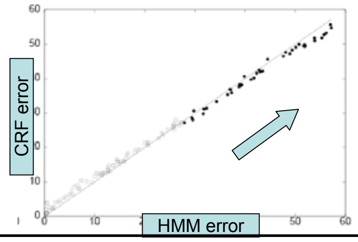
- In practice, gradient ascent has very slow convergence
  - Alternatives:
    - Conjugate Gradient method
    - Limited Memory Quasi-Newton Methods

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

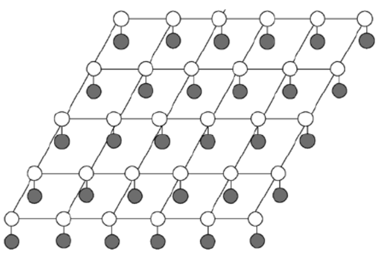
model	*error*	*oov error*
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM$^+$	4.81%	26.99%
CRF$^+$	4.27%	23.76%

$^+$Using spelling features

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

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