# Neural networks

## Training CRFs - loss function

# LINEAR CHAIN CRF

**Topics:** reminder of notation

- Then we have:

$$p(\mathbf{y}|\mathbf{X}) = \exp\left(\sum_{k=1}^{K} a_u(y_k) + \sum_{k=1}^{K-1} a_p(y_k, y_{k+1})\right)/Z(\mathbf{X})$$

where

$$Z(\mathbf{X}) = \sum_{y'_1}\sum_{y'_2}\cdots\sum_{y'_K}\exp\left(\sum_{k=1}^{K} a_u(y'_k) + \sum_{k=1}^{K-1} a_p(y'_k, y'_{k+1})\right)$$

- Two types of (log-)factors:
  - unary: $a_u(y_k) = a^{(L+1,0)}(\mathbf{x}_k)_{y_k} + 1_{k>1}\; a^{(L+1,-1)}(\mathbf{x}_{k-1})_{y_k} + 1_{k<K}\; a^{(L+1,+1)}(\mathbf{x}_{k+1})_{y_k}$
  - pairwise: $a_p(y_k, y_{k+1}) = 1_{1\leq k<K}\; V_{y_k, y_{k+1}}$

# MACHINE LEARNING

**Topics:** empirical risk minimization, regularization

- Empirical risk minimization
  - framework to design learning algorithms

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_t l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)}) + \lambda \Omega(\boldsymbol{\theta})$$

  - $l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)})$ is a loss function
  - $\Omega(\boldsymbol{\theta})$ is a regularizer (penalizes certain values of $\boldsymbol{\theta}$)
- Learning is cast as optimization
  - ideally, we'd optimize classification error, but it's not smooth
  - loss function is a surrogate for what we truly should optimize

# MACHINE LEARNING

**Topics:** stochastic gradient descent (SGD)

- Algorithm that performs updates after each example
  - initialize $\boldsymbol{\theta}$
  - for N iterations
    - for each training example $(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})$
      - ✓ $\Delta = -\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)}) - \lambda \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$
      - ✓ $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\, \Delta$

training epoch
=
iteration over **all** examples

- To apply this algorithm to a CRF, we need
  - the loss function $l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)})$
  - a procedure to compute the parameter gradients $\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)})$
  - the regularizer $\Omega(\boldsymbol{\theta})$ (and the gradient $\nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$ )
  - initialization method

# LOSS FUNCTION

**Topics:** loss function for sequential classification with CRF

- CRF estimates $p(\mathbf{y}|\mathbf{X})$
  - we could maximize the probabilities of $\mathbf{y}^{(t)}$ given $\mathbf{X}^{(t)}$ in the training set

- To frame as minimization, we minimize the negative log-likelihood

$$l(\mathbf{f}(\mathbf{X}), \mathbf{y}) = -\log p(\mathbf{y}|\mathbf{X})$$

  - unlike for non-sequential classification, we never explicitly compute the value of $p(\mathbf{y}|\mathbf{X})$ for all values of $\mathbf{y}$

# Neural networks

## Training CRFs - unary log-factor gradient

# MACHINE LEARNING

**Topics:** stochastic gradient descent (SGD)

- Algorithm that performs updates after each example
  - initialize $\boldsymbol{\theta}$
  - for N iterations
    - for each training example $(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})$
      - ✓ $\Delta = -\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)}) - \lambda \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$
      - ✓ $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\, \Delta$

training epoch
=
iteration over **all** examples

- To apply this algorithm to a CRF, we need
  - the loss function $l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)})$
  - a procedure to compute the parameter gradients $\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)})$
  - the regularizer $\Omega(\boldsymbol{\theta})$ (and the gradient $\nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$ )
  - initialization method

# LOSS FUNCTION

**Topics:** loss function for sequential classification with CRF

- CRF estimates $p(\mathbf{y}|\mathbf{X})$
  - we could maximize the probabilities of $\mathbf{y}^{(t)}$ given $\mathbf{X}^{(t)}$ in the training set

- To frame as minimization, we minimize the negative log-likelihood

$$l(\mathbf{f}(\mathbf{X}), \mathbf{y}) = -\log p(\mathbf{y}|\mathbf{X})$$

  - unlike for non-sequential classification, we never explicitly compute the value of $p(\mathbf{y}|\mathbf{X})$ for all values of $\mathbf{y}$

# PARAMETER GRADIENTS

**Topics:** loss gradient at unary log-factors

- Partial derivative wrt $a_u(y_k')$:

$$\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_u(y'_k)} \quad = \quad -(1_{y_k=y'_k} - p(y'_k|\mathbf{X}))$$

- Gradient for each unary (log-)factors:

$$\begin{aligned}
\nabla_{\mathbf{a}^{(L+1,0)}(\mathbf{x}_k)} - \log p(\mathbf{y}|\mathbf{X}) \quad &= \quad -(\mathbf{e}(y_k) - \mathbf{p}(y_k|\mathbf{X})) \\
\nabla_{\mathbf{a}^{(L+1,-1)}(\mathbf{x}_{k-1})} - \log p(\mathbf{y}|\mathbf{X}) \quad &= \quad -1_{k>1}\,(\mathbf{e}(y_k) - \mathbf{p}(y_k|\mathbf{X})) \\
\nabla_{\mathbf{a}^{(L+1,+1)}(\mathbf{x}_{k+1})} - \log p(\mathbf{y}|\mathbf{X}) \quad &= \quad -1_{k<K}\,(\mathbf{e}(y_k) - \underbrace{\mathbf{p}(y_k|\mathbf{X})}_{\text{vector of all marginal probabilities}})
\end{aligned}$$

$$\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_u(y'_k)} \quad = \quad \frac{\partial}{\partial a_u(y'_k)} - \left( \left( \sum_{k=1}^{K} a_u(y_k) + \sum_{k=1}^{K-1} a_p(y_k, y_{k+1}) \right) - \log Z(\mathbf{X}) \right)$$

$$
\begin{aligned}
\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_u(y'_k)} &= \frac{\partial}{\partial a_u(y'_k)} - \left( \left( \sum_{k=1}^{K} a_u(y_k) + \sum_{k=1}^{K-1} a_p(y_k, y_{k+1}) \right) - \log Z(\mathbf{X}) \right) \\
&= - \left( 1_{y_k = y'_k} - \frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) \right)
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_u(y'_k)} &= \frac{\partial}{\partial a_u(y'_k)} - \left( \left( \sum_{k=1}^{K} a_u(y_k) + \sum_{k=1}^{K-1} a_p(y_k, y_{k+1}) \right) - \log Z(\mathbf{X}) \right) \\
&= - \left( 1_{y_k = y'_k} - \frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) \right)
\end{aligned}
$$

$$
\frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) = \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} Z(\mathbf{X})
$$

$$\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_u(y'_k)} = \frac{\partial}{\partial a_u(y'_k)} - \left( \left( \sum_{k=1}^{K} a_u(y_k) + \sum_{k=1}^{K-1} a_p(y_k, y_{k+1}) \right) - \log Z(\mathbf{X}) \right)$$

$$= - \left( 1_{y_k = y'_k} - \frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) \right)$$

$$\frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) = \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} Z(\mathbf{X})$$

$$= \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} \exp \left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right)$$

$$\begin{aligned}
\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_u(y'_k)} &= \frac{\partial}{\partial a_u(y'_k)} - \left( \left( \sum_{k=1}^{K} a_u(y_k) + \sum_{k=1}^{K-1} a_p(y_k, y_{k+1}) \right) - \log Z(\mathbf{X}) \right) \\
&= - \left( 1_{y_k = y'_k} - \frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) \right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) &= \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} Z(\mathbf{X}) \\
&= \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} \exp\left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right) \\
&= \frac{1}{Z(\mathbf{X})} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} \frac{\partial}{\partial a_u(y'_k)} \exp\left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right)
\end{aligned}$$

$$\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_u(y'_k)} = \frac{\partial}{\partial a_u(y'_k)} - \left( \left( \sum_{k=1}^{K} a_u(y_k) + \sum_{k=1}^{K-1} a_p(y_k, y_{k+1}) \right) - \log Z(\mathbf{X}) \right)$$
$$= -\left( 1_{y_k = y'_k} - \frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) \right)$$

$$\frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) = \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} Z(\mathbf{X})$$
$$= \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} \exp\left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right)$$
$$= \frac{1}{Z(\mathbf{X})} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} \frac{\partial}{\partial a_u(y'_k)} \exp\left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right)$$
$$= \frac{1}{Z(\mathbf{X})} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} 1_{y'_k = y''_k} \exp\left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right)$$

$$\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_u(y'_k)} = \frac{\partial}{\partial a_u(y'_k)} - \left( \left( \sum_{k=1}^{K} a_u(y_k) + \sum_{k=1}^{K-1} a_p(y_k, y_{k+1}) \right) - \log Z(\mathbf{X}) \right)$$
$$= -\left( 1_{y_k = y'_k} - \frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) \right)$$

$$\frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) = \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} Z(\mathbf{X})$$
$$= \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} \exp \left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right)$$
$$= \frac{1}{Z(\mathbf{X})} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} \frac{\partial}{\partial a_u(y'_k)} \exp \left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right)$$
$$= \frac{1}{Z(\mathbf{X})} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} 1_{y'_k = y''_k} \exp \left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right)$$
$$= \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} 1_{y'_k = y''_k}\ p(y''_1, \ldots, y''_K|\mathbf{X})$$

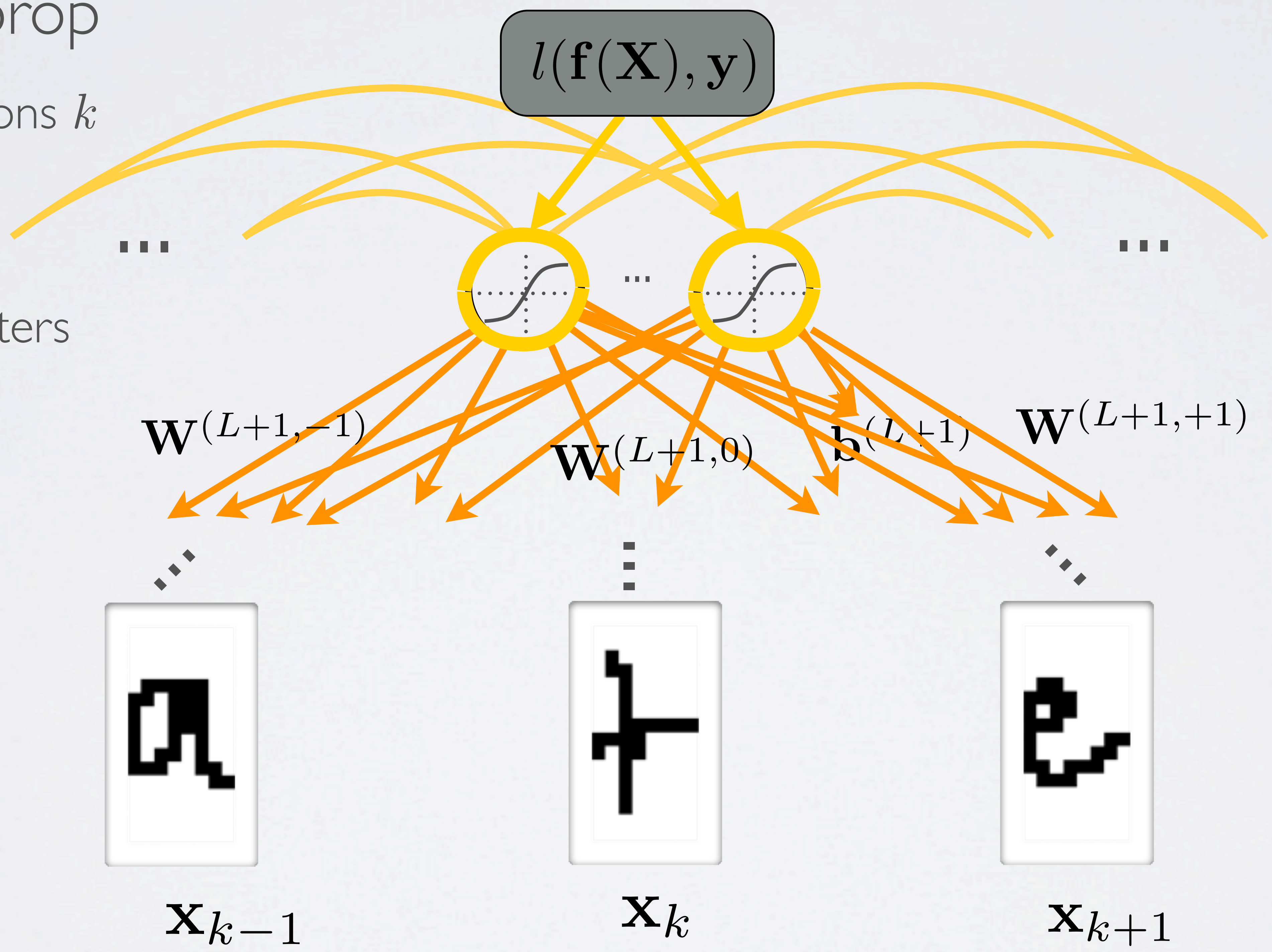
$$\begin{aligned}
\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_u(y'_k)} &= \frac{\partial}{\partial a_u(y'_k)} - \left( \left( \sum_{k=1}^{K} a_u(y_k) + \sum_{k=1}^{K-1} a_p(y_k, y_{k+1}) \right) - \log Z(\mathbf{X}) \right) \\
&= - \left( 1_{y_k = y'_k} - \frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) \right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial a_u(y'_k)} \log Z(\mathbf{X}) &= \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} Z(\mathbf{X}) \\
&= \frac{1}{Z(\mathbf{X})} \frac{\partial}{\partial a_u(y'_k)} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} \exp\left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right) \\
&= \frac{1}{Z(\mathbf{X})} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} \frac{\partial}{\partial a_u(y'_k)} \exp\left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right) \\
&= \frac{1}{Z(\mathbf{X})} \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} 1_{y'_k = y''_k} \exp\left( \sum_{k=1}^{K} a_u(y''_k) + \sum_{k=1}^{K-1} a_p(y''_k, y''_{k+1}) \right) \\
&= \sum_{y''_1} \sum_{y''_2} \cdots \sum_{y''_K} 1_{y'_k = y''_k} \, p(y''_1, \ldots, y''_K | \mathbf{X}) \\
&= p(y'_k | \mathbf{X})
\end{aligned}$$

# PARAMETER GRADIENTS

**Topics:** loss gradient at unary log-factor parameters

- Use regular backprop
  - backprop at all positions $k$
  - accumulate all gradients, from every position, into parameters



# PARAMETER GRADIENTS

**Topics:** loss gradient at unary log-factor parameters

- For linear log-factors:
  - the log-factors are directly connected to the input:

$$\mathbf{a}^{(1,0)}(\mathbf{x}_k) = \mathbf{b}^{(1)} + \mathbf{W}^{(1,0)}\mathbf{x}_k$$

$$\mathbf{a}^{(1,-1)}(\mathbf{x}_k) = \mathbf{W}^{(1,-1)}\mathbf{x}_k$$

$$\mathbf{a}^{(1,+1)}(\mathbf{x}_k) = \mathbf{W}^{(1,+1)}\mathbf{x}_k$$

# PARAMETER GRADIENTS

**Topics:** loss gradient at unary log-factor parameters

- For linear log-factors:
  - the gradients are:

$$\nabla_{\mathbf{b}^{(1)}} - \log p(\mathbf{y}|\mathbf{X}) = \sum_{k=1}^{K} \left(\nabla_{\mathbf{a}^{(1,0)}(\mathbf{x}_k)} - \log p(\mathbf{y}|\mathbf{X})\right) \quad = \quad \sum_{k=1}^{K} -(\mathbf{e}(y_k) - \mathbf{p}(y_k|\mathbf{X}))$$

$$\nabla_{\mathbf{W}^{(1,0)}} - \log p(\mathbf{y}|\mathbf{X}) = \sum_{k=1}^{K} \left(\nabla_{\mathbf{a}^{(1,0)}(\mathbf{x}_k)} - \log p(\mathbf{y}|\mathbf{X})\right) \ \mathbf{x}_k^\top \quad = \quad \sum_{k=1}^{K} -(\mathbf{e}(y_k) - \mathbf{p}(y_k|\mathbf{X}))\, \mathbf{x}_k^\top$$

$$\nabla_{\mathbf{W}^{(1,-1)}} - \log p(\mathbf{y}|\mathbf{X}) = \sum_{k=2}^{K} \left(\nabla_{\mathbf{a}^{(1,-1)}(\mathbf{x}_k)} - \log p(\mathbf{y}|\mathbf{X})\right) \ \mathbf{x}_{k-1}^\top \quad = \quad \sum_{k=2}^{K} -\left(\mathbf{e}(y_k) - \mathbf{p}(y_k|\mathbf{X})\right) \ \mathbf{x}_{k-1}^\top$$

$$\nabla_{\mathbf{W}^{(1,+1)}} - \log p(\mathbf{y}|\mathbf{X}) = \sum_{k=1}^{K-1} \left(\nabla_{\mathbf{a}^{(1,+1)}(\mathbf{x}_k)} - \log p(\mathbf{y}|\mathbf{X})\right) \ \mathbf{x}_{k+1}^\top \quad = \quad \sum_{k=1}^{K-1} -\left(\mathbf{e}(y_k) - \mathbf{p}(y_k|\mathbf{X})\right) \ \mathbf{x}_{k+1}^\top$$

# Neural networks

## Training CRFs - pairwise log-factor gradient

# MACHINE LEARNING

**Topics:** stochastic gradient descent (SGD)

- Algorithm that performs updates after each example
  - initialize $\boldsymbol{\theta}$
  - for N iterations
    - for each training example $(\mathbf{X}^{(t)}, \mathbf{y}^{(t)})$
      - ✓ $\Delta = -\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)}) - \lambda \nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$
      - ✓ $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\, \Delta$

training epoch = iteration over **all** examples

- To apply this algorithm to a CRF, we need
  - the loss function $l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)})$
  - a procedure to compute the parameter gradients $\nabla_{\boldsymbol{\theta}} l(\mathbf{f}(\mathbf{X}^{(t)}; \boldsymbol{\theta}), \mathbf{y}^{(t)})$
  - the regularizer $\Omega(\boldsymbol{\theta})$ (and the gradient $\nabla_{\boldsymbol{\theta}} \Omega(\boldsymbol{\theta})$ )
  - initialization method

# PARAMETER GRADIENTS

**Topics:** loss gradient at pairwise log-factor and parameters

- Partial derivative for log-factor:

$$\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial a_p(y'_k, y'_{k+1})} = -(1_{y_k=y'_k, y_{k+1}=y'_{k+1}} - p(y'_k, y'_{k+1}|\mathbf{X}))$$

- Partial derivative of log-factor parameters:

$$\frac{\partial - \log p(\mathbf{y}|\mathbf{X})}{\partial V_{y'_k, y'_{k+1}}} = \sum_{k=1}^{K-1} -(1_{y_k=y'_k, y_{k+1}=y'_{k+1}} - p(y'_k, y'_{k+1}|\mathbf{X}))$$

- Gradient of log-factor parameters

$$\begin{aligned}\nabla_{\mathbf{V}} - \log p(\mathbf{y}|\mathbf{X}) &= \sum_{k=1}^{K-1} -(\mathbf{e}(y_k)\,\mathbf{e}(y_{k+1})^\top - \overbrace{\mathbf{p}(y_k, y_{k+1}|\mathbf{X})}^{\text{matrix of all pairwise marginal probabilities}}) \\ &= -\left(\underbrace{\mathrm{freq}(y_k, y_{k+1})}_{\text{matrix of all pairwise label frequencies}} - \sum_{k=1}^{K-1} \mathbf{p}(y_k, y_{k+1}|\mathbf{X})\right)\end{aligned}$$

# REGULARIZATION

**Topics:** regularization

- For regularization, we can use the same regularizers as for a non-sequential neural network
  - add a regularizing term for all connection matrices
  - do not regularize the bias vectors
- We could scale $\lambda$ by the sequence size
- With the loss and regularization gradients, we have all the ingredients to perform stochastic gradient descent

# Neural networks

Training CRFs - discriminative vs. generative learning

# GENERATIVE VS. DISCRIMINATIVE

**Topics:** discriminative learning, generative learning

- In discriminative learning, we optimize the conditional likelihood $-\log p(\mathbf{y}|\mathbf{X})$
  - CRFs are discriminative
- In generative learning, we optimize the joint log-likelihood:

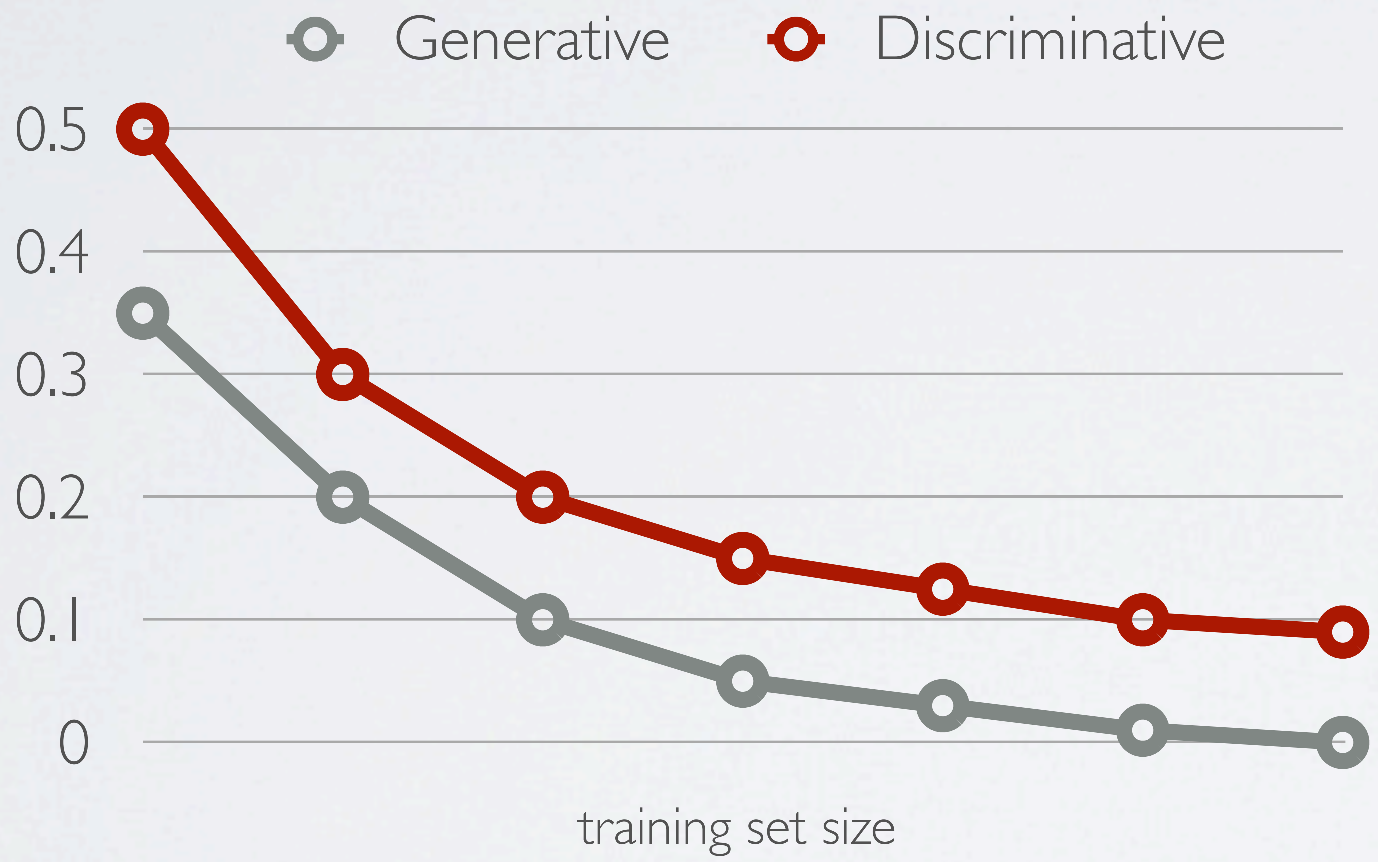
$$-\log p(\mathbf{y}, \mathbf{X}) = -\log \left(p(\mathbf{y}|\mathbf{X})p(\mathbf{X})\right) = -\log p(\mathbf{y}|\mathbf{X}) - \log p(\mathbf{X})$$

  - HMMs are usually trained generatively
  - $-\log p(\mathbf{X})$ is similar to a regularizer

# GENERATIVE VS. DISCRIMINATIVE

**Topics:** generative learning, discriminative learning

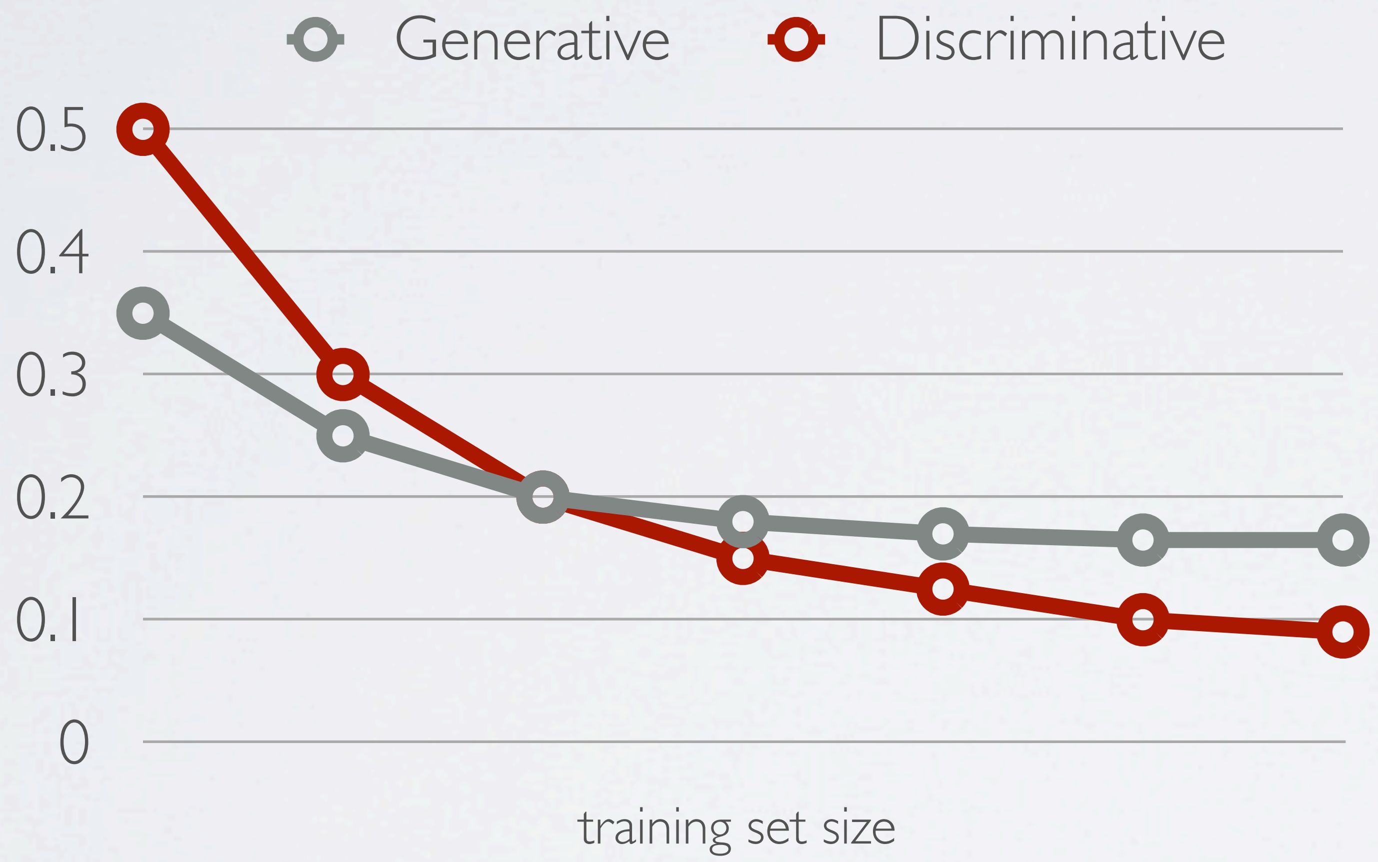
- It can be shown that:
  - if model is well-specified (i.e. is the true model) generative learning is better



# GENERATIVE VS. DISCRIMINATIVE

**Topics:** generative learning, discriminative learning

- It can be shown that:
  - if model is not well-specified (i.e. most of the time), it depends:



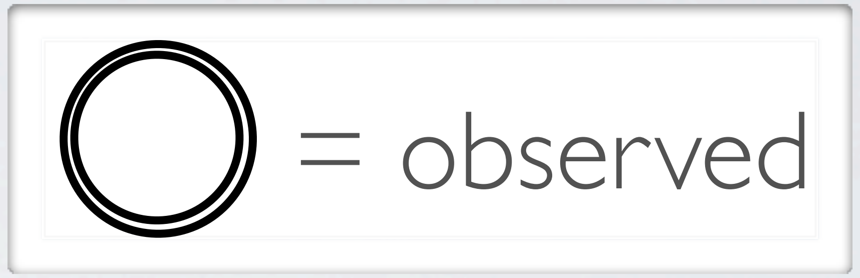

  - See these papers for more details:
    - On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes. Andrew Ng and Michael Jordan, 2001

# Neural networks

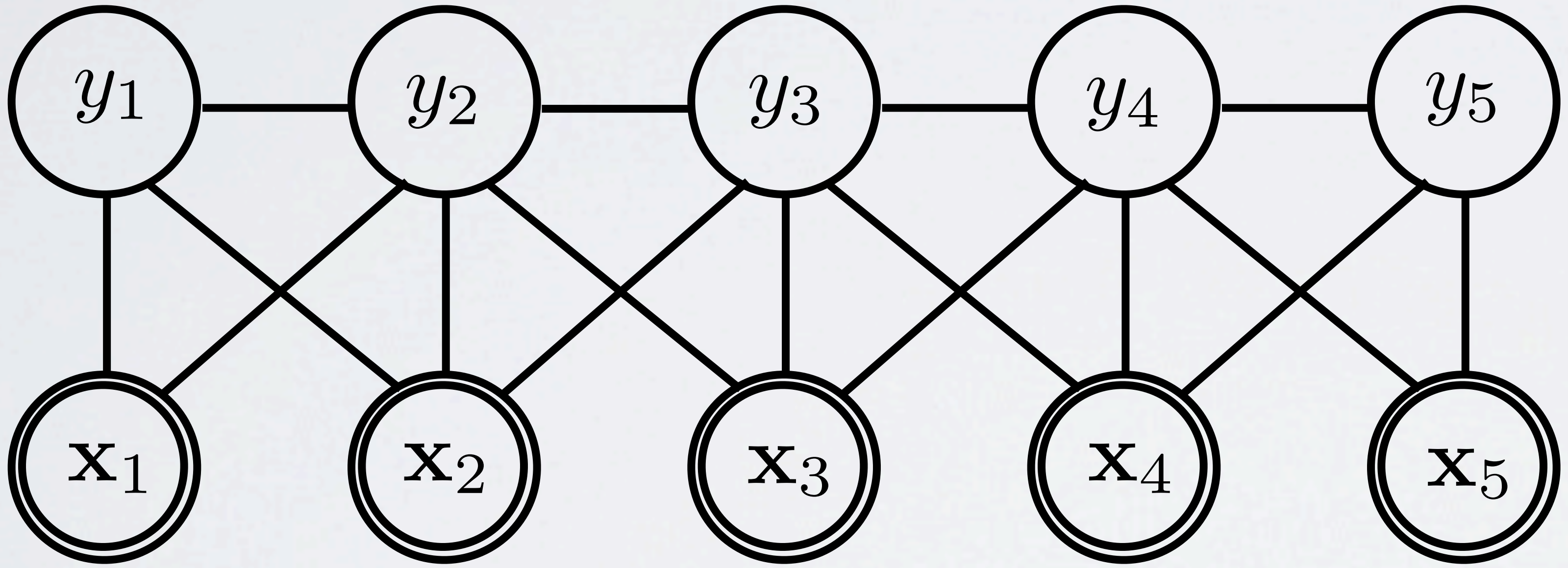
Training CRFs - maximum-entropy Markov model

# LINEAR CHAIN CRF

**Topics:** Markov network




- Illustration for $K=5$




$\exp(a^{(L+1,0)}(\mathbf{x}_3)_{y_3})$

$\exp(a^{(L+1,-1)}(\mathbf{x}_2)_{y_3})$

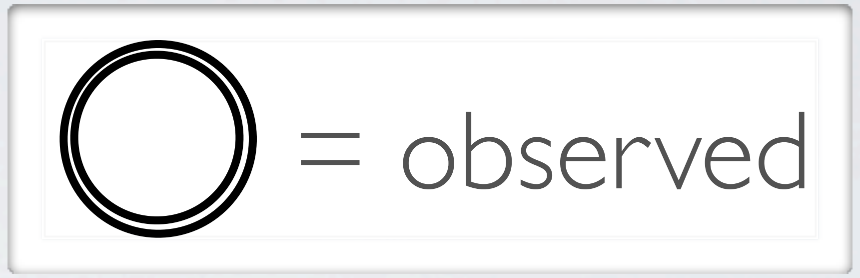
$\exp(a^{(L+1,+1)}(\mathbf{x}_4)_{y_3})$

$\exp(V_{y_3,y_4})$

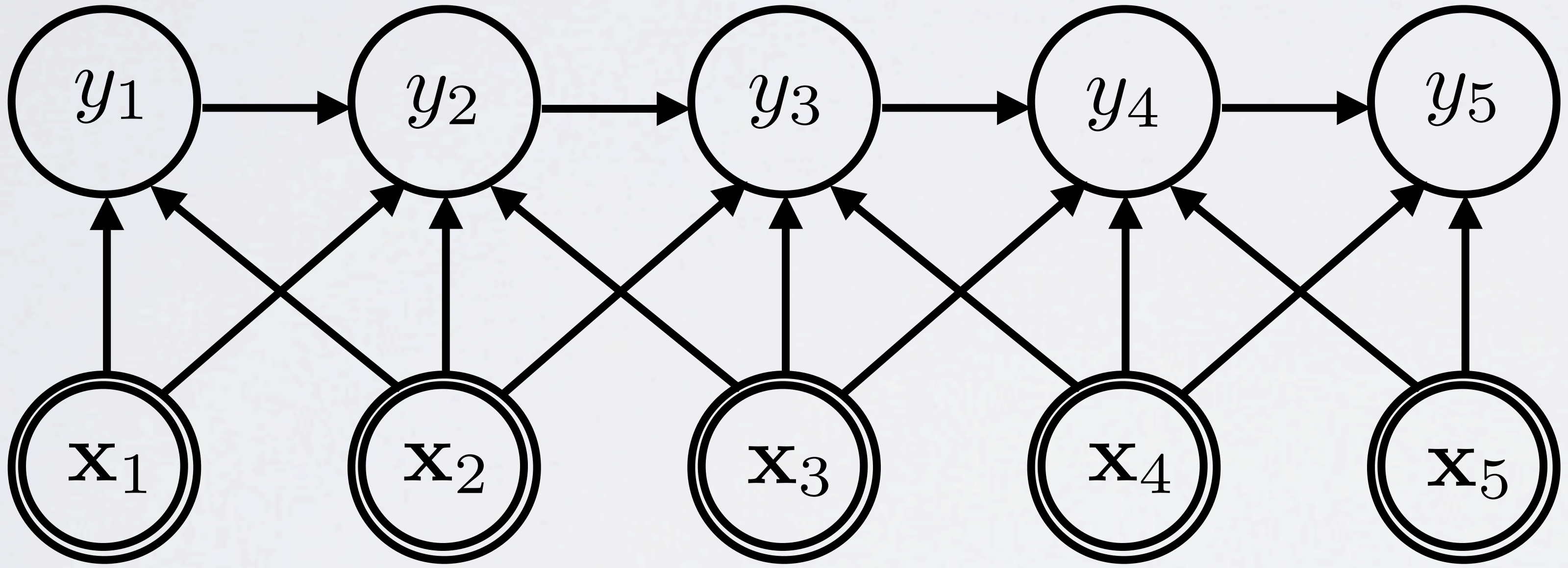
- Conditional random field are discriminatively trained, which should work better with more data
- Other alternative discriminatively trained sequence model?

# MAXIMUM-ENTROPY MARKOV MODEL

**Topics:** MEMM



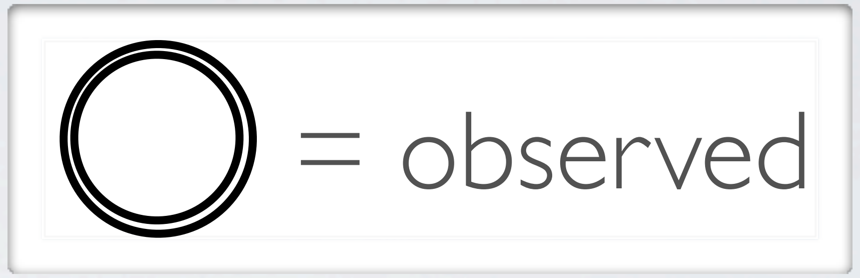

- MEMM is directed and discriminative:



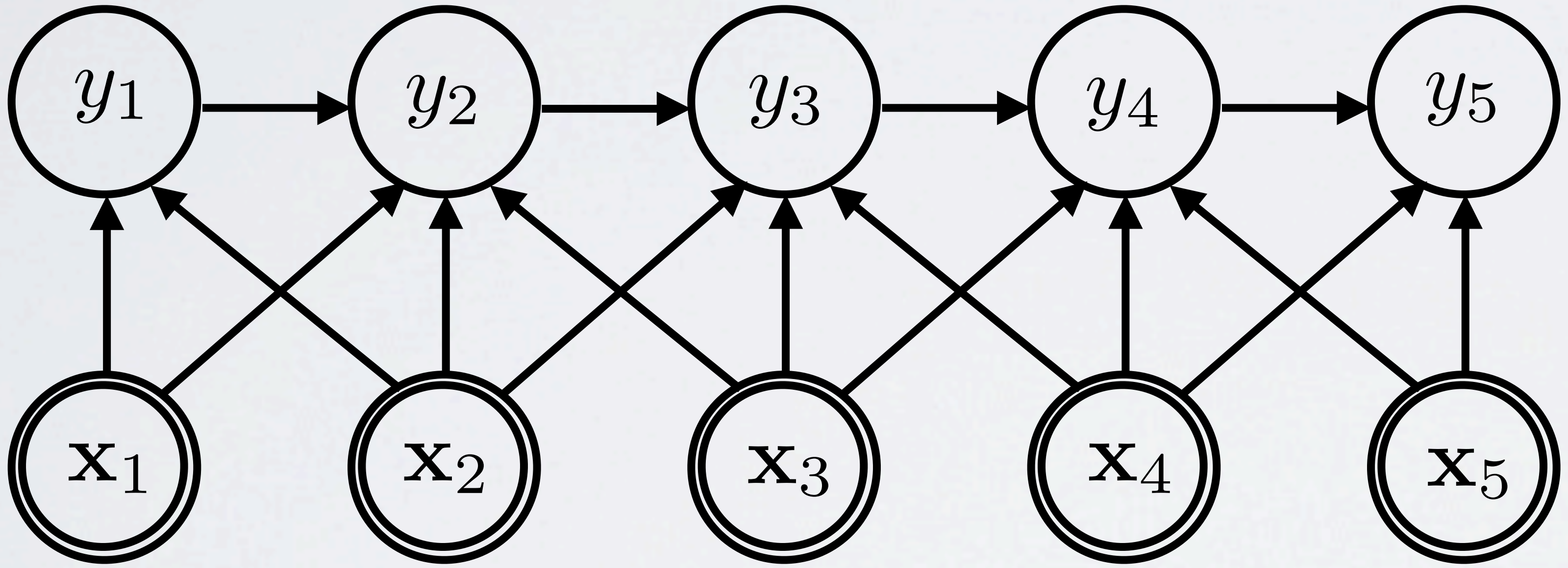

  - it's a Markov model where the transition probabilities are given by logistic regressors (or neural networks):
    - $p(\mathbf{y}|\mathbf{X}) = \prod_{k=1}^{K} p(y_k|y_{k-1}, \mathbf{X})$
    - $p(y_k|y_{k-1}, \mathbf{X}) = \frac{1}{Z(y_{k-1}, \mathbf{X})} \exp\left(a_u(y_k) + a_p(y_{k-1}, y_k)\right)$

# MAXIMUM-ENTROPY MARKOV MODEL

**Topics:** MEMM




- MEMM is directed and discriminative:



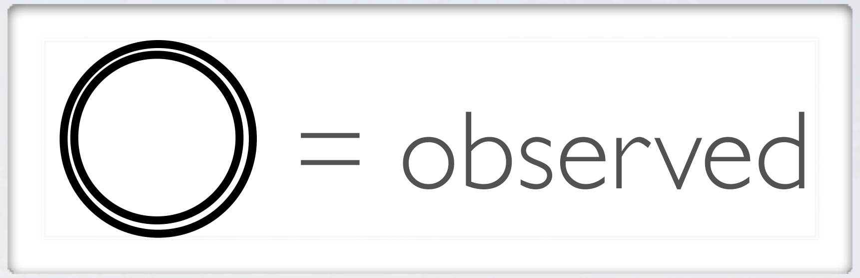

  - «label bias» problem: observations far away don't impact early predictions
    - example: $p(y_3|\mathbf{X}) = p(y_3|\mathbf{x}_1, \ldots, \mathbf{x}_4)$
    - observations after $\mathbf{x}_4$ do not change our decision about $y_3$!

# Neural networks

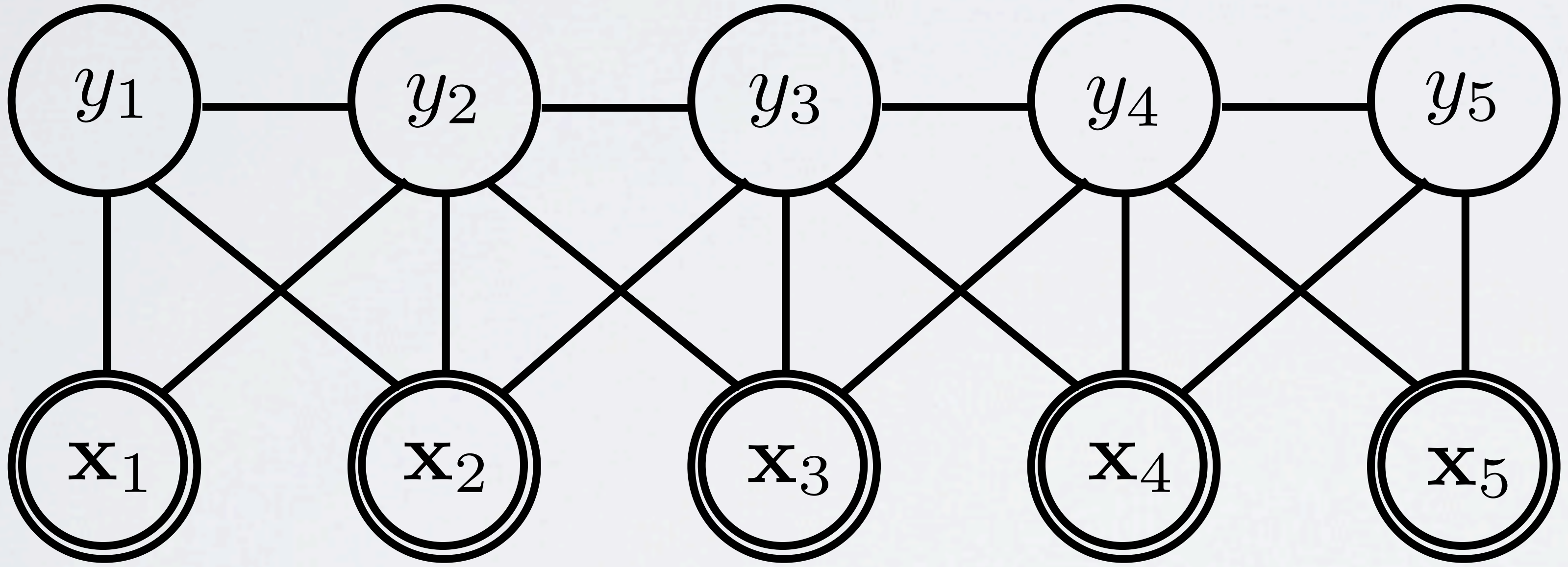
## Training CRFs - hidden Markov model

# LINEAR CHAIN CRF

**Topics:** Markov network




- Illustration for $K{=}5$




$$\exp(a^{(L+1,0)}(\mathbf{x}_3)_{y_3})$$

$$\exp(a^{(L+1,-1)}(\mathbf{x}_2)_{y_3})$$

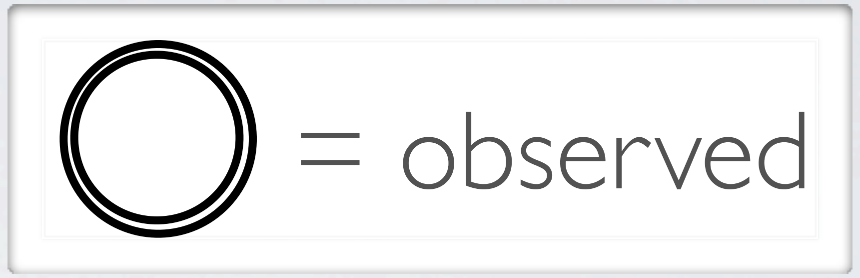
$$\exp(a^{(L+1,+1)}(\mathbf{x}_4)_{y_3})$$

$$\exp(V_{y_3,y_4})$$

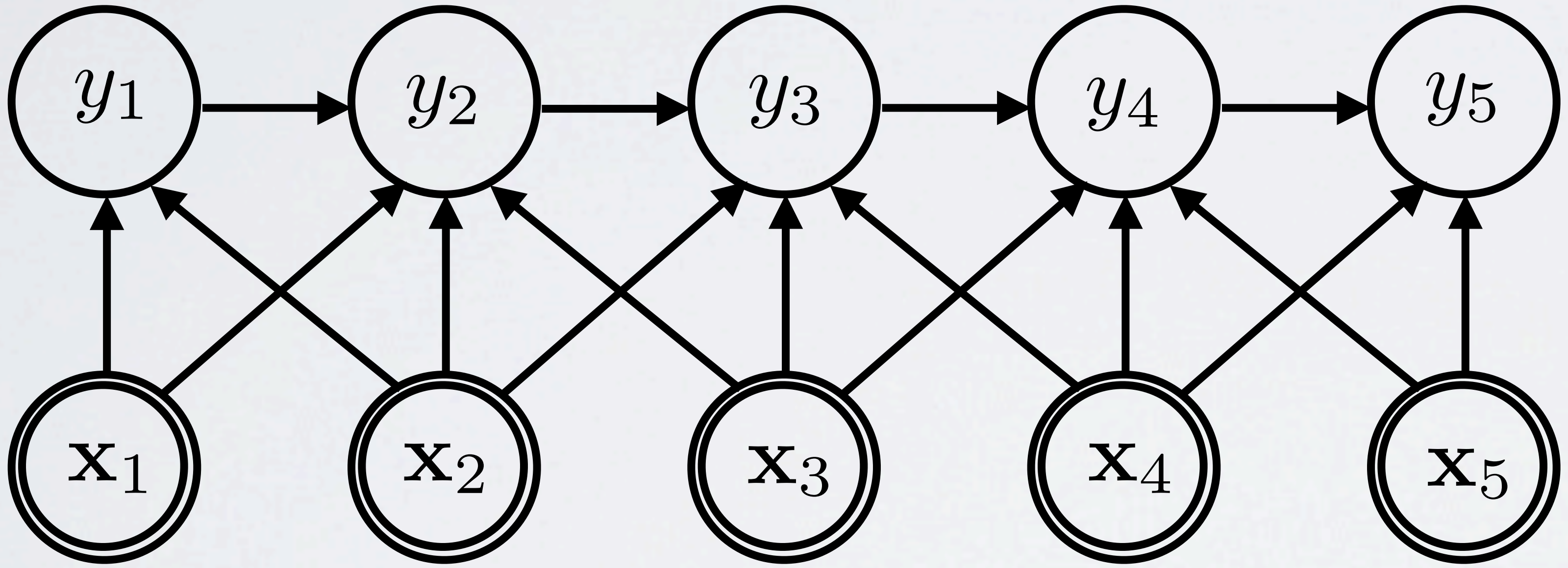
- Conditional random field are discriminatively trained, which should work better with more data
- Other alternative discriminatively trained sequence model?

# MAXIMUM-ENTROPY MARKOV MODEL

**Topics:** MEMM

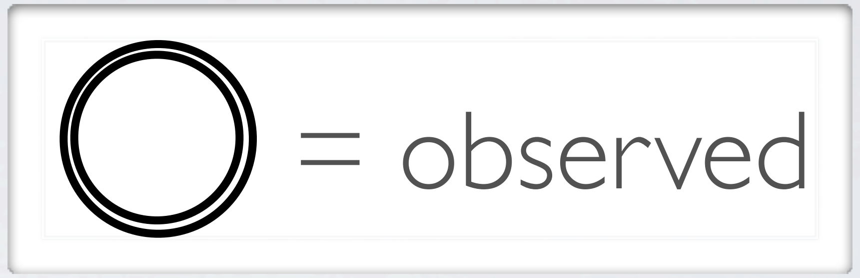



- MEMM is directed and discriminative:



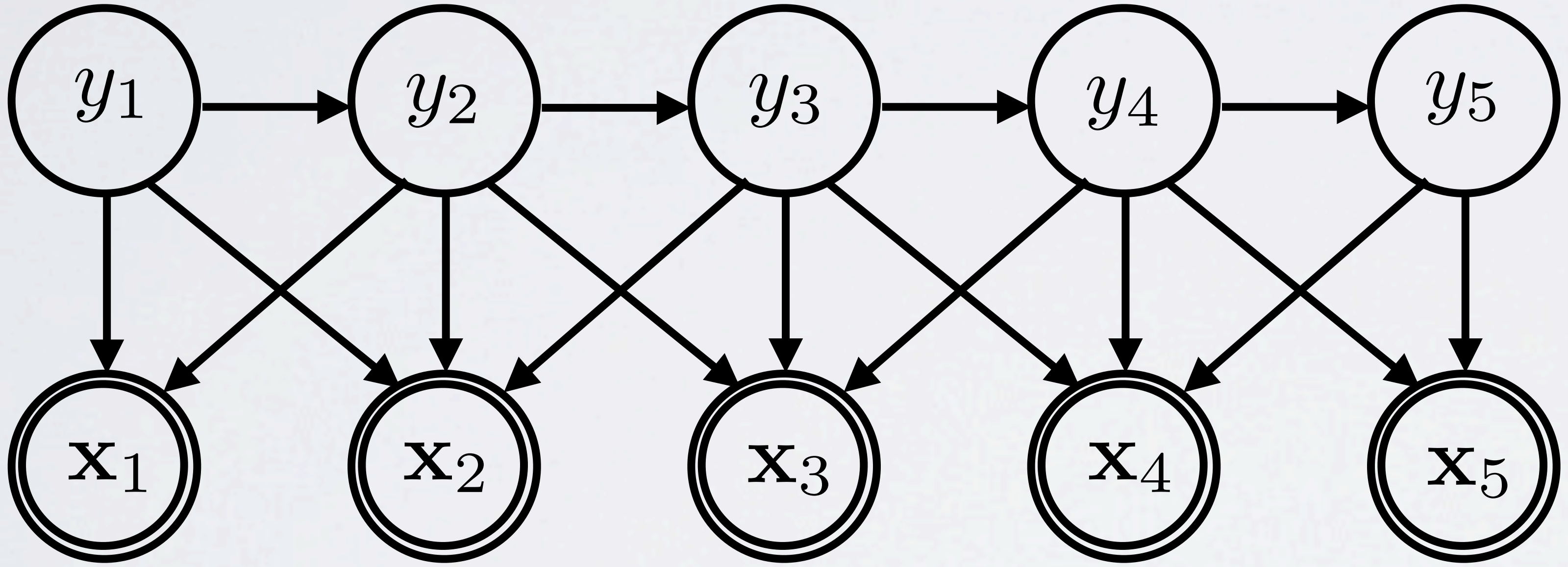

  - «label bias» problem: observations far away don't impact early predictions
    - example: $p(y_3|\mathbf{X}) = p(y_3|\mathbf{x}_1, \ldots, \mathbf{x}_4)$
    - observations after $\mathbf{x}_4$ do not change our decision about $y_3$!

# HIDDEN MARKOV MODEL




**Topics:** discriminative HMM

- HMMs can be trained discriminatively
  (i.e. minimize $-\log p(\mathbf{y}|\mathbf{X})$ )




  - used a lot in speech recognition (called «maximum mutual information training»)
  - we don’t have the same label bias anymore
  - however, optimization might be more complicated, since factors most correspond to normalized probabilities

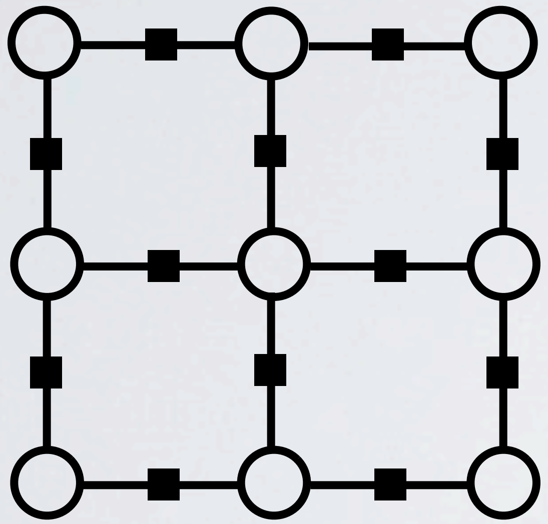
# Neural networks

## Training CRFs - general conditional random field

# GENERAL CRF

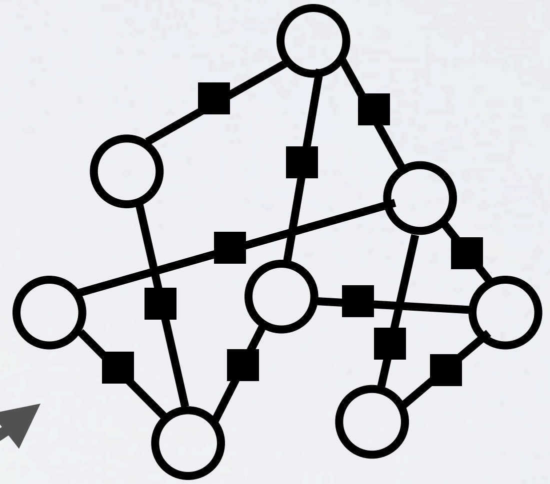
**Topics:** CRFs in general

- We don't have to restrict the CRF structure to linear chains



Grid structure
(pixels in image)

$$p(\mathbf{y}|\mathbf{X}) = \frac{1}{Z(\mathbf{X})} \prod_f \Psi_f(\mathbf{y}, \mathbf{X})$$



General pair-wise structure
(webpages sharing a link)

- We could also have $n$-ary factors, with $n>2$

# GENERAL CRF

**Topics:** CRFs in general

- Gradients in general CRFs always take the form:

$$\frac{\partial -\log p(\mathbf{y}^{(t)}|\mathbf{X}^{(t)})}{\partial \theta} = -\Bigg(\overbrace{\textstyle\sum_f \frac{\partial}{\partial \theta} \log \Psi_f(\mathbf{y}^{(t)}, \mathbf{X}^{(t)})}^{\text{make } y^{(t)} \text{ more likely}}$$

$$\underbrace{- \mathrm{E}_{\mathbf{y}} \left[ \textstyle\sum_f \frac{\partial}{\partial \theta} \log \Psi_f(\mathbf{y}, \mathbf{X}^{(t)}) \,\middle|\, \mathbf{X}^{(t)} \right]}_{\text{make everything less likely}} \Bigg)$$

- The expectation over $\mathbf{y}$ will often need to be approximated, using loopy belief propagation
  - it will often involve only a few of the $y_k$ variables

# (LOOPY) BELIEF PROPAGATION

**Topics:** CRFs in general

- Marginals can be approximated with:

$$p(y_k|\mathbf{X}) = \frac{\exp(\log\phi_f(y_k) + \sum_{f' \in \mathrm{Ne}(k) \setminus f} \log \mu_{f' \to k}(y_k))}{\sum_{y'_k} \exp(\log\phi_f(y'_k) + \sum_{f' \in \mathrm{Ne}(k) \setminus f} \log \mu_{f' \to k}(y'_k))}$$

- In general, an approximated marginal is computed by
  1. summing all the log-factors that involve only the $y_k$ variables of interest
  2. summing all the log-messages coming into the $y_k$ variables from other factors
  3. exponentiating
  4. renormalizing

# Neural networks

Training CRFs - pseudolikelihood

# GENERAL CRF

**Topics:** CRFs in general

- Gradients in general CRFs always take the form:

$$\frac{\partial -\log p(\mathbf{y}^{(t)}|\mathbf{X}^{(t)})}{\partial\theta} = -\Bigg(\overbrace{\textstyle\sum_f \frac{\partial}{\partial\theta}\log\Psi_f(\mathbf{y}^{(t)},\mathbf{X}^{(t)})}^{\text{make } y^{(t)} \text{ more likely}}$$

$$\underbrace{-\,\mathrm{E}_{\mathbf{y}}\left[\textstyle\sum_f \frac{\partial}{\partial\theta}\log\Psi_f(\mathbf{y},\mathbf{X}^{(t)})\,\middle|\mathbf{X}^{(t)}\right]}_{\text{make everything less likely}}\Bigg)$$

- The expectation over $\mathbf{y}$ will often need to be approximated, using loopy belief propagation
  - it will often involve only a few of the $y_k$ variables

# GENERAL CRF

**Topics:** pseudolikelihood

- Why not just change the loss function to a tractable one

$$-\sum_{k=1}^{K} \log p(y_k | y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_K, \mathbf{X})$$

  - predict, in turn, each $y_k$ not just from $\mathbf{X}$, but also all the other elements of $\mathbf{y}$
  - can compute the exact gradients
    - the probabilities only require normalizing $y_k$ individually, like in a regular softmax
    - each conditional often only depend on few variables (local Markov property)
  - however, often tends to work less well
  - we still need to compute $p(y_k|\mathbf{X})$ to do predictions anyways