Deep graphical models

JASON HARTFORD

Motivation

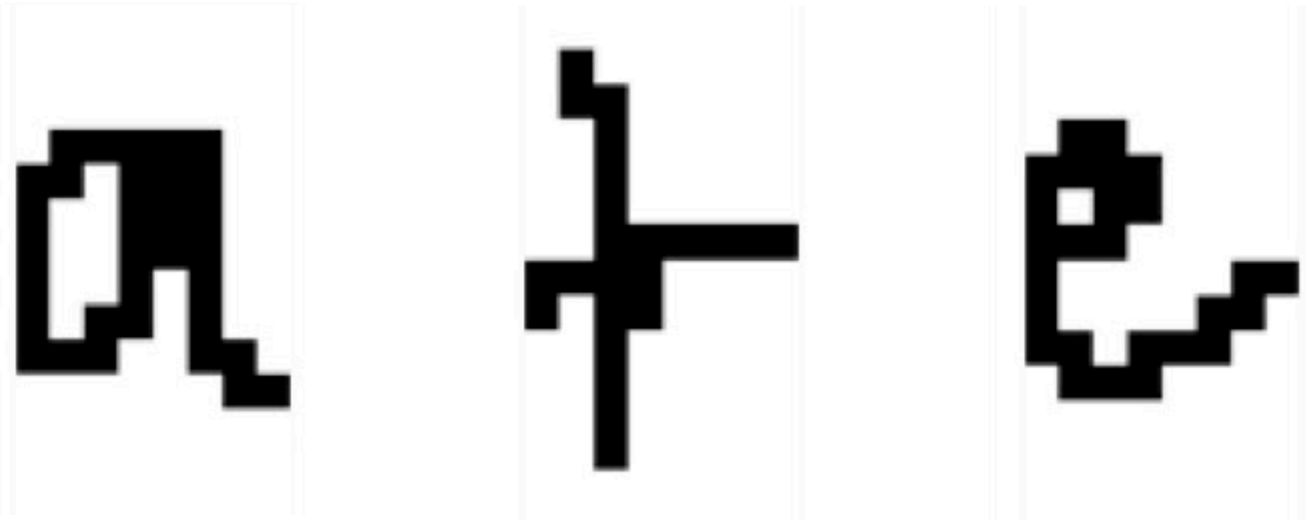
What character is this?

“t” or “+” of “f”?

Adapted from Hugo Larochelle’s slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

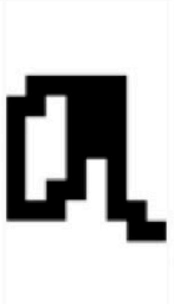
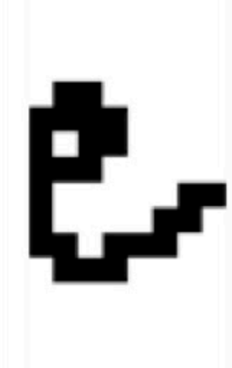
Motivation

What character is this?



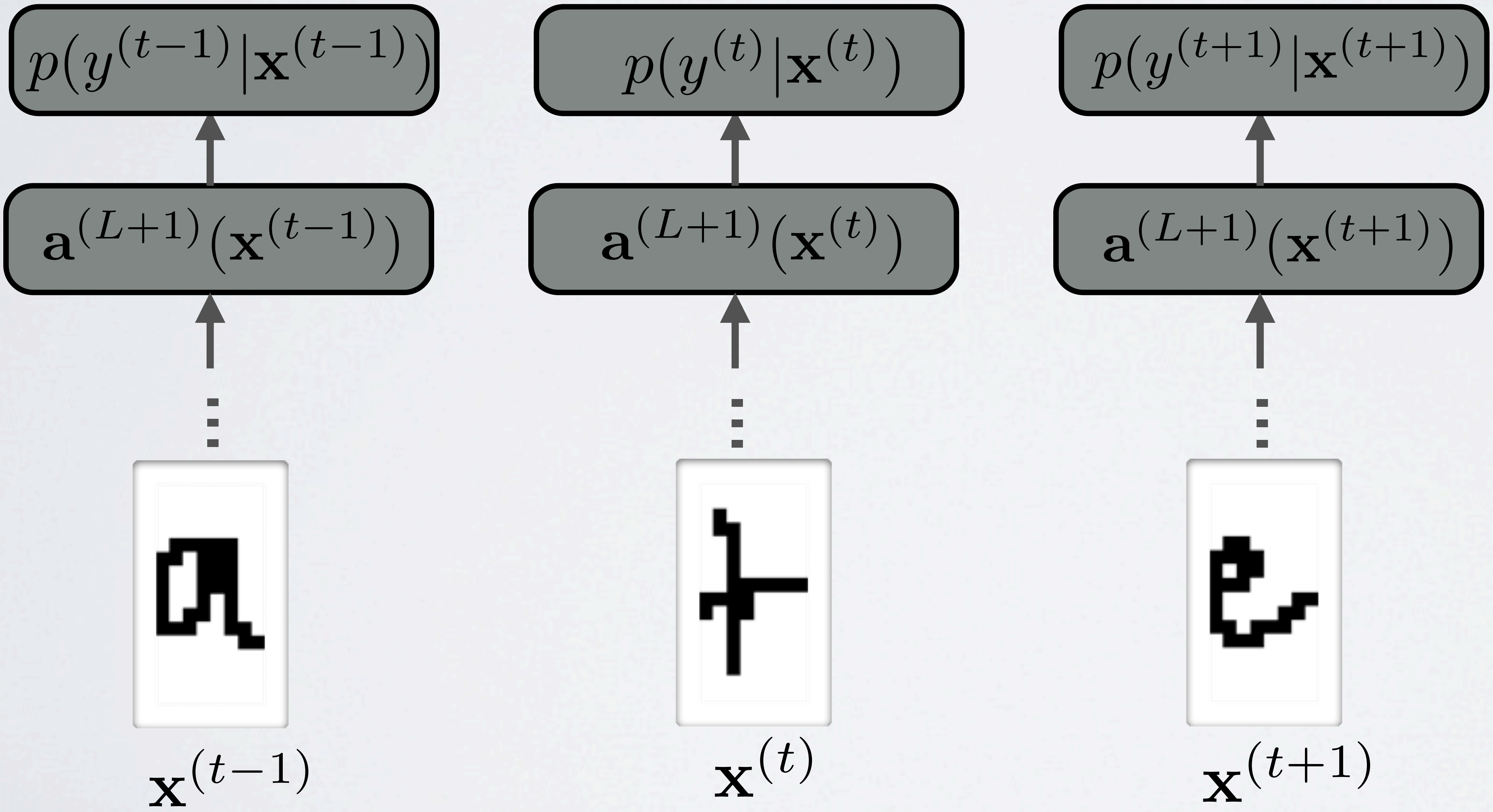
Adapted from Hugo Larochelle's slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

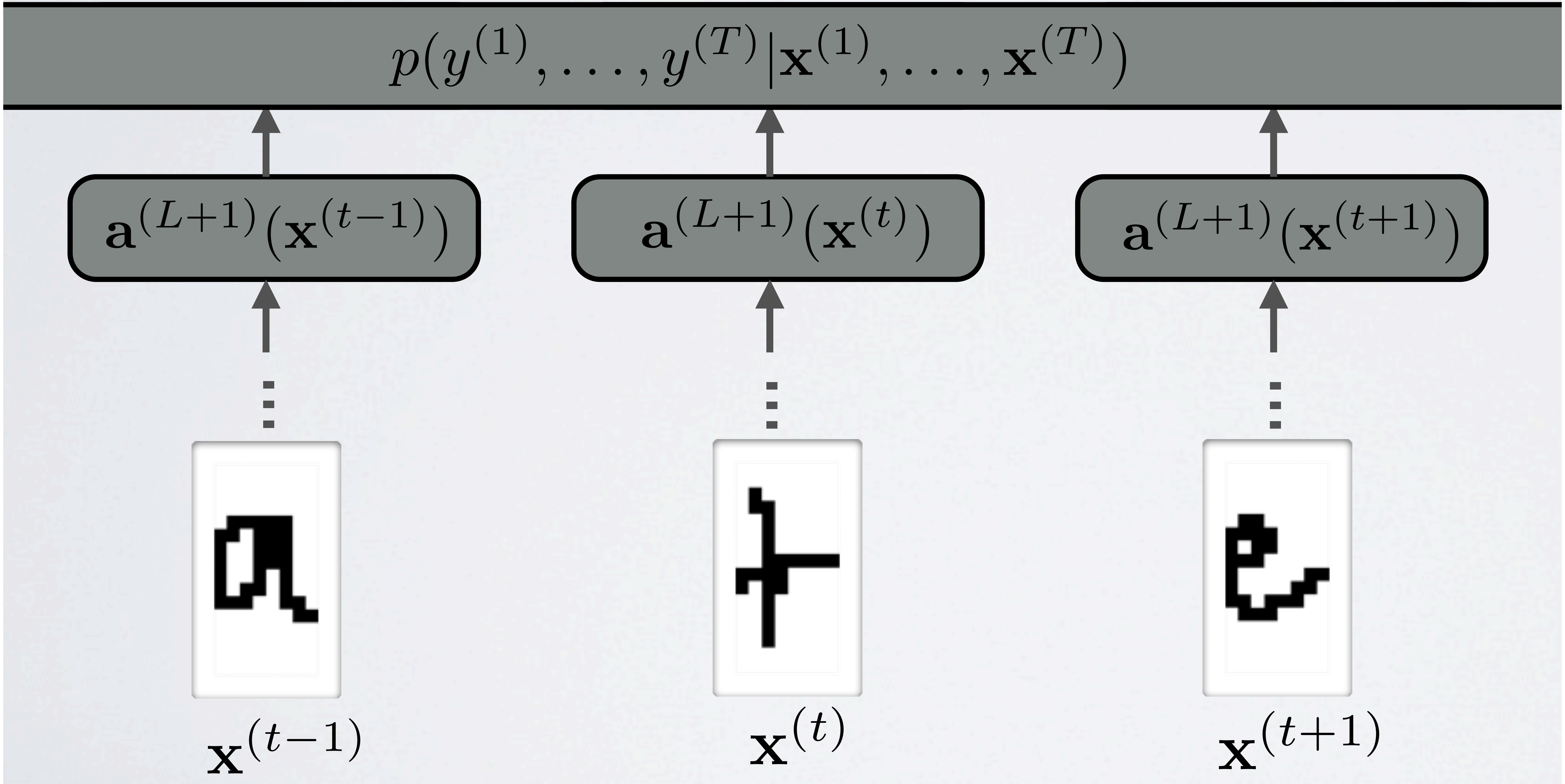
How would we solve this using tools we've learnt so far?

$x_1 =$  $, x_2 =$  $, x_3 =$ 

$$P(y_1, y_2, y_3 \mid x_1, x_2, x_3) = \prod_i P(y_i \mid x_i)$$

Adapted from Hugo Larochelle's slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html


$p(y^{(t-1)}|\mathbf{x}^{(t-1)})$
$p(y^{(t)}|\mathbf{x}^{(t)})$
$p(y^{(t+1)}|\mathbf{x}^{(t+1)})$
$\mathbf{a}^{(L+1)}(\mathbf{x}^{(t-1)})$
$\mathbf{a}^{(L+1)}(\mathbf{x}^{(t)})$
$\mathbf{a}^{(L+1)}(\mathbf{x}^{(t+1)})$
$\mathbf{x}^{(t-1)}$
$\mathbf{x}^{(t)}$
$\mathbf{x}^{(t+1)}$


$p(y^{(1)}, \ldots, y^{(T)} | \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)})$
$\mathbf{a}^{(L+1)}(\mathbf{x}^{(t-1)})$
$\mathbf{a}^{(L+1)}(\mathbf{x}^{(t)})$
$\mathbf{a}^{(L+1)}(\mathbf{x}^{(t+1)})$
$\mathbf{x}^{(t-1)}$
$\mathbf{x}^{(t)}$
$\mathbf{x}^{(t+1)}$

Recall the independent case with a softmax output function

Treat each output independently…

Definition of the softmax function

$$p(\mathbf{y}|\mathbf{X}) = \prod_k p(y_k|\mathbf{x}_k) = \prod_k \exp(a^{(L+1)}(\mathbf{x}_k)_{y_k})/Z(\mathbf{x}_k)$$

$$= \exp\left(\sum_k a^{(L+1)}(\mathbf{x}_k)_{y_k}\right) / \left(\prod_k Z(\mathbf{x}_k)\right)$$

Properties of the exp(x) function

Adapted from Hugo Larochelle's slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Linear Chain Conditional Random Field

We'll come back to how to calculate the partition function

$$p(\mathbf{y}|\mathbf{X}) = \exp\left(\underbrace{\sum_{k=1}^{K} a^{(L+1)}(\mathbf{x}_k)_{y_k}}_{\text{is } y_k \text{ likely given input ?}} + \underbrace{\sum_{k=1}^{K-1} V_{y_k, y_{k+1}}}_{\text{is } y_k \text{ followed by } y_{k+1} \text{ likely?}}\right) / \underbrace{Z(\mathbf{X})}_{\text{partition function}}$$

Adapted from Hugo Larochelle's slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Example

$$p(\mathbf{y} = (\text{“a”}, \text{“t”}, \text{“e”})|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$$

$$= \exp\left(a^{(L+1)}(\mathbf{x}_1)_{\text{“a”}} + a^{(L+1)}(\mathbf{x}_2)_{\text{“t”}} + a^{(L+1)}(\mathbf{x}_3)_{\text{“e”}} + V_{\text{“a”},\text{“t”}} + V_{\text{“t”},\text{“e”}}\right)/Z(\mathbf{X})$$

Adapted from Hugo Larochelle’s slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Example

$$p(\mathbf{y} = (\text{“9”}, \text{“t”}, \text{“e”})|\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$$

$$= \exp\Big(a^{(L+1)}(\mathbf{x}_1)_{\text{“9”}} + a^{(L+1)}(\mathbf{x}_2)_{\text{“t”}} + a^{(L+1)}(\mathbf{x}_3)_{\text{“e”}} +$$

$$V_{\text{“9”},\text{“t”}} + V_{\text{“t”},\text{“e”}}\Big)/Z(\mathbf{X})$$

$p(\mathbf{y} = (\text{“a”})|$  $) \approx p(\mathbf{y} = (\text{“9”})|$  $)$

Adapted from Hugo Larochelle’s slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Example

$$p(\mathbf{y} = (\text{"9"}, \text{"t"}, \text{"e"}) | \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$$

$$= \exp\left(a^{(L+1)}(\mathbf{x}_1)_{\text{"9"}} + a^{(L+1)}(\mathbf{x}_2)_{\text{"t"}} + a^{(L+1)}(\mathbf{x}_3)_{\text{"e"}} + V_{\text{"9"},\text{"t"}} + V_{\text{"t"},\text{"e"}}\right) / Z(\mathbf{X})$$

$$V_{\text{"a"},\text{"t"}} > V_{\text{"9"},\text{"t"}}$$

Adapted from Hugo Larochelle's slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Context windows

$$p(\mathbf{y}|\mathbf{X}) = \exp\Bigg(\sum_{k=1}^{K} a^{(L+1,0)}(\mathbf{x}_k)_{y_k} + \sum_{k=1}^{K-1} V_{y_k, y_{k+1}} + \sum_{k=2}^{K} a^{(L+1,-1)}(\mathbf{x}_{k-1})_{y_k} + \sum_{k=1}^{K-1} a^{(L+1,+1)}(\mathbf{x}_{k+1})_{y_k}\Bigg) / Z(\mathbf{X})$$

Unary log-factor $a_u(y_k)$

Pairwise log-factor $a_p(y_k, y_{k+1})$

Unary log-factor $a_u(y_k)$

Unary log-factor $a_u(y_k)$

is y_k likely given input on the left ?

is y_k likely given input on the right ?

Adapted from Hugo Larochelle's slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

How do we do inference?

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

where

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

hard to compute

Naïve solution is **exponential** in the number of classes

Can solve in O(KC^2) where C is the number of classes using **dynamic programming** (Forward-Backward algorithm). The same procedure allows us to compute marginals, i.e. p($y_k|\boldsymbol{X}$)

Adapted from Hugo Larochelle's slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

How do we do classification?

Two options:

- **Option 1:** At each k, pick y_k with the largest marginal probability p($y_k|\boldsymbol{X}$). i.e. $y_k = argmax_{y_i} p(y_i|\boldsymbol{X})$. If the CRF is the true distribution, this minimizes classification error on expectation.
- **Option 2**: Find the mode of the distribution, $\boldsymbol{y}^* = argmax_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{X})$. Also can be done with dynamic programming using the **Viterbi decoding** algorithm.

How do we train the network?

As before, we train the network by **minimizing NLL** using **SGD**. Our loss is, $l(f(\boldsymbol{X}), \boldsymbol{y}) = -\log p(\boldsymbol{y}|\boldsymbol{X})$

Need **gradients** with respect to the parameters of the **unary** and **pairwise** potentials.

Do **forward pass** -> **forward backward** -> **backprop**

Gradient of unary potentials

$$\nabla_{\mathbf{a}^{(L+1,0)}(\mathbf{x}_k)} - \log p(\mathbf{y}|\mathbf{X}) \quad = \quad -(\mathbf{e}(y_k) - \mathbf{p}(y_k|\mathbf{X}))$$

Gradient of pairwise potentials

matrix of all pairwise marginal probabilities

$$\nabla_{\mathbf{V}} - \log p(\mathbf{y}|\mathbf{X}) \quad = \quad \sum_{k=1}^{K-1} -(\mathbf{e}(y_k)\,\mathbf{e}(y_{k+1})^\top - \overbrace{\mathbf{p}(y_k, y_{k+1}|\mathbf{X})})$$

$$= -\left(\underbrace{\mathrm{freq}(y_k, y_{k+1})} - \sum_{k=1}^{K-1} \mathbf{p}(y_k, y_{k+1}|\mathbf{X})\right)$$

matrix of all pairwise label frequencies

Taking this idea further

We are not limited to linear chain CRFs.

Can use other structures. E.g. grid structure for pixels on image or general n-wise structures (but gets expensive quickly).

In general:

- Training: **Forward pass** to get activations, **forward-backward** to do inference, **backprop** to get gradients.
- If the gradients involve an expectation over y that gets too expensive, could estimate using **sampling** (we'll see an example in a couple of slides).
- Find the most likely sequence (decode) using a **forward pass** to compute activations and **Viterbi decoding**

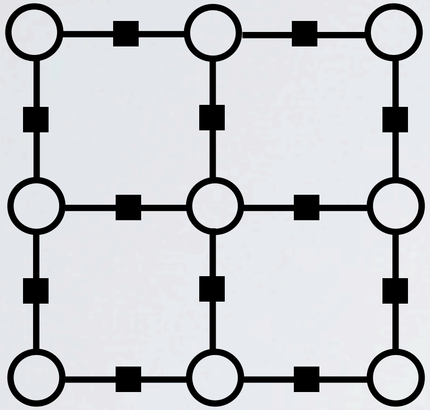
Adapted from Hugo Larochelle's slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Taking this idea further

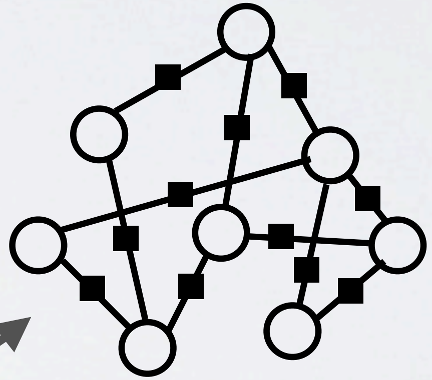
Can use more complex graphical models with neural networks to parameterize the factors.

Examples:

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



Grid structure
(pixels in image)



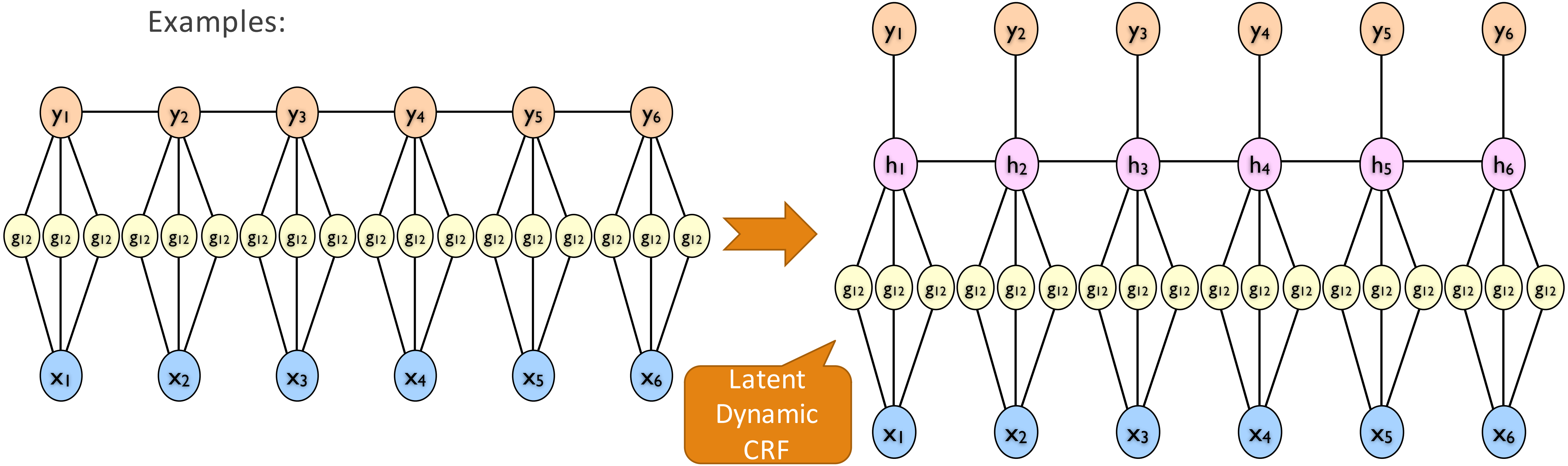
General pair-wise structure
(webpages sharing a link)

Adapted from Hugo Larochelle's slides: http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Taking this idea further

Can use more complex graphical models with neural networks to parameterize the factors.

Examples:



Adapted from: https://www.cs.ubc.ca/~schmidtm/Documents/2014_Notes_LatentCRF.pdf

Discriminative vs generative models

Conditional random fields are discriminative. i.e. We optimize:

$$-\log p(\mathbf{y}|\mathbf{X})$$

Could also model the joint probability by optimizing:

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

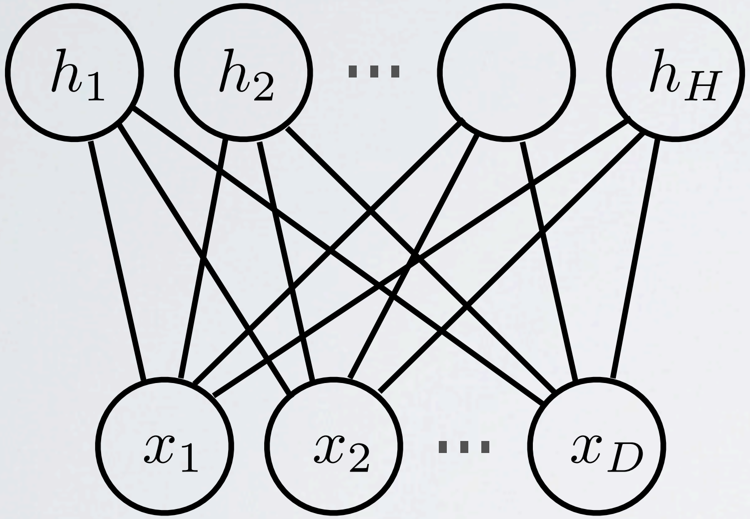
In small data settings / if you have a lot of unlabeled data, the latter can be useful

Unsupervised learning & Generative models

Restricted Boltzman Machine

Special case of the the Boltzman Machine. Restrict the connectivity to make learning easier

In an RBM, the hidden units are conditionally independent given the visible states. This makes sampling easier.

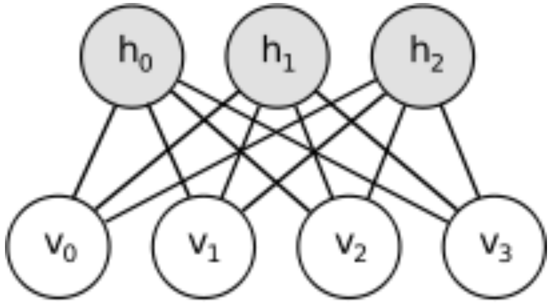


$$p(\mathbf{x}, \mathbf{h}) = \frac{1}{Z} \overbrace{\prod_j \prod_k \exp(W_{j,k} h_j x_k)}^{\text{pair-wise factors}} \underbrace{\prod_k \exp(c_k x_k) \prod_j \exp(b_j h_j)}_{\text{unary factors}}$$

Restricted Boltzman Machine

Special case of the the Boltzman Machine. Restrict the connectivity to make learning easier

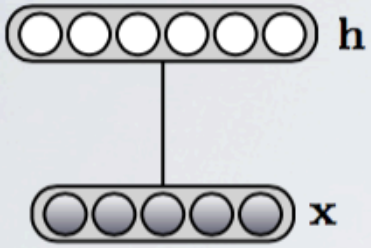
In an RBM, the hidden units are conditionally independent given the visible states. This makes sampling easier.



$$
\begin{aligned}
E(\mathbf{x}, \mathbf{h}) &= -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h} \\
&= -\sum_j \sum_k W_{j,k} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j
\end{aligned}
$$

Distribution: $p(\mathbf{x}, \mathbf{h}) = \exp(-E(\mathbf{x}, \mathbf{h}))/Z$

Inference

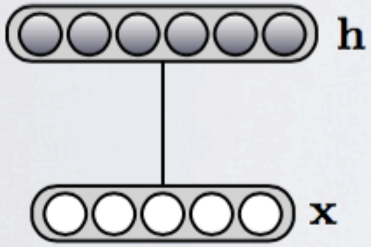


$$p(\mathbf{h}|\mathbf{x}) = \prod_j p(h_j|\mathbf{x})$$

$$p(h_j = 1|\mathbf{x}) = \frac{1}{1 + \exp(-(b_j + \mathbf{W}_{j}.\mathbf{x}))}$$

$$= \text{sigm}(b_j + \mathbf{W}_{j}.\mathbf{x})$$

j th row of $\mathbf{W}$



$$p(\mathbf{x}|\mathbf{h}) = \prod_k p(x_k|\mathbf{h})$$

$$p(x_k = 1|\mathbf{h}) = \frac{1}{1 + \exp(-(c_k + \mathbf{h}^\top \mathbf{W}_{.k}))}$$

$$= \text{sigm}(c_k + \mathbf{h}^\top \mathbf{W}_{.k})$$

k th column of $\mathbf{W}$

Training a Restricted Boltzman Machine

As before, we minimize the average negative log-likelihood of the data:

$$\frac{1}{T}\sum_t l(f(\mathbf{x}^{(t)})) = -\frac{1}{T}\sum_t \log p(\mathbf{x}^{(t)})$$

And then do stochastic gradient decent

$$\frac{\partial - \log p(\mathbf{x}^{(t)})}{\partial \theta} = E_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}^{(t)}, \mathbf{h})}{\partial \theta} \middle| \mathbf{x}^{(t)}\right] - E_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \theta}\right]$$

Positive Phase

Negative Phase
(Hard to compute)

Contrastive Divergence

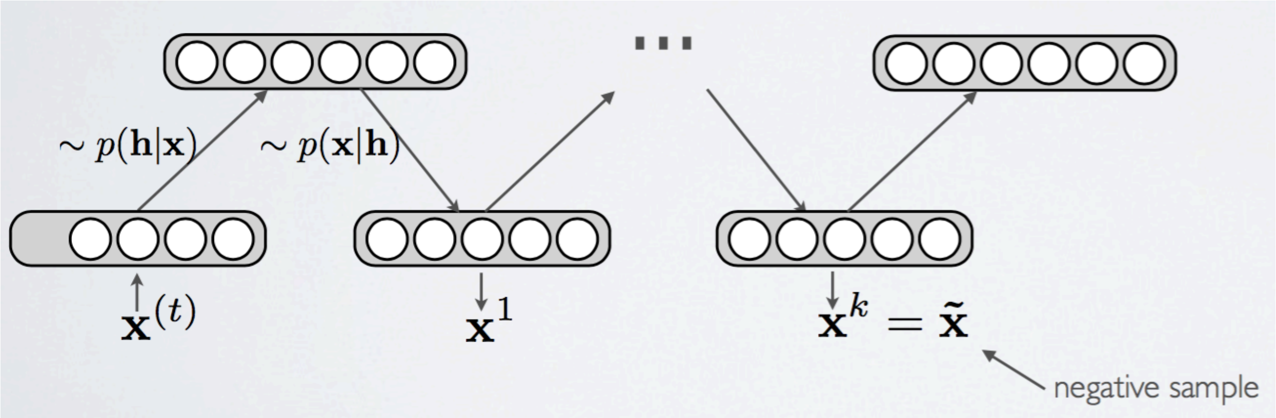
1. The expectation in the negative phase is hard to compute… replace it with a point estimate.

$$-E_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right] \approx -\frac{\partial E(\tilde{\mathbf{x}},\mathbf{h}(\tilde{\mathbf{x}}))}{\partial\theta}$$

Think "Monte Carlo estimate with a single sample"

2. Obtain $\tilde{\mathbf{x}}$ by Gibbs sampling and update parameters.

3. Start the Gibbs sampling chain at $\mathbf{x}^{(t)}$

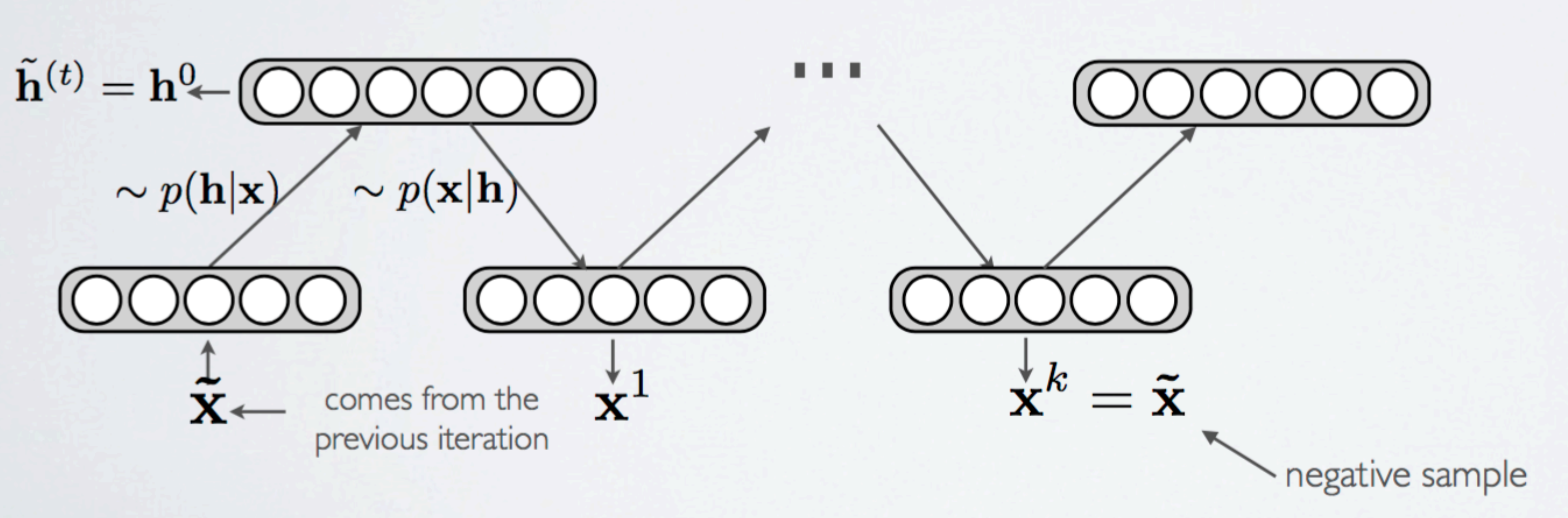


Aside: Gibbs sampling

1. Start from some $\tilde{\mathbf{x}}^{(1)} = (x_1^{(1)}, \ldots, x_n^{(1)})$
2. Sample from conditional $\tilde{\mathbf{x}}^{(k)} \sim p(x_i | x_1^{(1)}, \ldots, x_n^{(i-1)}, x_n^{(i+1)}, \ldots x_n^{(1)})$
3. Repeat

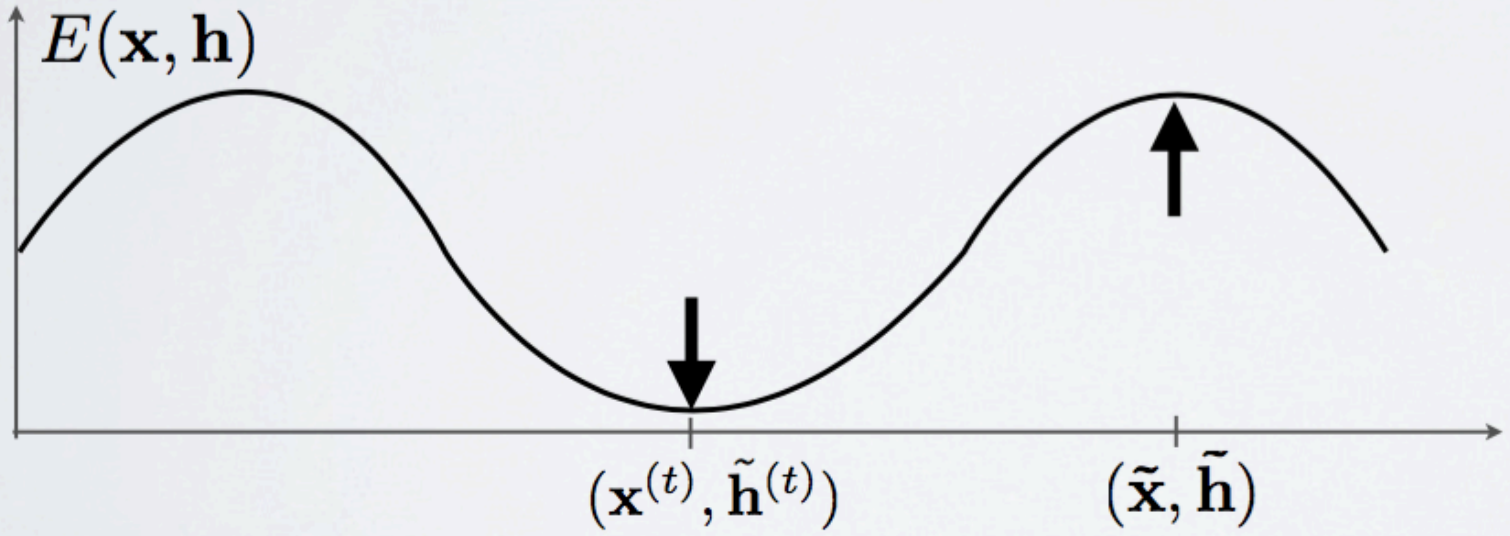
For more MCMC see: http://nbviewer.jupyter.org/github/jhartford/MCMC_talk/blob/master/MCMC%20talk.ipynb

Persistent Contrastive Divergence



Contrastive Divergence

$$\mathrm{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}^{(t)},\mathbf{h})}{\partial\theta}\,\middle|\,\mathbf{x}^{(t)}\right] \approx \frac{\partial E(\mathbf{x}^{(t)},\tilde{\mathbf{h}}^{(t)})}{\partial\theta} \qquad \mathrm{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right] \approx \frac{\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}$$



Deep Belief Network

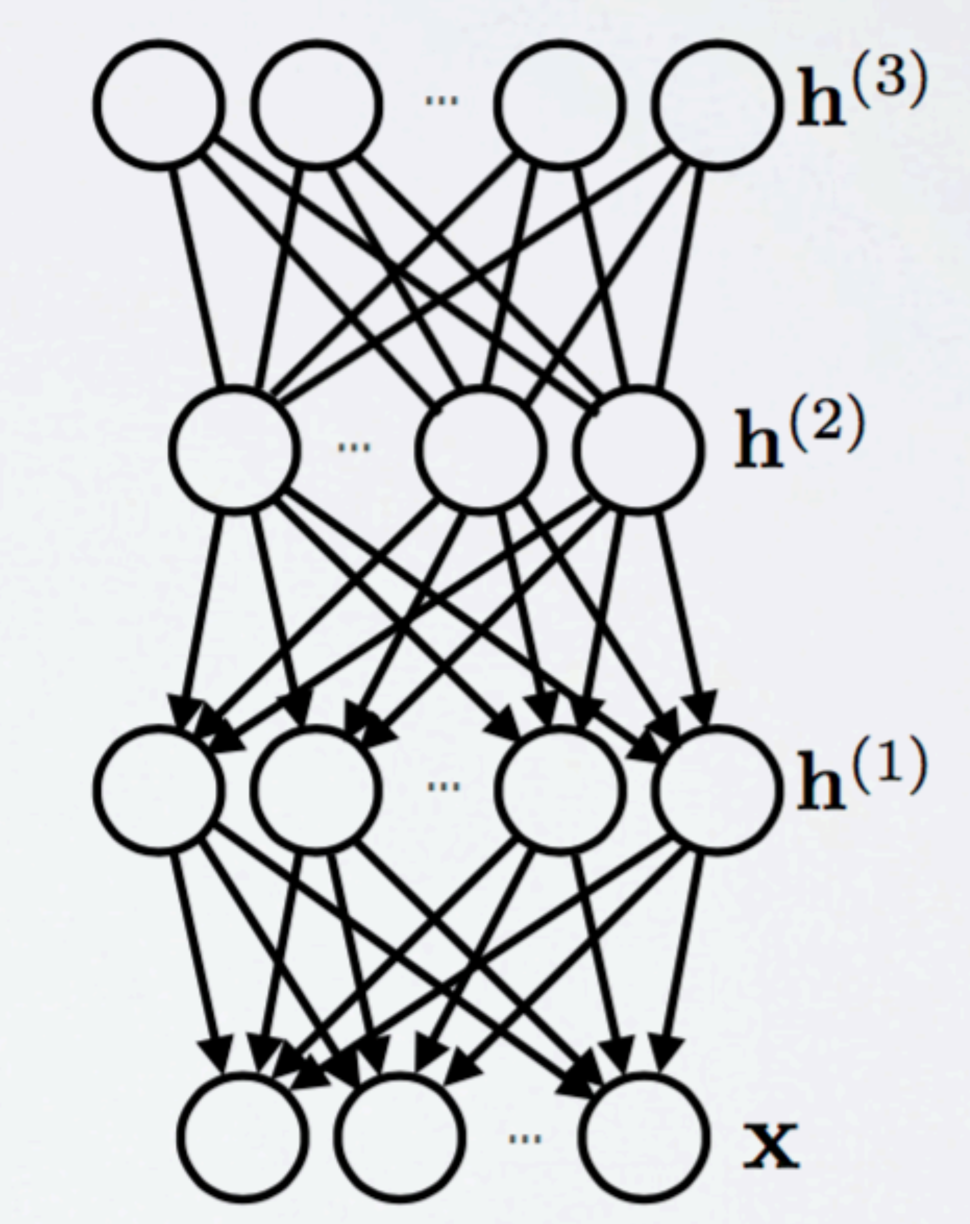
Top two layers for a Restricted Boltzman machine

$$p(\mathbf{h}^{(3)}, \mathbf{h}^{(2)})$$

Bottom three layers form a directed graphical model (sigmoid belief network) where conditional distributions are as follows:

$$p(h_j^{(1)} = 1 | \mathbf{h}^{(2)}) = \text{sigm}(\mathbf{W}^{(1)^T} \mathbf{h}^{(2)} + \mathbf{b}^{(1)})$$

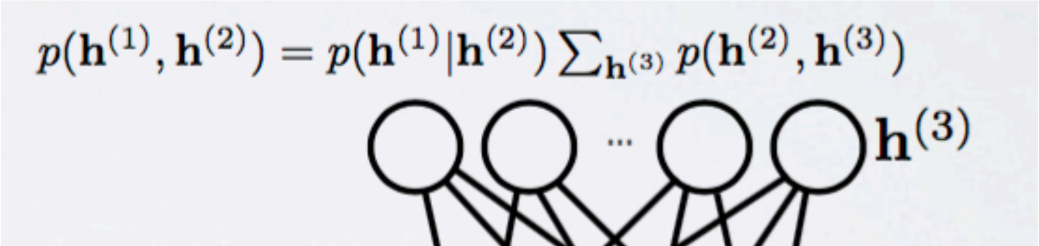
Note differences with regular neural net (sampling)

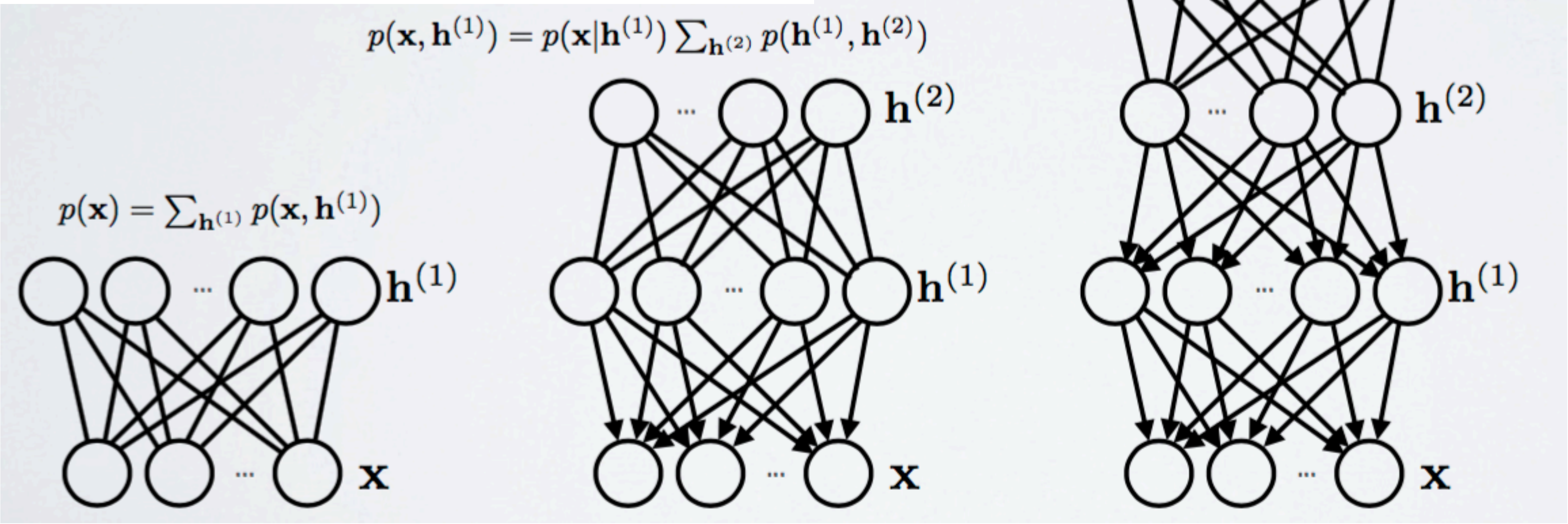


Layer-wise pre-training of a Deep Belief Network

Idea: improve the prior distribution on the last layer by adding another hidden layer

Train RBM using samples from layer below.





Deep belief network

See: http://www.cs.toronto.edu/~hinton/adi/index.htm