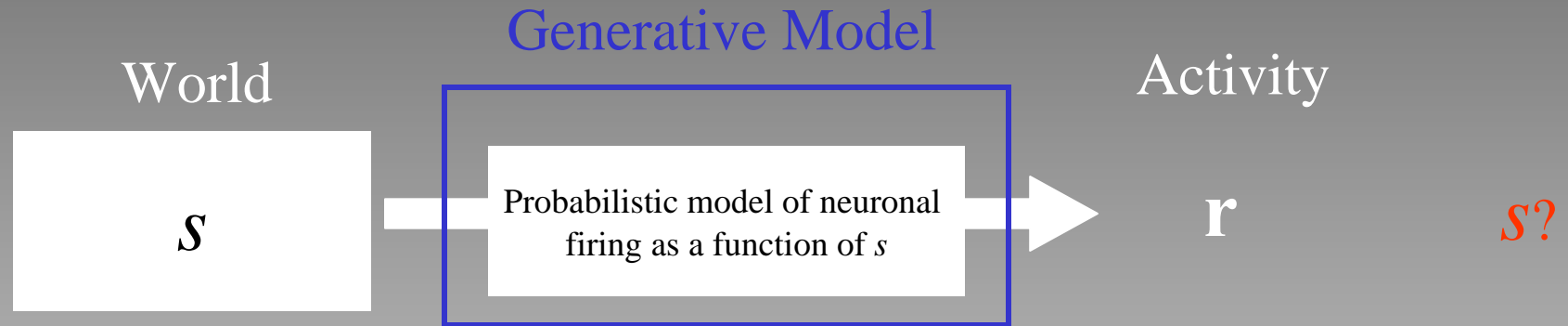
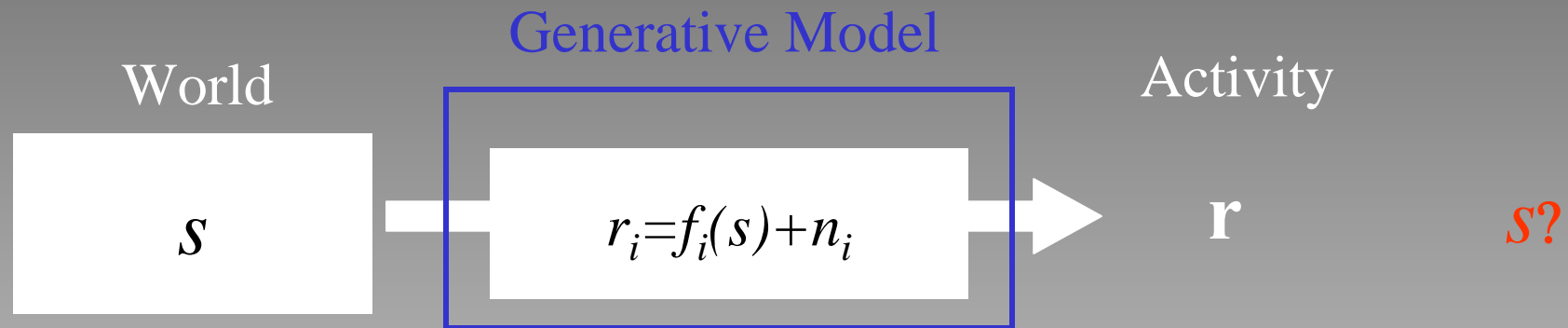


Learning Representations

Maximum likelihood



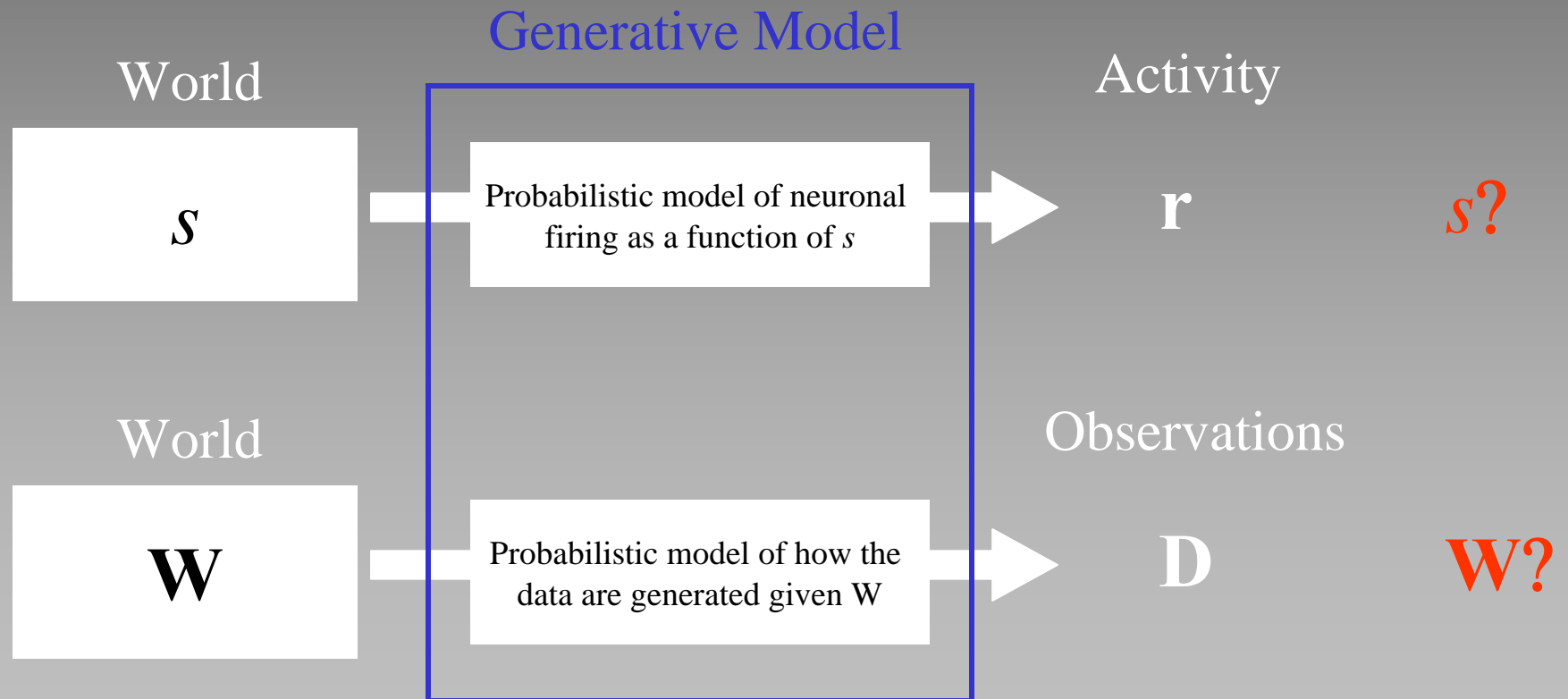
Maximum likelihood



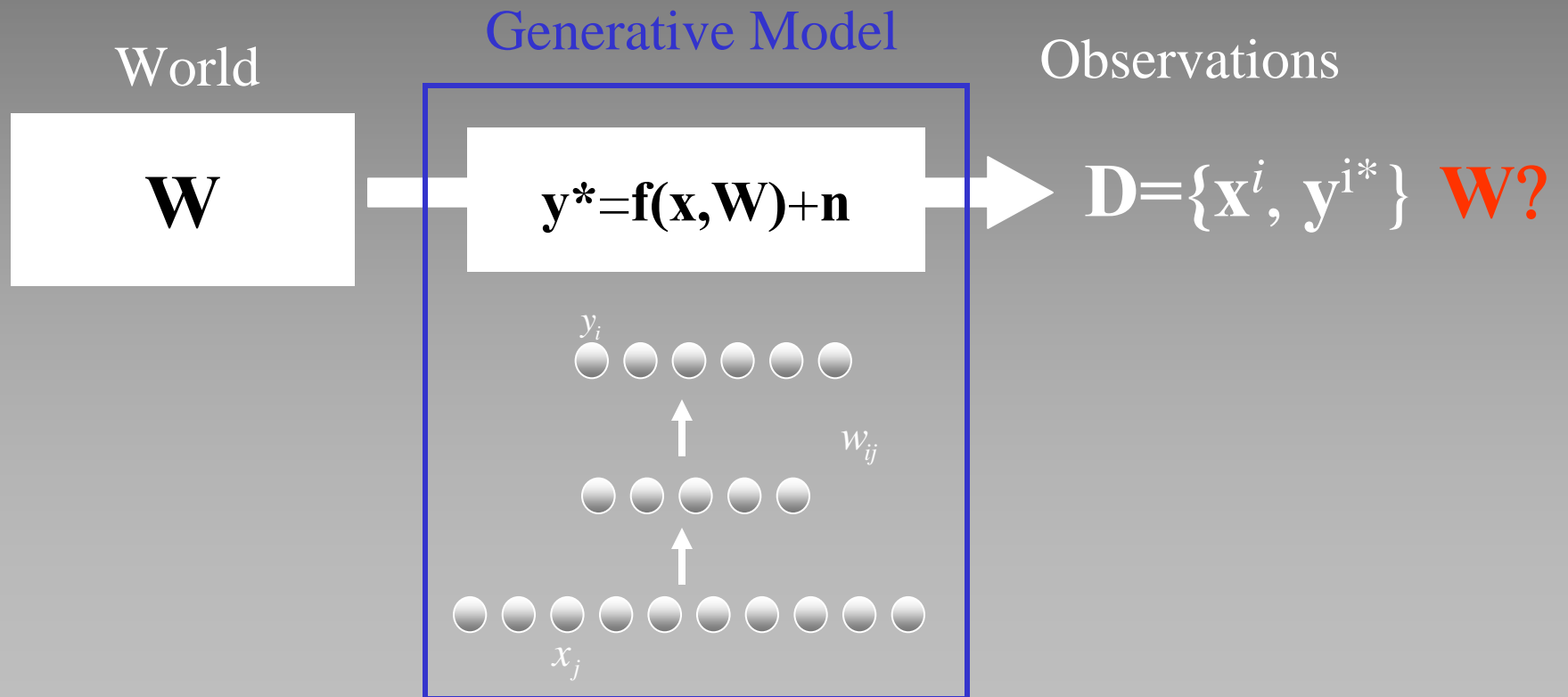
$$p(\mathbf{r} | s) = \prod_i p(r_i | s) = \frac{1}{Z} \exp \left(\sum_i -\frac{(r_i - f_i(s))^2}{2\sigma_i^2} \right)$$

$$\hat{s} = \arg \max_s p(\mathbf{r} | s)$$

Maximum likelihood



Maximum likelihood

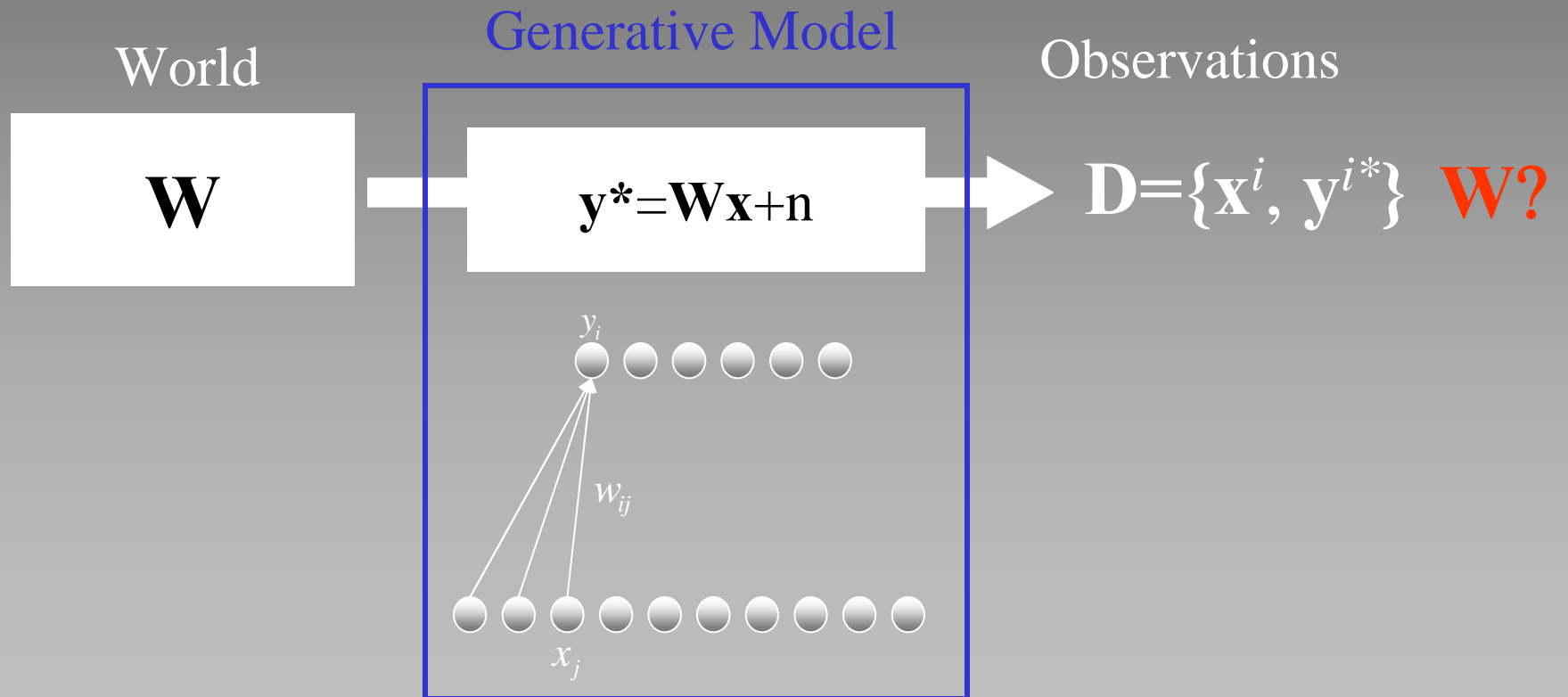


Maximum likelihood learning

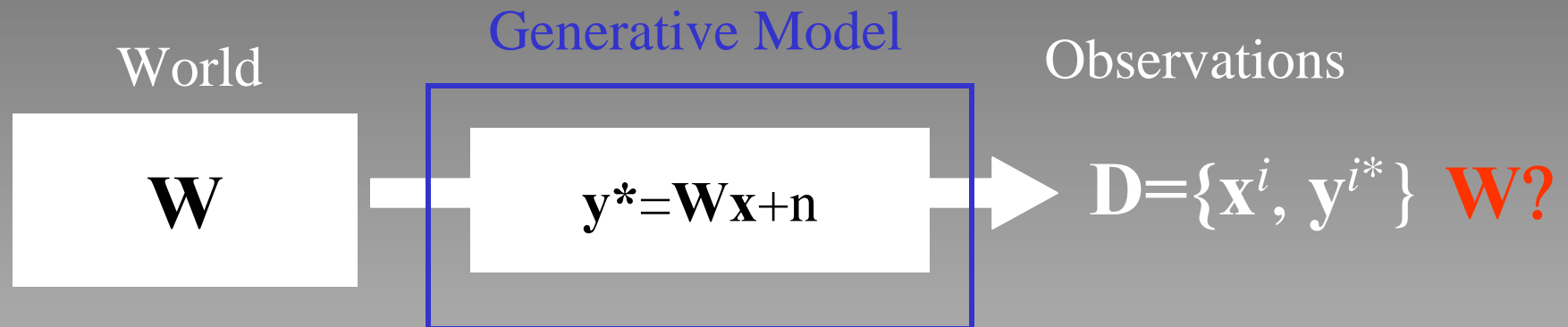
- To learn the optimal parameters \mathbf{W} , we seek to maximum the likelihood of the data which can be done through gradient descent or analytically in some special cases.

$$\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} P(\mathbf{D} | \mathbf{W})$$

Maximum likelihood



Maximum likelihood



$$\begin{aligned} P(\mathbf{D} | \mathbf{W}) &= P(\{\mathbf{x}^i, \mathbf{y}^{i*}\} | \mathbf{W}) \\ &= \prod_i P(\mathbf{y}^{i*} | \mathbf{x}^i, \mathbf{W}) P(\mathbf{x}^i | \mathbf{W}) \\ &\propto \frac{1}{Z} \exp\left(\sum_i -\frac{\|\mathbf{y}^{i*} - \mathbf{y}^i\|^2}{2\sigma^2}\right) \\ &\propto \frac{1}{Z} \exp\left(\sum_i -\frac{\|\mathbf{y}^{i*} - \mathbf{W}\mathbf{x}^i\|^2}{2\sigma^2}\right) \end{aligned}$$

Product over all examples

Note that the \mathbf{y}^{i*} 's are treated as corrupted data

Maximum likelihood learning

$$\log P(D | \mathbf{W}) \propto \sum_i -\frac{\|\mathbf{y}^{i*} - \mathbf{y}^i\|^2}{2\sigma^2}$$

$$\mathbf{W} = \arg \max_{\mathbf{W}} [\log P(D | \mathbf{W})]$$

$$\mathbf{W} = \arg \min_{\mathbf{W}} \sum_i \frac{\|\mathbf{y}^{i*} - \mathbf{y}^i\|^2}{2\sigma^2}$$

- Minimizing quadratic distance is equivalent to maximizing a gaussian likelihood function.

Maximum likelihood learning

$$\mathbf{W} = \arg \min_{\mathbf{W}} \sum_i \frac{\|\mathbf{y}^{i*} - \mathbf{y}^i\|^2}{2\sigma^2}$$

$$\mathbf{W} = \mathbf{C}_{\mathbf{XX}}^{-1} \mathbf{C}_{\mathbf{XY}}$$

$$E = \sum_i \frac{\|\mathbf{y}^{i*} - \mathbf{y}^i\|^2}{2\sigma^2} = \sum_i \frac{\|\mathbf{y}^{i*} - \mathbf{W}\mathbf{x}^i\|^2}{2\sigma^2}$$

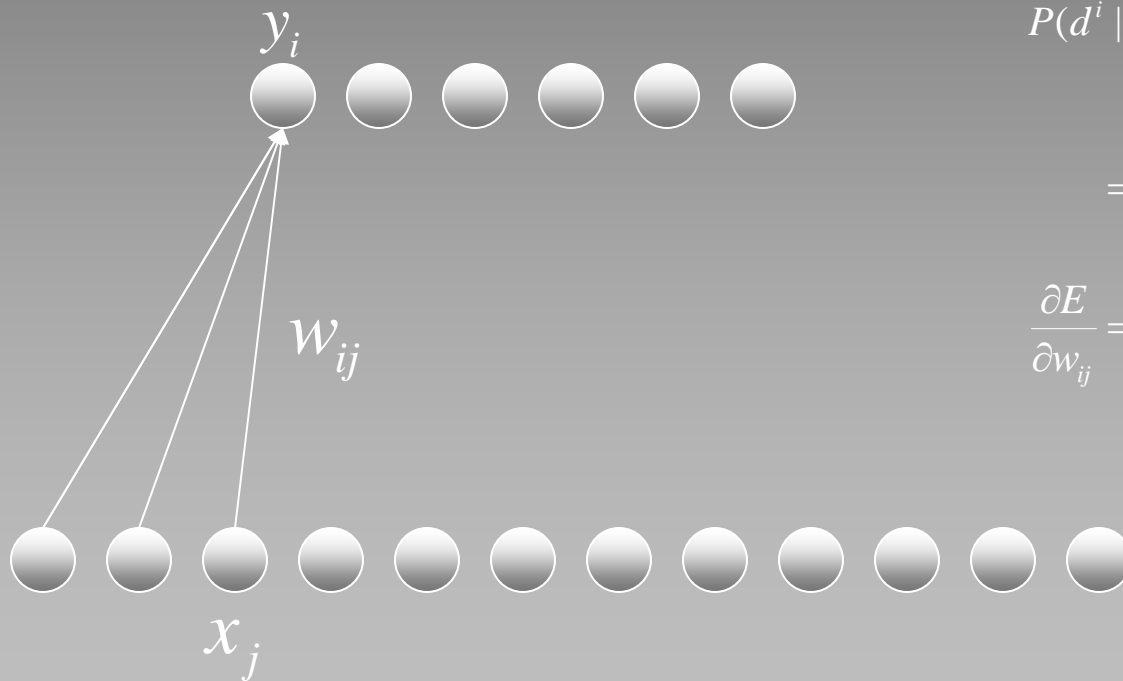
$$\frac{\partial E}{\partial \mathbf{W}} = (\mathbf{y}^{i*} - \mathbf{y}^i) \mathbf{x}_i^T$$

Analytical Solution

Gradient descent:
Delta rule

Maximum likelihood learning

- Example: training a two layer network



$$P(d^i | \mathbf{W}) = \exp\left(-\frac{\|\mathbf{y}^{i*} - \mathbf{y}^i\|^2}{2\sigma^2}\right)$$

$$= \exp\left(-\frac{\|\mathbf{y}^{i*} - \mathbf{W}\mathbf{x}^i\|^2}{2\sigma^2}\right)$$

$$\frac{\partial E}{\partial w_{ij}} = (y^* - y_i)x_j \quad \text{Delta rule}$$

Very important: you need to cross validate

Maximum likelihood learning

Supervised learning:

- The data consists of pairs of input/output vectors $\{\mathbf{x}^i, \mathbf{y}^{i*}\}$.
- Assume that the data were generated by a network and then corrupted by gaussian noise.
- Learning: Adjust the parameters of your network to increase the likelihood that the data were indeed generated by your network.
- Note: if your network is nothing like the system that generated the data, you could be in trouble.

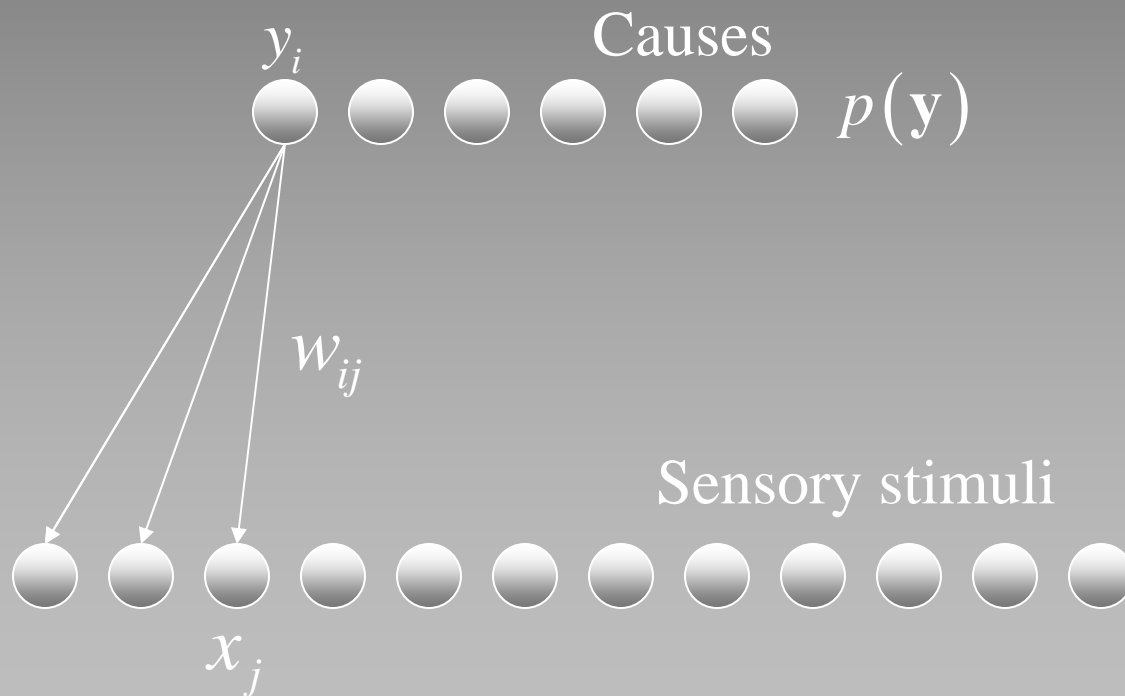
Maximum likelihood learning

Unsupervised learning

- The data consists of input vectors only $\{\mathbf{x}^i\}$.
- Causal models assume that the data are due to some hidden causes plus noise. This is the generative model.
- Goal of learning: given a set of observations, find the parameters of the generative model.
- As usual, we will find the parameters by maximizing the likelihood of the observations.

Maximum likelihood learning

- Example: unsupervised learning in a two layer network



Generative model

$$x_i = \sum_j w_j y_j + n_i$$

$$p(\mathbf{x}, \mathbf{y} | \mathbf{w}) = p(\mathbf{x} | \mathbf{y}, \mathbf{w}) p(\mathbf{y} | \mathbf{w}) \\ = p(\mathbf{x} | \mathbf{y}, \mathbf{w}) p(\mathbf{y})$$

$$p(\mathbf{x} | \mathbf{y}, \mathbf{w}) \sim N(\mathbf{w}\mathbf{y}, \sigma_N)$$

The network represents the joint distribution

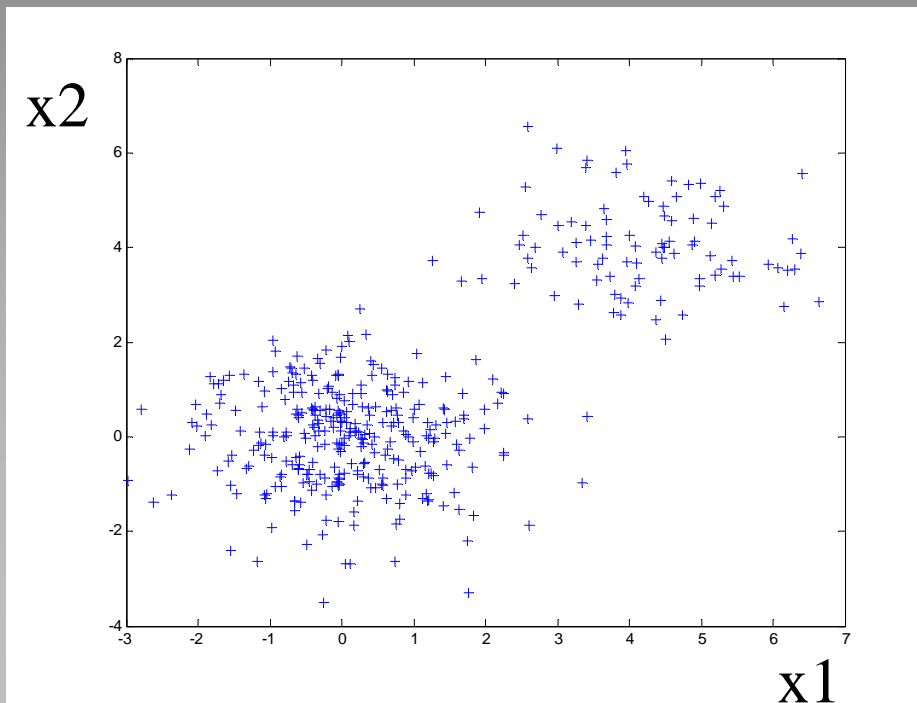
Maximum likelihood learning

- Wait! The network is upside down! Arent't we doing things the wrong way around?
- No: the idea is that what's responsible for the sensory stimuli are high order causes like the presence of physical objects in the world, their identity, their location, their color and so on.
- Generative model goes from the causes/objects to the sensory stimuli
- Recognition will go from stimuli to objects/causes

Maximum likelihood learning

- The network represents the joint distribution $P(\mathbf{x}, \mathbf{y})$. Given the joint distribution, inferences are easy. We use Bayes rule to compute $P(\mathbf{y}|\mathbf{x})$ (recognition) or $P(\mathbf{x}|\mathbf{y})$ (expectations).

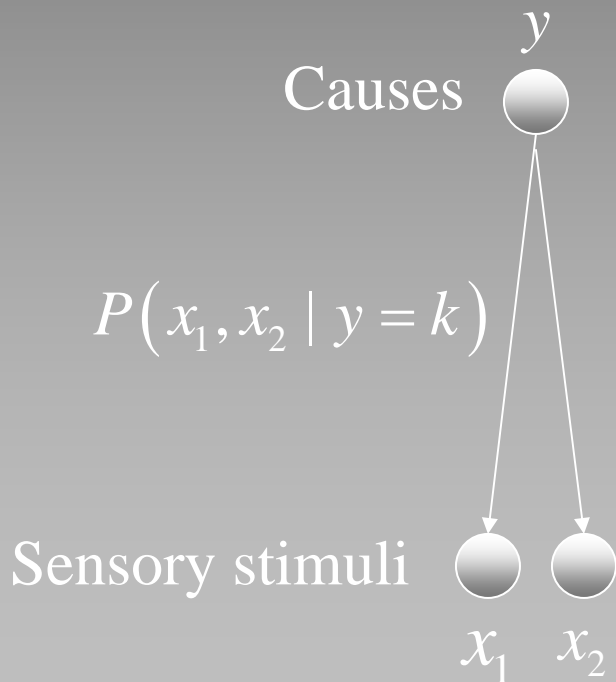
Mixture of Gaussians



Maximum likelihood learning

- Example: Mixture of gaussians

Generative model: 5 parameters,
 $p(y=1), p(y=2), \sigma^2, \mu_k^1, \mu_k^2$.



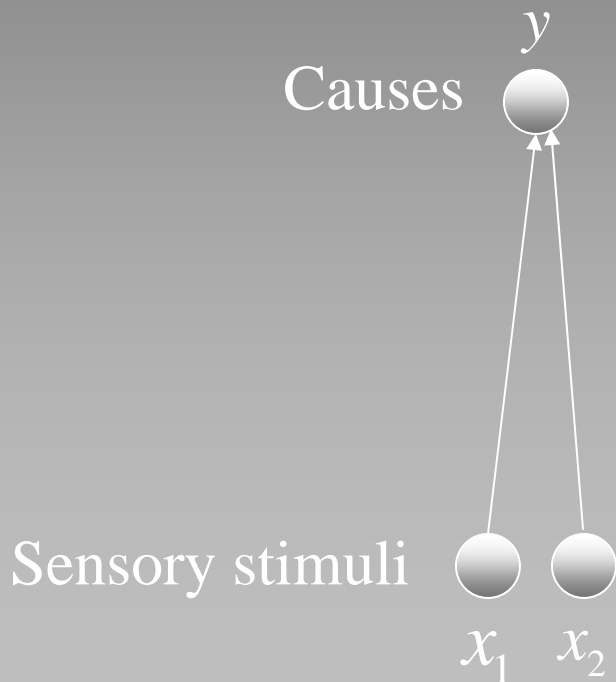
$$P(x_1, x_2 | y = k)$$

$$P(x_1, x_2 | y = k) = \frac{1}{Z} \exp\left(-\frac{(x_1 - \mu_k^1) + (x_2 - \mu_k^2)}{2\sigma^2}\right)$$

Maximum likelihood learning

- Example: mixture of gaussians

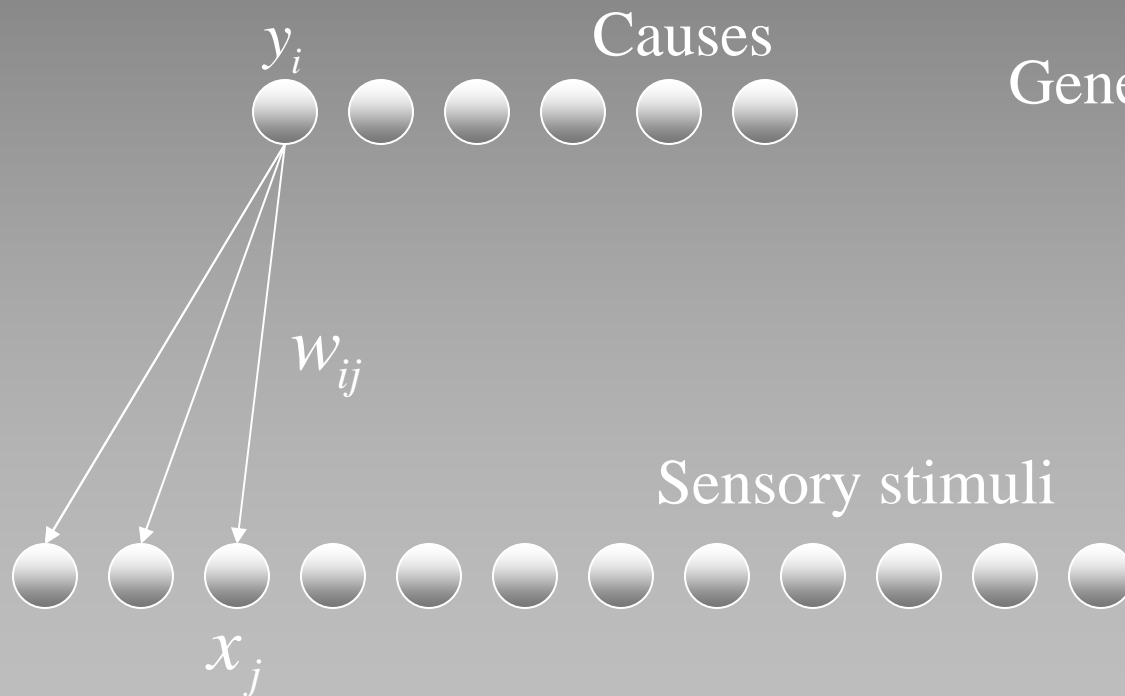
Recognition model: given a pair \mathbf{x} 's, what was the cluster?



$$P(y | \mathbf{x}) = \frac{P(\mathbf{x} | y) P(y)}{P(\mathbf{x})}$$

Maximum likelihood learning

- Example: unsupervised learning in a two layer network



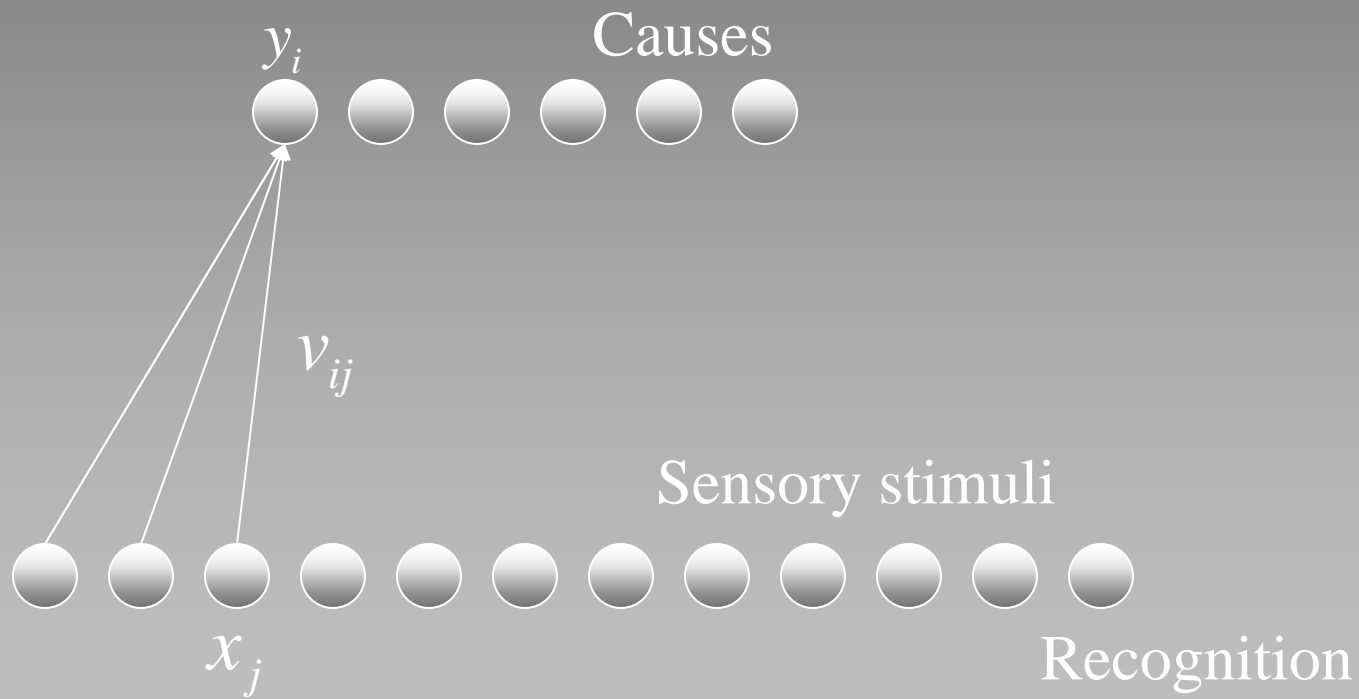
Generative model

$$x_i = \sum_j w_j y_j + n_i$$

$$p(\mathbf{x}, \mathbf{y} | \mathbf{w}) = p(\mathbf{x} | \mathbf{y}, \mathbf{w}) p(\mathbf{y} | \mathbf{w}) \\ = p(\mathbf{x} | \mathbf{y}, \mathbf{w}) p(\mathbf{y})$$

$$p(\mathbf{x} | \mathbf{y}, \mathbf{w}) \sim N(\mathbf{w}\mathbf{y}, \sigma_N)$$

Maximum likelihood learning



Maximum likelihood learning

- Learning consist in adjusting the weights to maximum the likelihood of the data, $P(\{\mathbf{x}^i\} | \mathbf{w})$
- Problem: the generative model specifies:

$$P(\mathbf{x}^i | \mathbf{y}^i, \mathbf{w})$$

The data set does not specify the hidden causes \mathbf{y}^i , which we would need for a learning rule like delta rule...

Maximum likelihood learning

Fix #1. You don't know y^i ? Estimate it! Pick the MAP estimate of y^i on each trial (Olshausen and Field, as we will see later):

$$\begin{aligned}\hat{y}^i &= \arg \max_y p(\mathbf{y} | \mathbf{x}^i, \mathbf{w}) \\ &= \arg \max_y p(\mathbf{x}^i | \mathbf{y}, \mathbf{w}) p(\mathbf{y} | \mathbf{w})\end{aligned}$$

Note that the weights are presumably incorrect (otherwise, there would be no need for learning). As a result, the y 's obtained this way are also incorrect. Hopefully, they're good enough...

Main problem: this breaks down if $p(\mathbf{y}^i | \mathbf{x}^i, \mathbf{w})$ is multimodal.

Maximum likelihood learning

Fix #2. Sample y^i from $P(y/x^i, w)$ using Gibbs sampling.

Slow, and again we're sampling from the wrong distribution...However, this is a much better idea for multimodal distributions.

Maximum likelihood learning

Fix # 3. Marginalization

$$\begin{aligned} p(\mathbf{x}^i | \mathbf{w}) &= \sum_y p(\mathbf{x}^i, \mathbf{y} | \mathbf{w}) \\ &= \sum_y p(\mathbf{x}^i | \mathbf{y}, \mathbf{w}) p(\mathbf{y} | \mathbf{w}) \end{aligned}$$

Use gradient descent to adjust the parameters of the likelihood and prior (very slow).

Maximum likelihood learning

Fix #3. Marginalization. Gradient descent for mixture of two gaussians

$$\begin{aligned} E &= \sum_{i=1, N} \log P(\mathbf{x}^i | \mathbf{w}) \\ &= \sum_{i=1, N} \log(p(\mathbf{x}^i | y = 1, \mu_1, \sigma_1^2) p(y = 1) \\ &\quad + p(\mathbf{x}^i | y = 2, \mu_2, \sigma_2^2) p(y = 2)) \end{aligned}$$

Even when the $p(\mathbf{x}|y)$'s are gaussian, the resulting likelihood function is not quadratic.

Maximum likelihood learning

Fix #3. Marginalization

$$\begin{aligned} p(\mathbf{x}^i | \mathbf{w}) &= \sum_{\mathbf{y}} p(\mathbf{x}^i, \mathbf{y} | \mathbf{w}) \\ &= \sum_{\mathbf{y}} p(\mathbf{x}^i | \mathbf{y}, \mathbf{w}) p(\mathbf{y}) \end{aligned}$$

Rarely feasible in practice

If \mathbf{y} is binary vector of N dimension, that sum contains 2^N terms...

Maximum likelihood learning

Fix #4. The expectation-maximization algorithm
(EM)

$$p(\mathbf{x}^i | \mathbf{w}) = \sum_y p(\mathbf{x}^i | y, \mathbf{w})p(y | \mathbf{w})$$

Maximum likelihood learning

EM: How can we optimize $p(\mathbf{y}|\mathbf{w})$? Let γ_1 be $p(\mathbf{y}=\mathbf{1}|\mathbf{w})$

$$\begin{aligned}\gamma_1 &= p_{true}(y=1) \\ &= \sum_{\mathbf{x}} p_{true}(y=1, \mathbf{x}) \\ &= \sum_{\mathbf{x}} p_{true}(y=1|\mathbf{x})p_{true}(\mathbf{x}) \\ &= \frac{1}{N} \sum_{i=1, N} p_{true}(y=1|\mathbf{x}^i)\end{aligned}$$

We have samples of this

But we don't know this...
Trick: use an approximation

Maximum likelihood learning

E-step: use the current parameters to approximate $p_{\text{true}}(y|\mathbf{x})$ with $p(y|\mathbf{x},\mathbf{w})$

$$p(y = 1 | \mathbf{x}, \mathbf{w}) = \frac{p(\mathbf{x} | y = 1, \mathbf{w}) p(y = 1 | \mathbf{w})}{p(\mathbf{x})}$$

Maximum likelihood learning

EM: M step: optimize $p(y)$

$$\begin{aligned}\gamma_1 &= \sum_{\mathbf{x}} p_{true}(y=1|\mathbf{x}) p_{true}(\mathbf{x}) \\ &= \sum_{\mathbf{x}} p(y=1|\mathbf{x}, \mathbf{w}) p_{true}(\mathbf{x}) \\ &= \sum_{\mathbf{x}} \frac{p(\mathbf{x}|y=1, \mathbf{w}) p(y=1|\mathbf{w})}{p(\mathbf{x})} p_{true}(\mathbf{x}) \\ &= \sum_{i=1, N} p(\mathbf{x}^i | y=1, \mathbf{w}) p(y=1|\mathbf{w})\end{aligned}$$

From E step

Maximum likelihood learning

EM: M step. For the mean of $p(\mathbf{x}/y=1)$ use:

$$\begin{aligned}\mu_1 &= \sum_{\mathbf{x}} p_{true}(\mathbf{x} | y = 1) \mathbf{x} \\ &= \sum_{\mathbf{x}} \frac{p_{true}(y = 1 | \mathbf{x}) p_{true}(\mathbf{x}) \mathbf{x}}{p_{true}(y = 1)} \\ &= \sum_{\mathbf{x}} \frac{p(y = 1 | \mathbf{x}, \mathbf{w}) p_{true}(\mathbf{x}) \mathbf{x}}{p(y = 1 | \mathbf{w})} \\ &= \frac{1}{N} \sum_{i=1, N} \frac{p(y = 1 | \mathbf{x}^i, \mathbf{w}) \mathbf{x}^i}{\gamma_1}\end{aligned}$$

From E step

Maximum likelihood learning

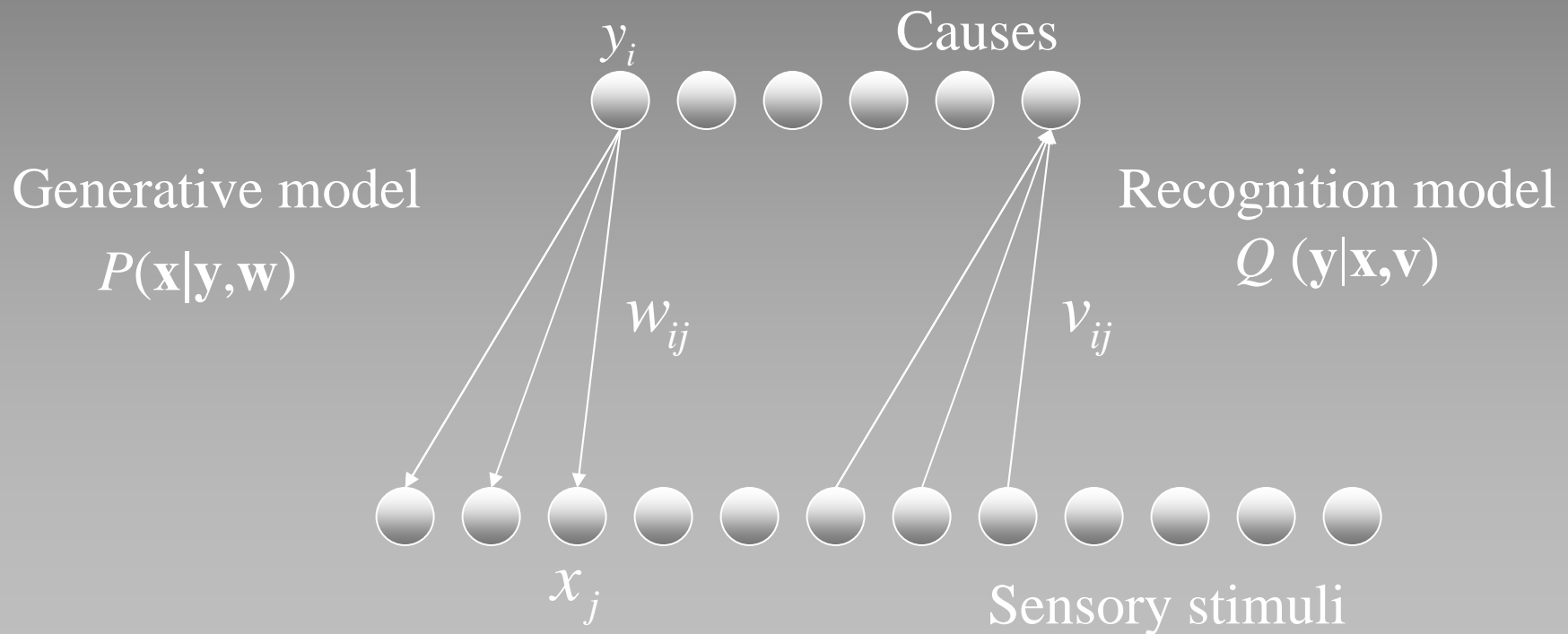
EM: Iterate the E and M step. Guaranteed to converge, but it could be a local minima.

Maximum likelihood learning

Fix #5. Model the recognition distribution and use EM for training. Wake-Sleep algorithm (Helmholtz machine).

Maximum likelihood learning

Helmholtz machine



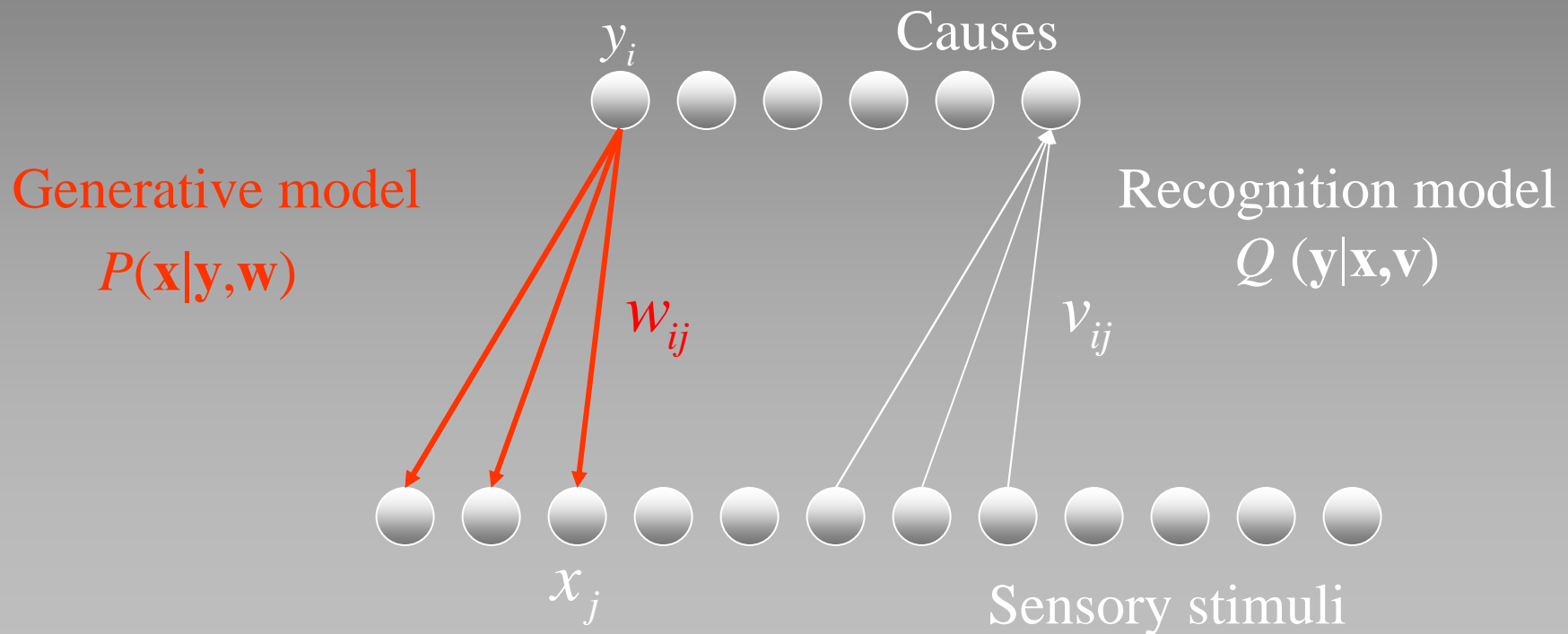
Maximum likelihood learning

Fix #5. Model the recognition distribution and use EM for training. Wake-Sleep algorithm (Helmholtz machine).

(Wake) M step: Use \mathbf{x} 's to generate \mathbf{y} according to $Q(\mathbf{y}|\mathbf{x},\mathbf{v})$, and adjust the \mathbf{w} in $P(\mathbf{x}|\mathbf{y},\mathbf{w})$.

Maximum likelihood learning

Helmholtz machine



Maximum likelihood learning

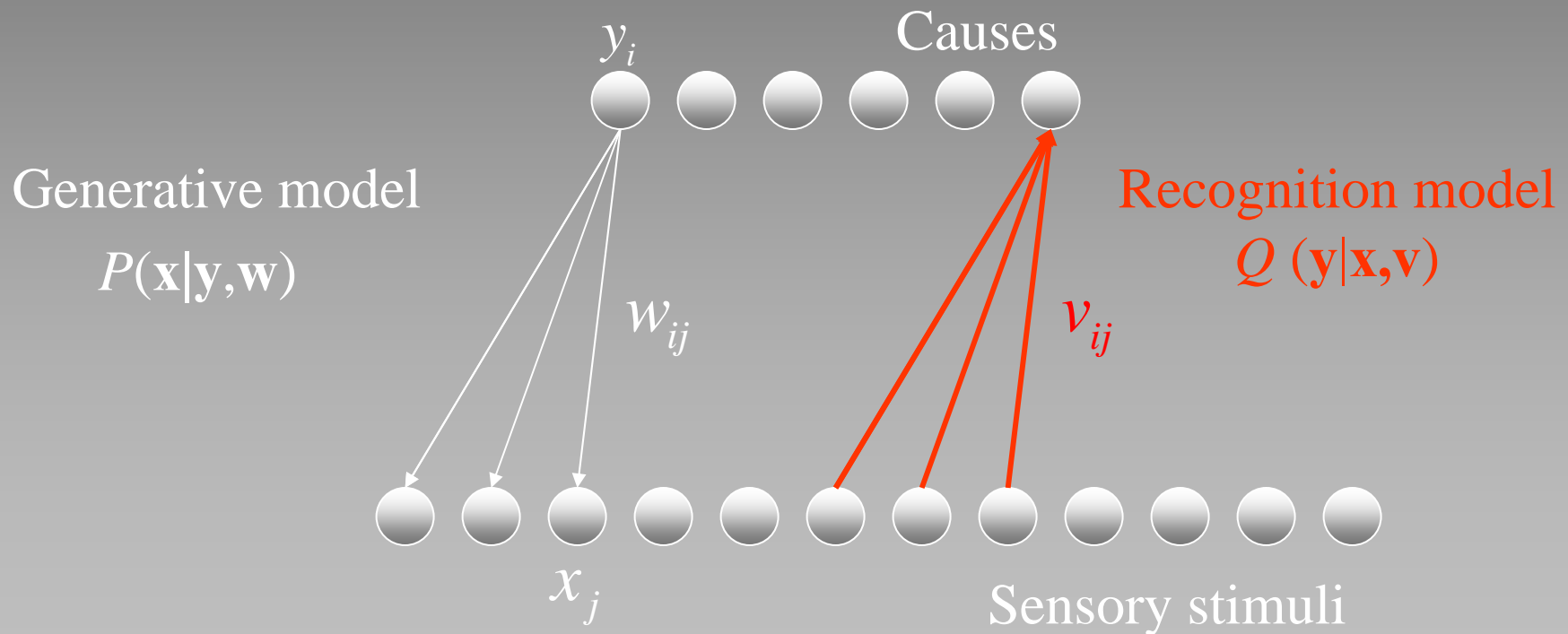
Fix #5. Model the recognition distribution and use EM for training. Wake-Sleep algorithm (Helmholtz machine).

(Wake) M step: Use \mathbf{x} 's to generate \mathbf{y} according to $Q(\mathbf{y}|\mathbf{x},\mathbf{v})$, and adjust the \mathbf{w} in $P(\mathbf{x}|\mathbf{y},\mathbf{w})$.

(Sleep) E step: Generate \mathbf{y} with $P(\mathbf{y}|\mathbf{w})$, and use it to generate \mathbf{x} according to $P(\mathbf{x}|\mathbf{y},\mathbf{w})$. Then adjust the \mathbf{v} in $Q(\mathbf{y}|\mathbf{x},\mathbf{v})$.

Maximum likelihood learning

Helmholtz machine



Maximum likelihood learning

Fix #5. Model the recognition distribution and use EM for training. Wake-Sleep algorithm (Helmholtz machine).

Advantage: After several approximations, you can get both learning rules to look like delta rule...

Usable for hierarchical architectures.

Sparse representations in V1

- Ex: Olshausen and Field: a natural image is generated according to a two step process:

1. a set of coefficients, $\{a_i\}$ is drawn according to a sparse prior (Cauchy or related)

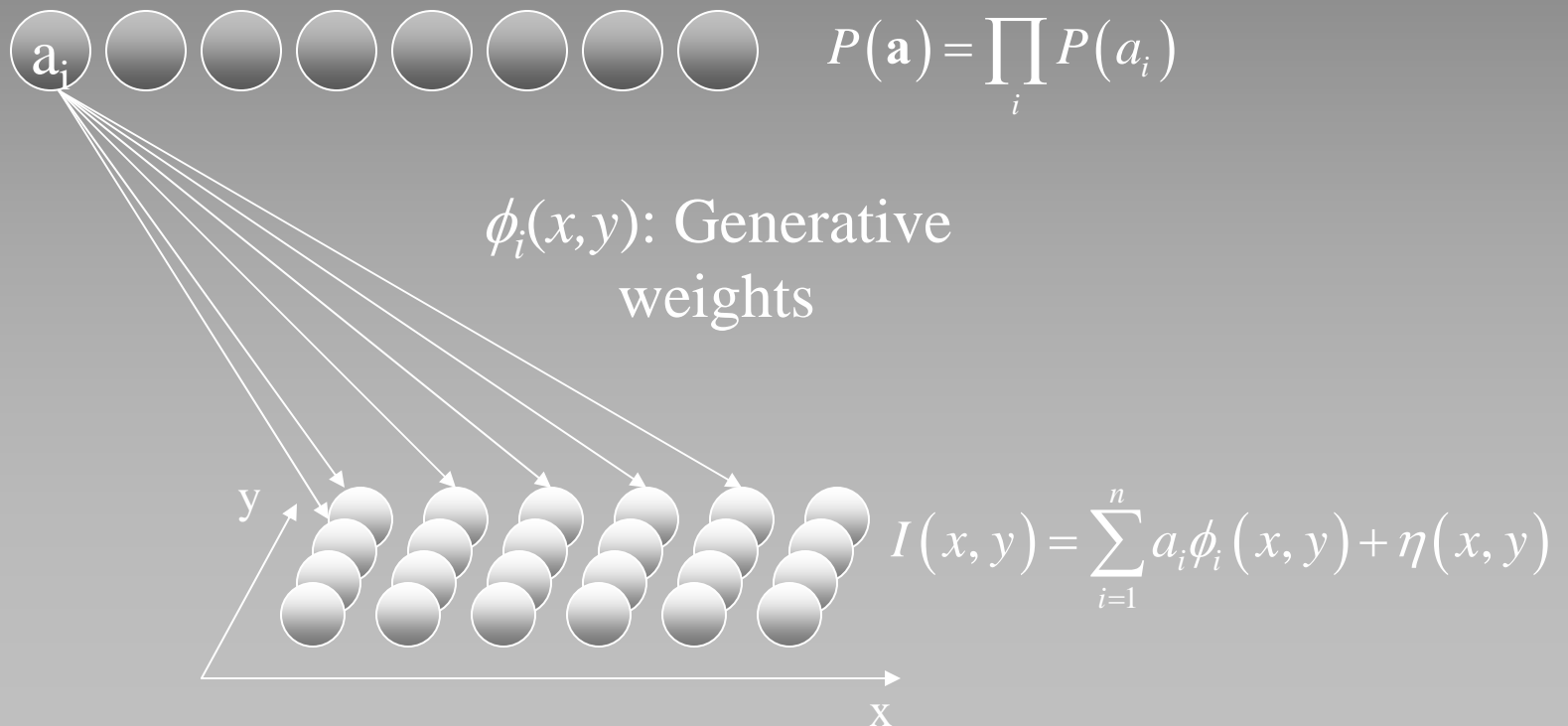
$$P(\mathbf{a}) = \prod_i P(a_i) = \prod_i \frac{1}{Z_\beta} \exp(-\beta |a_i|)$$

2. The image is the result of combining a set of basis functions weighted by the coefficient c_i and corrupted by gaussian noise

$$I(x, y) = \sum_{i=1}^n a_i \phi_i(x, y) + \eta(x, y)$$

Sparse representations in V1

- Network representation



Sparse representations in V1

- The sparse prior favors solution with most coefficients set to zero and a few with a high value.

$$P(\mathbf{a}) = \prod_i P(a_i) = \prod_i \frac{1}{Z_\beta} \exp(-\beta |a_i|)$$

- Why a sparse prior? Because the response of neurons to natural images is non gaussian and tend to be sparse.

Sparse representations in V1

- The likelihood function:

$$I(x, y) = \sum_{i=1}^n a_i \phi_i(x, y) + \eta(x, y)$$

$$P(I | \mathbf{a}, \{\phi_i\}) = \prod_{x, y} P(I(x, y) | \mathbf{a}, \{\phi_i\})$$

$$= \prod_{x, y} \frac{1}{Z_{\sigma_N}} \exp \left(- \frac{\left(I(x, y) - \sum_{i=1}^n a_i \phi_i(x, y) \right)^2}{2\sigma_N^2} \right)$$

$$= \frac{1}{Z_{\sigma_N}} \exp \left(- \frac{\sum_{x, y} \left(I(x, y) - \sum_{i=1}^n a_i \phi_i(x, y) \right)^2}{2\sigma_N^2} \right)$$

$$\log(P(I | \mathbf{a}, \{\phi_i\})) = - \frac{\sum_{x, y} \left(I(x, y) - \sum_{i=1}^n a_i \phi_i(x, y) \right)^2}{2\sigma_N^2}$$

Sparse representations in V1

- The generative model is a model of the joint distribution $P(I, \mathbf{a} | \{\phi_i\})$

$$\begin{aligned} P(I, \mathbf{a} | \{\phi_i\}) &= P(I | \mathbf{a}, \{\phi_i\}) P(\mathbf{a} | \{\phi_i\}) \\ &= P(I | \mathbf{a}, \{\phi_i\}) P(\mathbf{a}) \end{aligned}$$

Sparse representations in V1

Learning:

1. Given a set of natural images, how do you learn the basis functions?

Answer: find the basis functions maximizing the likelihood of the images, $P(\{I_k\}|\{\phi_i\})$. Sure, but where to you get the \mathbf{a} 's?

Olhausen and Field: For each image, pick the \mathbf{a} 's maximizing the posterior over \mathbf{a} , $P(\mathbf{a}|I_k, \{\phi_i\})$ (Fix#1).

$$P(\mathbf{a} | I_k, \{\phi_i\}) \propto P(I_k | \mathbf{a}, \{\phi_i\}) P(\mathbf{a})$$

Network implementation

$$P(\mathbf{a} | I_k, \{\phi_i\}) \propto P(I_k | \mathbf{a}, \{\phi_i\}) P(\mathbf{a})$$

$$\ln P(\mathbf{a} | \mathbf{I}, \boldsymbol{\phi}) \propto \ln P(\mathbf{I} | \mathbf{a}, \boldsymbol{\phi}) + \ln P(\mathbf{a})$$

$$\tau \frac{d\mathbf{a}}{dt} = \frac{\partial \ln P(\mathbf{I} | \mathbf{a}, \boldsymbol{\phi})}{\partial \mathbf{a}} + \frac{\partial \ln P(\mathbf{a})}{\partial \mathbf{a}}$$

$$= \frac{\partial \|\mathbf{I} - \boldsymbol{\phi}\mathbf{a}\|^2}{\partial \mathbf{a}} + \frac{\partial \ln P(\mathbf{a})}{\partial \mathbf{a}}$$

$$= \boldsymbol{\phi}^T (\mathbf{I} - \boldsymbol{\phi}\mathbf{a}) + f'(\mathbf{a})$$

$$\tau \frac{d\mathbf{a}}{dt} = \boldsymbol{\phi}^T \mathbf{I} - \boldsymbol{\phi}^T \boldsymbol{\phi}\mathbf{a} + f'(\mathbf{a})$$

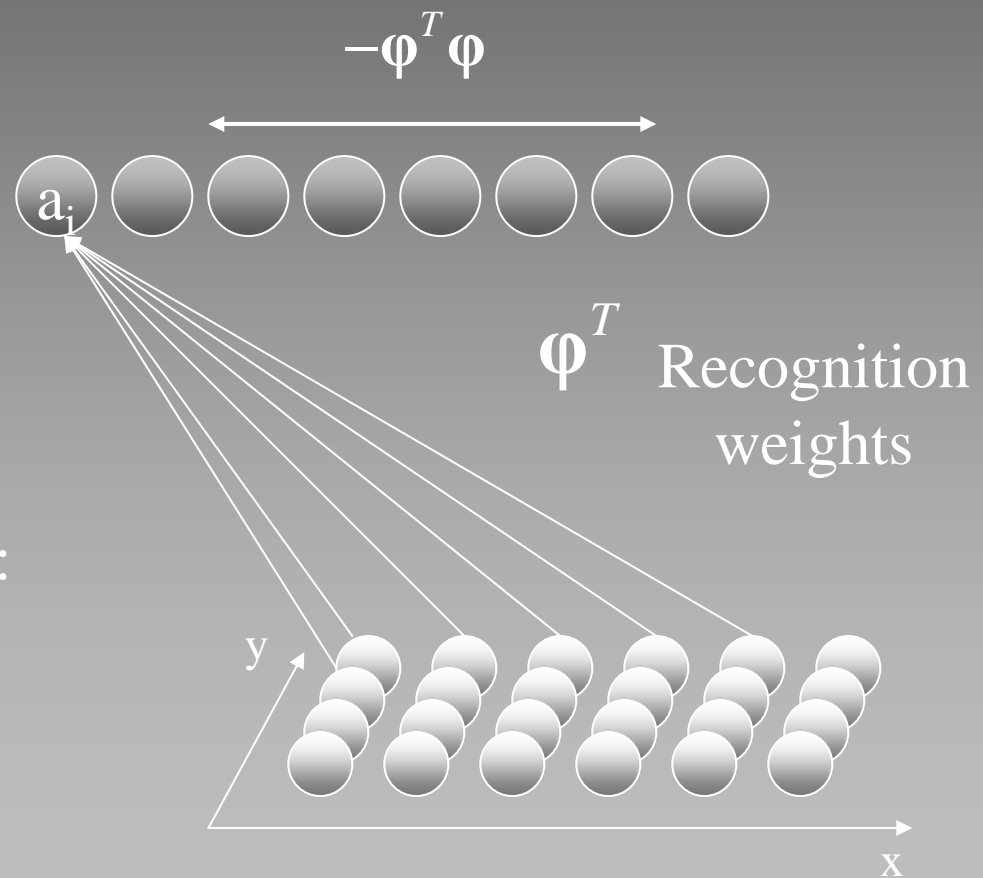
Network implementation

$$\tau \frac{d\mathbf{a}}{dt} = -\boldsymbol{\Phi}^T \boldsymbol{\Phi} \mathbf{a} + \boldsymbol{\Phi}^T \mathbf{I} + f'(\mathbf{a})$$

special case (Cauchy prior):

$$g'(a_i) = -a_i$$

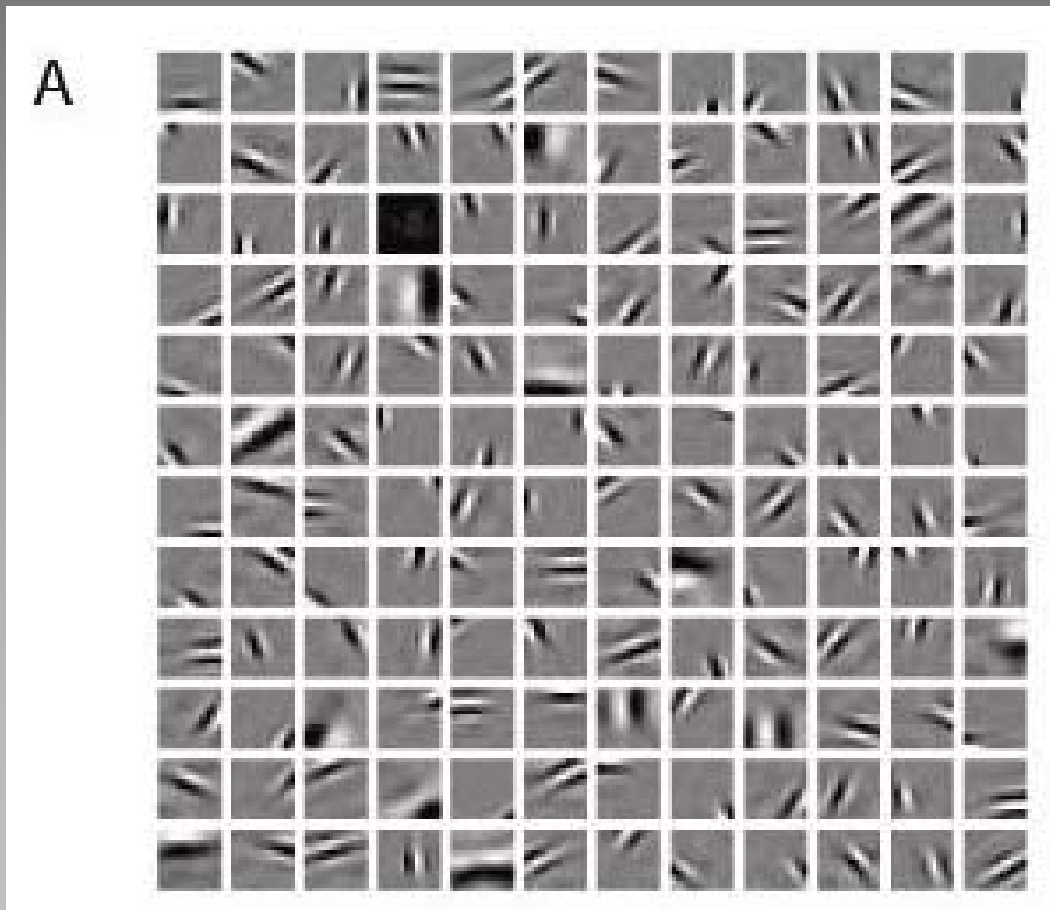
$$\tau \frac{d\mathbf{a}}{dt} = -\mathbf{a} - \boldsymbol{\Phi}^T \boldsymbol{\Phi} \mathbf{a} + \boldsymbol{\Phi}^T \mathbf{I}$$



Sparse representations in V1

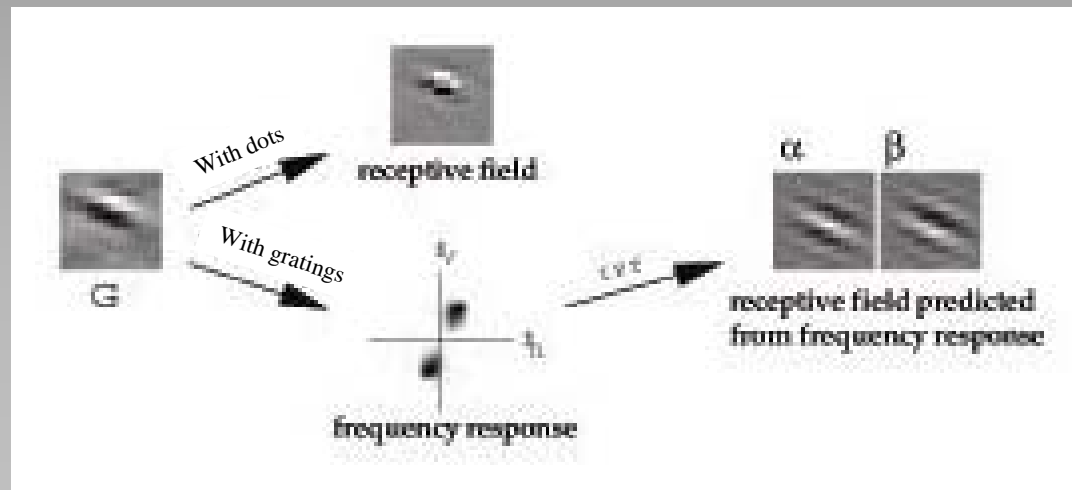
- The sparse prior favors patterns of activity for which most neurons are silent and a few are highly active.

Projective fields



Sparse representations in V1

- The true receptive fields are input dependent (because of the lateral interactions) in a way that seems somewhat consistent with experimental data



Infomax idea

- Represent the world in a format that maximizes mutual information given the limited information capacity of neurons.

$$I(R, S) = H(R) - H(R | S)$$

- Is this simply about packing bits in neuronal firing?
- What if the code is undecipherable?

Information theory and learning

- The features extracted by infomax algorithms are often meaningful because high level features are often good for compression.
- Example of scanned text: a page of text can be dramatically compressed if one treats it as a sequence of characters as opposed to pixels (e.g. this page: 800x700x8 vs 200x8, 2400 compression factor)
- General idea of unsupervised learning: compress the image and hope to discover high order description of the image

Information theory and learning

- Ex: Decorrelation in the retina leads to center-surround receptive fields.
- Ex: ICA (factorial code) leads to oriented receptive fields.
- Problem: what can you do beyond ICA?
- How can you extract features that simplify computation?
- We need other constraints...

Sparse Coding

- Ex: sparseness... Why sparseness?
- Grandmother cells: very easy to decode and very easy to use for further computation.
- Sparse is non gaussian which often corresponds to high level features (because it goes against the law of large number)

Learning Representations

- The main challenges for the future:
 - Representing hierarchical structure
 - Learning hierarchical structure