Neural networks:
Unsupervised learning

Previously

The supervised learning paradigm:

given example inputs **x** and target outputs *t* learning the mapping between them

the trained network is supposed to give 'correct response' for *any given* input stimulus

training is equivalent of learning the appropriate weights

to achiece this an *objective function* (or *error function*) is defined, which is minimized during training

Previously

Optimization wrt. an objective function

$$M(\mathbf{w}) = G(\mathbf{w}) + \alpha E_W(\mathbf{w})$$

where

$$G(\mathbf{w}) = -\sum_n \left[t^{(n)} \ln y(\mathbf{x}^{(n)}; \mathbf{w}) + (1 - t^{(n)}) \ln(1 - y(\mathbf{x}^{(n)}; \mathbf{w})) \right]$$

(error function)

$$E_W(\mathbf{w}) = \frac{1}{2} \sum_i w_i^2.$$

(regularizer)

Previously

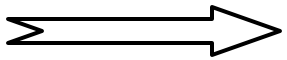
Interpret $y(\mathbf{x},\mathbf{w})$ as a probability:

$$\begin{aligned} P(t=1\,|\,\mathbf{w},\mathbf{x}) &= y \\ P(t=0\,|\,\mathbf{w},\mathbf{x}) &= 1-y \end{aligned}$$

Previously

Interpret $y(\mathbf{x},\mathbf{w})$ as a probability:

$$P(t=1\,|\,\mathbf{w},\mathbf{x}) = y$$
$$P(t=0\,|\,\mathbf{w},\mathbf{x}) = 1-y$$



the **likelihood** of the input data can be expressed with the original error function function

$$P(D\,|\,\mathbf{w}) = \exp[-G(\mathbf{w})]$$

Previously

Interpret $y(\mathbf{x},\mathbf{w})$ as a probability:

$$P(t=1\,|\,\mathbf{w},\mathbf{x}) = y$$
$$P(t=0\,|\,\mathbf{w},\mathbf{x}) = 1-y$$

⟹ the **likelihood** of the input data can be expressed with the original error function function

$$P(D\,|\,\mathbf{w}) = \exp[-G(\mathbf{w})]$$

⟹ the regularizer has the form of a prior!

$$P(\mathbf{w}\,|\,\alpha) = \frac{1}{Z_W(\alpha)}\exp(-\alpha E_W).$$

Previously

Interpret $y(\mathbf{x},\mathbf{w})$ as a probability:

$$P(t=1\,|\,\mathbf{w},\mathbf{x}) = y$$
$$P(t=0\,|\,\mathbf{w},\mathbf{x}) = 1-y$$

⟹ the **likelihood** of the input data can be expressed with the original error function function

$$P(D\,|\,\mathbf{w}) = \exp[-G(\mathbf{w})]$$

⟹ the regularizer has the form of a prior!

$$P(\mathbf{w}\,|\,\alpha) = \frac{1}{Z_W(\alpha)}\exp(-\alpha E_W).$$

what we get in the objective function **M(w)**:

the posterior distribution of **w**: $P(\mathbf{w}\,|\,D,\alpha) = \dfrac{P(D\,|\,\mathbf{w})P(\mathbf{w}\,|\,\alpha)}{P(D\,|\,\alpha)}$

Previously

Interpret $y(\mathbf{x},\mathbf{w})$ as a probability:

$$P(t=1\,|\,\mathbf{w},\mathbf{x}) = y$$
$$P(t=0\,|\,\mathbf{w},\mathbf{x}) = 1-y$$

⟹ the **likelihood** of the input data can be expressed with the original error function function

$$P(D\,|\,\mathbf{w}) = \exp[-G(\mathbf{w})]$$

⟹ the regularizer has the form of a prior!

$$P(\mathbf{w}\,|\,\alpha) = \frac{1}{Z_W(\alpha)}\exp(-\alpha E_W).$$

what we get in the objective function M(**w**):

the posterior distribution of **w**: $P(\mathbf{w}\,|\,D,\alpha) = \dfrac{P(D\,|\,\mathbf{w})P(\mathbf{w}\,|\,\alpha)}{P(D\,|\,\alpha)}$

The neuron's behavior is faithfully translated into probabilistic terms!

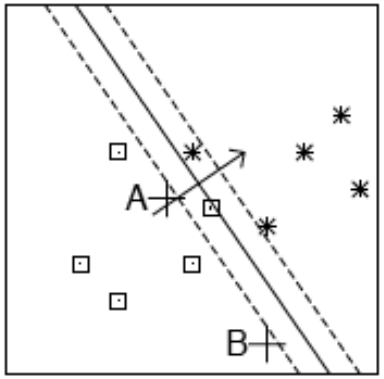
Previously

When making predictions....

Previously

When making predictions....

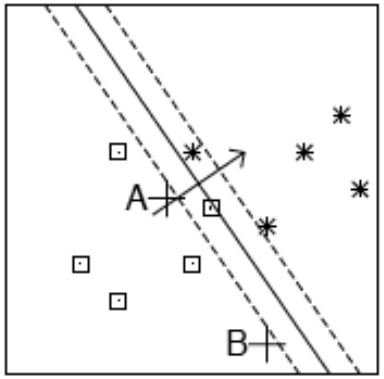
Original estimate



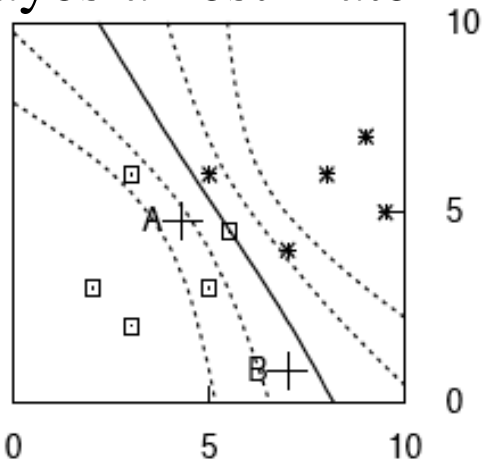
Previously

When making predictions....

Original estimate



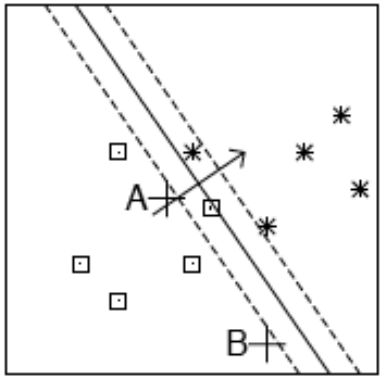
Bayesian estimate



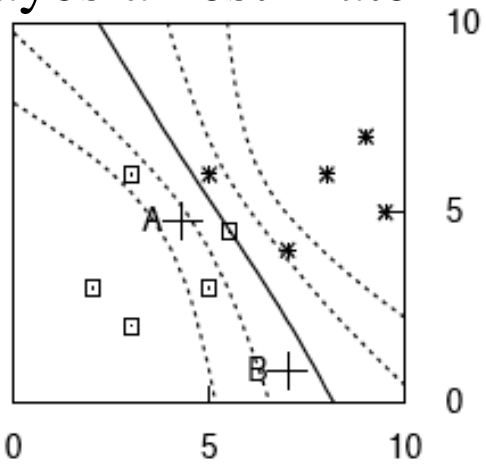
Previously

When making predictions....

Original estimate



Bayesian estimate



- The probabilistic interpretation makes our assumptions explicit:

  by the regularizer we imposed a soft constraint on the learned parameters, which expresses our *prior expecations*.

- An additional plus:

  beyond getting $\mathbf{w}_{\mathbf{MP}}$ we get a measure for learned parameter uncertainty

What's coming?

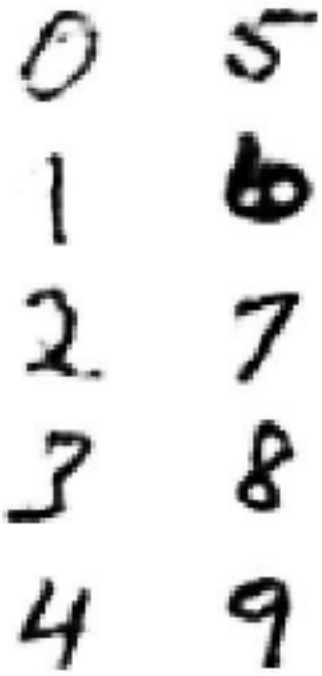
- Networks & probabilistic framework:
  from the Hopfield network to Boltzmann machine
- What we learn?

  Density estimation, neural architecture and optimization principles: principal component analysis (PCA)
- How we learn?

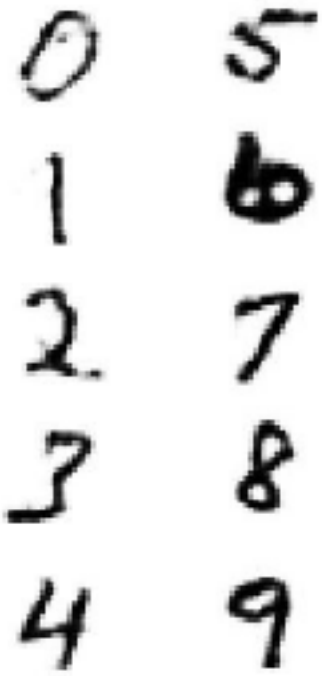
  Hebb et al: Learning rules
- Any biology?

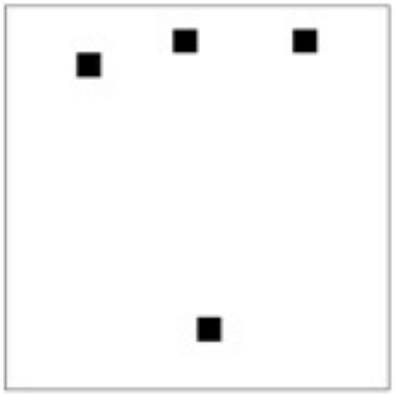
  Simple cells & ICA

Learning data...



Learning data...





Capacity of a single neuron is limited: certain data can only be learned
So far, we used a supervised learning paradigm: a teacher was necessary to teach an input-output relation

Hopfield networks try to cure both

Unsupervised learning: what is it about?

Hebb rule: an enlightening example

assuming 2 neurons and a weight modification process:

$$\frac{\mathrm{d}w_{ij}}{\mathrm{d}t} \sim \mathrm{Correlation}(x_i, x_j)$$

This simple rule realizes an associative memory!

Architecture: a set of I neurons

connected by *symmetric* synapses of weight w_{ij}

no self connections: $w_{ii}=0$

output of neuron i: x_i

Activity rule:

$$x(a) = \Theta(a) \equiv \begin{cases} 1 & a \geq 0 \\ -1 & a < 0. \end{cases}$$

Synchronous/ asynchronous update

Learning rule:

$$w_{ij} = \eta \sum_n x_i^{(n)} x_j^{(n)},$$

;

Architecture: a set of I neurons

connected by *symmetric* synapses of weight w_{ij}

no self connections: $w_{ii}=0$

output of neuron i: x_i

Activity rule:

$$x(a) = \Theta(a) \equiv \begin{cases} 1 & a \geq 0 \\ -1 & a < 0. \end{cases}$$

Synchronous/ asynchronous update

Learning rule:

$$w_{ij} = \eta \sum_n x_i^{(n)} x_j^{(n)},$$

alternatively, a continuous network can be defined as:

$$a_i = \sum_j w_{ij} x_j \; ; \quad x_i = \tanh(a_i).$$

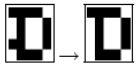
Are the memories stable? the activation and activity rule together define a Lyapunov function

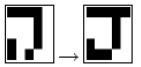
$$E(x, \mathbf{w}) = -\frac{1}{2}\sum_{m,n} w_{mn} x_m x_n - \sum_n w_{0n} x_n$$

Necessary conditions: symmetric weights; asynchronous update

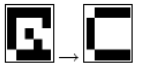
(a) 

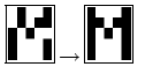
.	0	0	0	0	-2	2	-2	2	2	-2	0	0	0	2	0	0	-2	0	2	2	0	0	-2	-2
0	.	4	4	0	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2	
0	4	.	4	0	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2	
0	4	4	.	0	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2	
0	0	0	0	.	2	-2	-2	2	-2	2	-4	0	0	-2	4	-4	-2	0	-2	2	0	0	-2	2
-2	-2	-2	-2	2	.	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
2	-2	-2	-2	-2	0	.	0	0	4	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	-2	0	0	.	0	0	0	2	2	2	0	-2	2	4	2	0	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	.	0	0	-2	2	2	0	2	-2	0	2	0	4	-2	-2	-4	0
2	-2	-2	-2	-2	0	4	0	0	.	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	2	4	0	0	0	0	.	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
0	0	0	0	-4	-2	2	2	-2	2	-2	.	0	0	2	-4	4	2	0	2	-2	0	0	2	-2
0	-4	-4	-4	0	2	2	2	2	2	2	0	.	0	2	0	0	2	0	2	2	-4	-4	-2	2
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	.	-2	0	0	2	4	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	.	-2	2	0	-2	4	0	-2	-2	0	0
0	0	0	0	4	2	-2	-2	2	-2	2	-4	0	0	-2	.	-4	-2	0	-2	2	0	0	-2	2
0	0	0	0	-4	-2	2	2	-2	2	-2	4	0	0	2	-4	.	2	0	2	-2	0	0	2	-2
-2	-2	-2	-2	-2	0	0	4	0	0	0	2	2	2	0	-2	2	.	2	0	0	-2	-2	0	0
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	4	-2	0	0	2	.	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	4	-2	2	0	-2	.	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	0	4	0	0	-2	2	2	0	2	-2	0	2	0	.	-2	-2	-4	0
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	.	4	2	-2
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	.	2	-2
-2	2	2	2	-2	0	0	0	-4	0	0	2	-2	-2	0	-2	2	0	-2	0	-4	2	2	.	0
-2	-2	-2	-2	2	4	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	

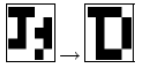
(b) 

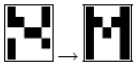
(c) 

(d) 

(e) 

(f) 

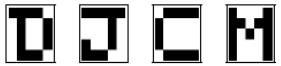
(g) 

(h) 

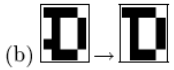
Are the memories stable? the activation and activity rule together define a Lyapunov function

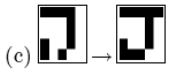
$$E(x, \mathbf{w}) = -\frac{1}{2}\sum_{m,n} w_{mn} x_m x_n - \sum_n w_{0n} x_n$$

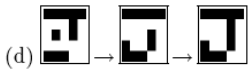
Necessary conditions: symmetric weights; asynchronous update

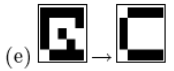
(a) 

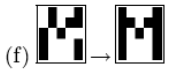
.	0	0	0	0	-2	2	-2	2	2	-2	0	0	0	2	0	0	-2	0	2	2	0	0	-2	-2
0	.	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	4	.	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	4	4	.	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	0	0	0	.	2	-2	-2	2	-2	2	-4	0	0	-2	4	-4	-2	0	-2	2	0	0	-2	2
-2	-2	-2	-2	2	.	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
2	-2	-2	-2	-2	0	.	0	0	4	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	-2	0	0	.	0	0	0	2	2	2	0	-2	2	4	2	0	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	.	0	0	-2	2	2	0	2	-2	0	2	0	4	-2	-2	-4	0
2	-2	-2	-2	-2	0	4	0	0	.	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	2	4	0	0	0	0	.	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
0	0	0	0	-4	-2	2	2	-2	2	-2	.	0	0	2	-4	4	2	0	2	-2	0	0	2	-2
0	-4	-4	-4	0	2	2	2	2	2	2	0	.	0	2	0	0	2	0	2	2	-4	-4	-2	2
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	.	-2	0	0	2	4	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	.	-2	2	0	-2	4	0	-2	-2	0	0
0	0	0	0	4	2	-2	-2	2	-2	2	-4	0	0	-2	.	-4	-2	0	-2	2	0	0	-2	2
0	0	0	0	-4	-2	2	2	-2	2	-2	4	0	0	2	-4	.	2	0	2	-2	0	0	2	-2
-2	-2	-2	-2	-2	0	0	4	0	0	0	2	2	2	0	-2	2	.	2	0	0	-2	-2	0	0
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	4	-2	0	0	2	.	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	4	-2	2	0	-2	.	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	0	4	0	0	-2	2	2	0	2	-2	0	2	0	.	-2	-2	-4	0
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	.	4	2	-2
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	.	2	-2
-2	2	2	2	-2	0	0	0	-4	0	0	2	-2	-2	0	-2	2	0	-2	0	-4	2	2	.	0
-2	-2	-2	-2	2	4	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	.

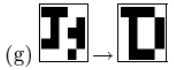
(b) 

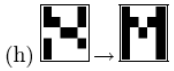
(c) 

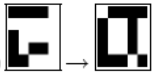
(d) 

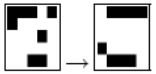
(e) 

(f) 

(g) 

(h) 

(i) 

(j) 

(k) 

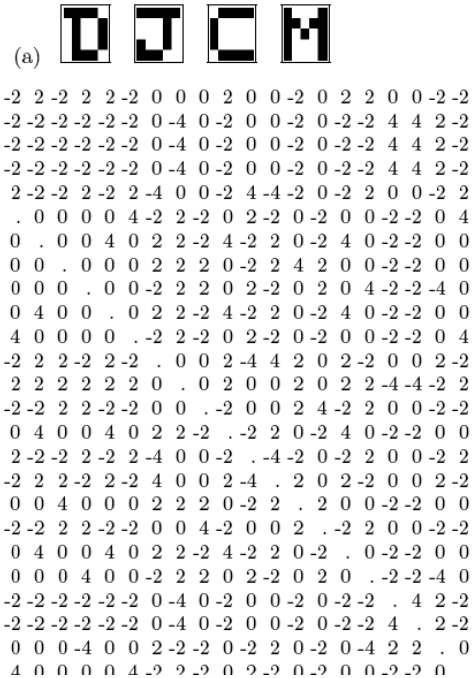
(l) 

(m) 

Are the memories stable? the activation and activity rule together define a Lyapunov function

$$E(x, \mathbf{w}) = -\frac{1}{2}\sum_{m,n} w_{mn} x_m x_n - \sum_n w_{0n} x_n$$

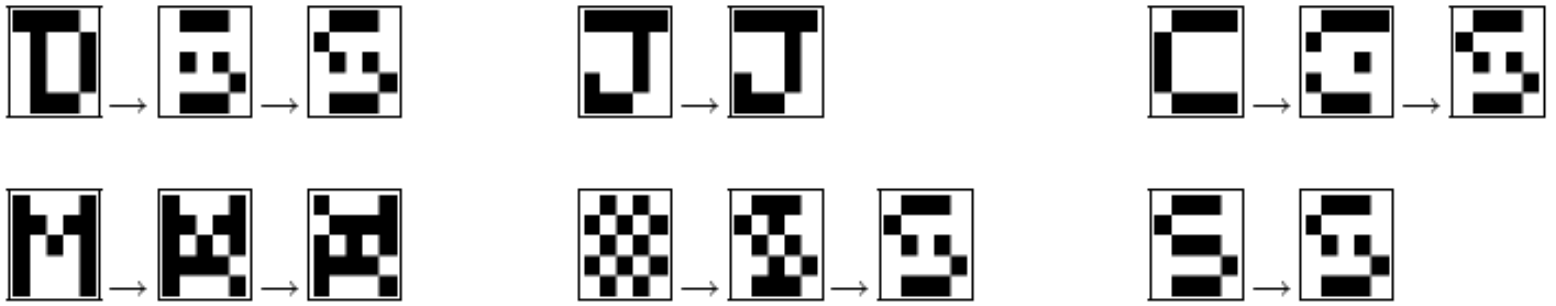
Necessary conditions: symmetric weights; asynchronous update



Robust against perturbation of a subset of weights

How many traces can be memorized by a network of *I* neurons?

Desired memories:



Failures of the Hopfield networks:

- Corrupted bits
- Missing memory traces
- Spurious states not directly related to training data

Activation rule: $$a_i = \sum_j w_{ij} x_j^{(n)},$$

Trace of the ‘desired memory and additional random memories:

$$w_{ij} = x_i^{(n)} x_j^{(n)} + \sum_{m \neq n} x_i^{(m)} x_j^{(m)}.$$

Activation rule: $$a_i = \sum_j w_{ij} x_j^{(n)},$$

Trace of the 'desired memory and additional random memories:

$$w_{ij} = x_i^{(n)} x_j^{(n)} + \sum_{m \neq n} x_i^{(m)} x_j^{(m)}.$$

$$\begin{aligned} a_i &= \sum_{j \neq i} x_i^{(n)} x_j^{(n)} x_j^{(n)} + \sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)} \\ &= (I-1) x_i^{(n)} + \sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)}. \end{aligned}$$

Activation rule: $$a_i = \sum_j w_{ij} x_j^{(n)},$$

Trace of the ‘desired memory and additional random memories:

$$w_{ij} = x_i^{(n)} x_j^{(n)} + \sum_{m \neq n} x_i^{(m)} x_j^{(m)}.$$

$$\begin{aligned} a_i &= \sum_{j \neq i} x_i^{(n)} x_j^{(n)} x_j^{(n)} + \sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)} \\ &= (I-1) x_i^{(n)} + \sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)}. \end{aligned}$$

desired state

Activation rule: $$a_i = \sum_j w_{ij} x_j^{(n)},$$

Trace of the ‘desired memory and additional random memories:

$$w_{ij} = x_i^{(n)} x_j^{(n)} + \sum_{m \neq n} x_i^{(m)} x_j^{(m)}.$$

$$\begin{aligned} a_i &= \sum_{j \neq i} x_i^{(n)} x_j^{(n)} x_j^{(n)} + \sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)} \\ &= (I-1) x_i^{(n)} + \sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)}. \end{aligned}$$

desired state

random contribution

Activation rule: $$a_i = \sum_j w_{ij} x_j^{(n)},$$

Trace of the 'desired memory and additional random memories:

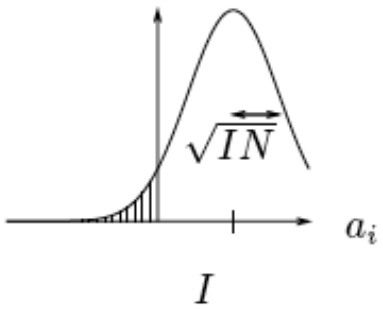
$$w_{ij} = x_i^{(n)} x_j^{(n)} + \sum_{m \neq n} x_i^{(m)} x_j^{(m)}.$$

$$\Longrightarrow \quad a_i = \sum_{j \neq i} x_i^{(n)} x_j^{(n)} x_j^{(n)} + \sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)}$$

$$= \boxed{(I-1) x_i^{(n)}} + \boxed{\sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)}}.$$

desired state

random contribution



$$P(i \text{ unstable}) = \Phi\left(-\frac{I}{\sqrt{IN}}\right) = \Phi\left(-\frac{1}{\sqrt{N/I}}\right),$$

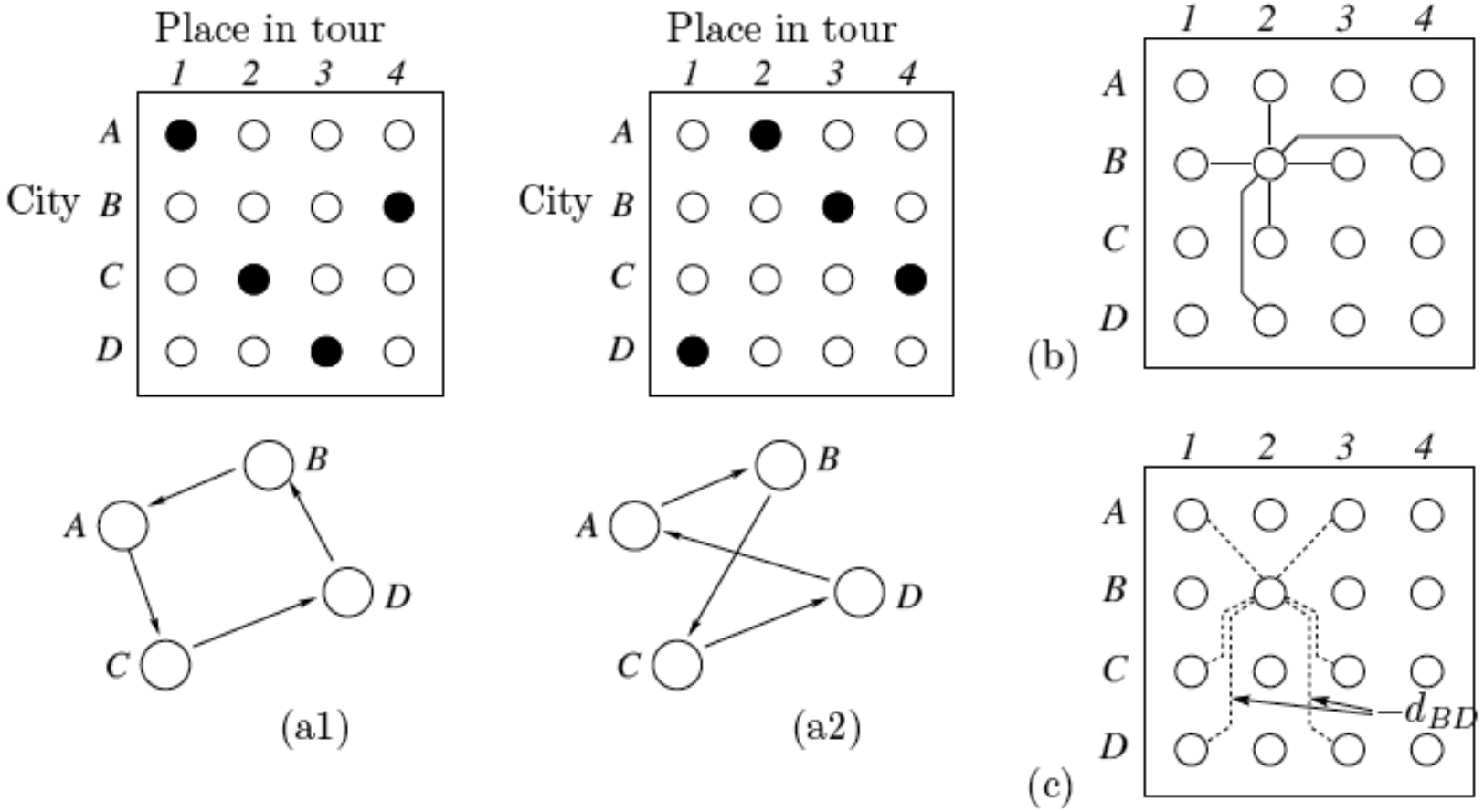
Failure in operation: avalanches

$$N_{\text{crit}} = 0.138I.$$

- N/I < 0.138: 'spin glass states'
- N/I $\in$ (0 0.138): states close to desired memories
- N/I $\in$ (0 0.05): desired states have lower energy than spurious states
- N/I $\in$ (0.05 0.138): spurious states dominate
- N/I $\in$ (0 0.03): mixture states

The Hebb rule determines how well it performs
other learning might do a better job
(reiforcement learning)

Hopfield network for optimization



The Boltzmann machine

The optimization performed by Hopfield network:

minimizing $E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}$

The Boltzmann machine

The optimization performed by Hopfield network:

minimizing $E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}$

Again: we can make a correspondence with a probabilistic model:

$$P(\mathbf{x} \mid \mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp[-E(\mathbf{x})] = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

The Boltzmann machine

The optimization performed by Hopfield network:

minimizing $E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}$

Again: we can make a correspondence with a probabilistic model:

$$P(\mathbf{x} \mid \mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp[-E(\mathbf{x})] = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

What do we gain by this:

- more transparent functioning
- superior performance than Hebb rule

The Boltzmann machine

The optimization performed by Hopfield network:

minimizing $E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}$

Again: we can make a correspondence with a probabilistic model:

$$P(\mathbf{x} \mid \mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp[-E(\mathbf{x})] = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

What do we gain by this:

- more transparent functioning
- superior performance than Hebb rule

Activity rule: set $x_i = +1$ with probability $\frac{1}{1 + e^{-2a_i}}$
else set $x_i = -1$.

The Boltzmann machine

The optimization performed by Hopfield network:

minimizing $E(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}$

Again: we can make a correspondence with a probabilistic model:

$$P(\mathbf{x} \mid \mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp[-E(\mathbf{x})] = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

What do we gain by this:

- more transparent functioning
- superior performance than Hebb rule

Activity rule: set $x_i = +1$ with probability $\frac{1}{1+e^{-2a_i}}$
else set $x_i = -1$.

How is learning performed?

Boltzmann machine -- EM

Likelihood function: $$P(\mathbf{x}|\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

Boltzmann machine -- EM

Likelihood function: $$P(\mathbf{x}|\,\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

Estimating the parameters: $$P(\mathbf{W}\,|\,\{\mathbf{x}^{(n)}\}_1^N\}) = \frac{\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] P(\mathbf{W})}{P(\{\mathbf{x}^{(n)}\}_1^N\})}.$$

$$\ln\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] = \sum_{n=1}^{N}\left[\frac{1}{2}\mathbf{x}^{(n)\mathsf{T}}\mathbf{W}\mathbf{x}^{(n)} - \ln Z(\mathbf{W})\right]$$

Boltzmann machine -- EM

Likelihood function: $$P(\mathbf{x}\,|\,\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

Estimating the parameters: $$P(\mathbf{W}\,|\,\{\mathbf{x}^{(n)}\}_1^N\}) = \frac{\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] P(\mathbf{W})}{P(\{\mathbf{x}^{(n)}\}_1^N\})}.$$

$$\ln\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] = \sum_{n=1}^{N}\left[\frac{1}{2}\mathbf{x}^{(n)\mathsf{T}}\mathbf{W}\mathbf{x}^{(n)} - \ln Z(\mathbf{W})\right]$$

Minimizing for *w*: $$\begin{aligned}\frac{\partial}{\partial w_{ij}} \ln P(\{\mathbf{x}^{(n)}\}_1^N\}\,|\,\mathbf{W}) &= \sum_{n=1}^{N}\left[x_i^{(n)}x_j^{(n)} - \langle x_i x_j\rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right] \\ &= N\left[\langle x_i x_j\rangle_{\text{Data}} - \langle x_i x_j\rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right]\end{aligned}$$

Boltzmann machine -- EM

Likelihood function: $$P(\mathbf{x}\,|\,\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

Estimating the parameters: $$P(\mathbf{W}\,|\,\{\mathbf{x}^{(n)}\}_1^N\}) = \frac{\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] P(\mathbf{W})}{P(\{\mathbf{x}^{(n)}\}_1^N\})}.$$

$$\ln\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] = \sum_{n=1}^{N}\left[\frac{1}{2}\mathbf{x}^{(n)\mathsf{T}}\mathbf{W}\mathbf{x}^{(n)} - \ln Z(\mathbf{W})\right]$$

Minimizing for *w*: $$\begin{aligned}\frac{\partial}{\partial w_{ij}} \ln P(\{\mathbf{x}^{(n)}\}_1^N\}\,|\,\mathbf{W}) &= \sum_{n=1}^{N}\left[x_i^{(n)}x_j^{(n)} - \langle x_i x_j\rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right] \\ &= N\left[\langle x_i x_j\rangle_{\text{Data}} - \langle x_i x_j\rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right]\end{aligned}$$

Sleeping and waking:

Boltzmann machine -- EM

Likelihood function: $$P(\mathbf{x}\,|\,\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

Estimating the parameters: $$P(\mathbf{W}\,|\,\{\mathbf{x}^{(n)}\}_1^N\}) = \frac{\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] P(\mathbf{W})}{P(\{\mathbf{x}^{(n)}\}_1^N\})}.$$

$$\ln\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] = \sum_{n=1}^{N}\left[\frac{1}{2}\mathbf{x}^{(n)\mathsf{T}}\mathbf{W}\mathbf{x}^{(n)} - \ln Z(\mathbf{W})\right]$$

Minimizing for *w*: $$\begin{aligned}\frac{\partial}{\partial w_{ij}} \ln P(\{\mathbf{x}^{(n)}\}_1^N\}\,|\,\mathbf{W}) &= \sum_{n=1}^{N}\left[x_i^{(n)}x_j^{(n)} - \langle x_i x_j\rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right] \\ &= N\left[\langle x_i x_j\rangle_{\text{Data}} - \langle x_i x_j\rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right]\end{aligned}$$

Sleeping and waking: $$\langle x_i x_j\rangle_{\text{Data}} \equiv \frac{1}{N}\sum_{n=1}^{N}\left[x_i^{(n)}x_j^{(n)}\right]$$

Boltzmann machine -- EM

Likelihood function: $$P(\mathbf{x}\,|\,\mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp\left[\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{W}\mathbf{x}\right]$$

Estimating the parameters: $$P(\mathbf{W}\,|\,\{\mathbf{x}^{(n)}\}_1^N\}) = \frac{\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] P(\mathbf{W})}{P(\{\mathbf{x}^{(n)}\}_1^N\})}.$$

$$\ln\left[\prod_{n=1}^{N} P(\mathbf{x}^{(n)}\,|\,\mathbf{W})\right] = \sum_{n=1}^{N}\left[\frac{1}{2}\mathbf{x}^{(n)\mathsf{T}}\mathbf{W}\mathbf{x}^{(n)} - \ln Z(\mathbf{W})\right]$$

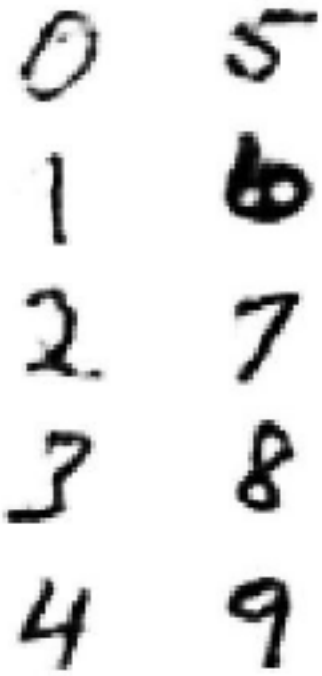
Minimizing for *w*: $$\begin{aligned}\frac{\partial}{\partial w_{ij}} \ln P(\{\mathbf{x}^{(n)}\}_1^N\}\,|\,\mathbf{W}) &= \sum_{n=1}^{N}\left[x_i^{(n)}x_j^{(n)} - \langle x_i x_j\rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right] \\ &= N\left[\langle x_i x_j\rangle_{\text{Data}} - \langle x_i x_j\rangle_{P(\mathbf{x}\,|\,\mathbf{W})}\right]\end{aligned}$$

Sleeping and waking:

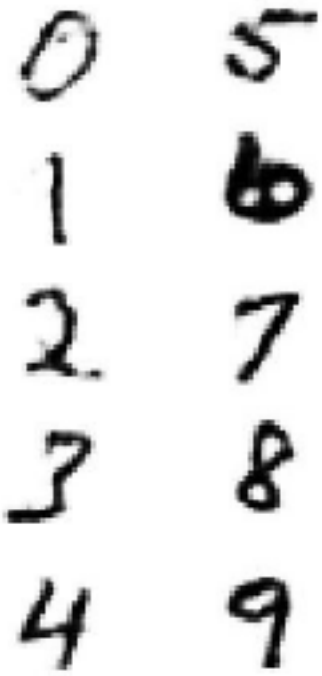
$$\langle x_i x_j\rangle_{\text{Data}} \equiv \frac{1}{N}\sum_{n=1}^{N}\left[x_i^{(n)}x_j^{(n)}\right]$$

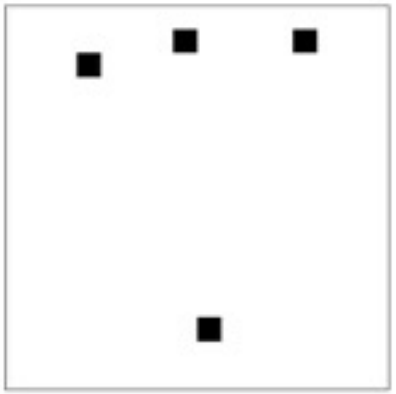
$$\langle x_i x_j\rangle_{P(\mathbf{x}\,|\,\mathbf{W})} \equiv \sum_{\mathbf{x}} x_i x_j P(\mathbf{x}\,|\,\mathbf{W})$$

Learning data...



Learning data...





Summary

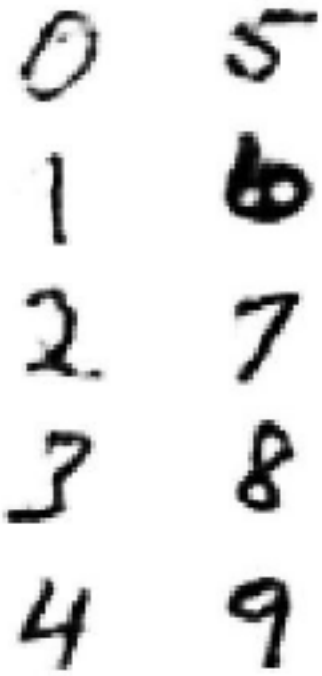
Boltmann translates the neural network mecanisms into a probablisitic framework

Its capabilities are limited

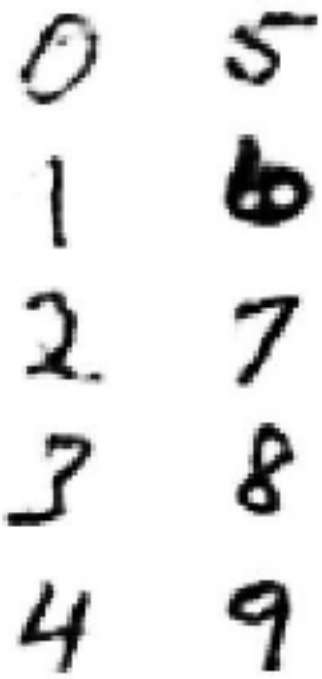
We learned that the probabilistic framework clarifies assumptions

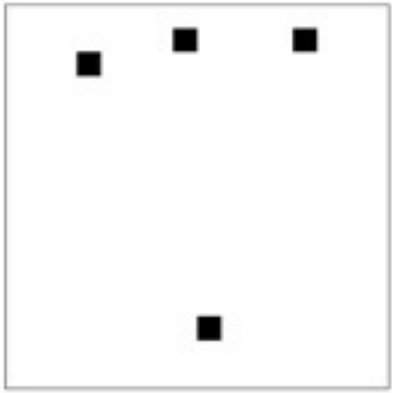
We learned that within the world constrained by our assumptions the probabilistic approach gives clear answers

Learning data...

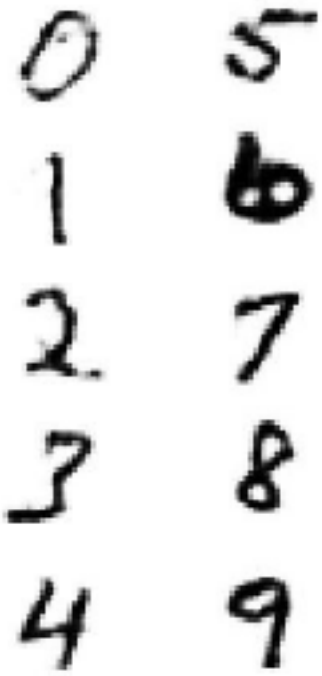


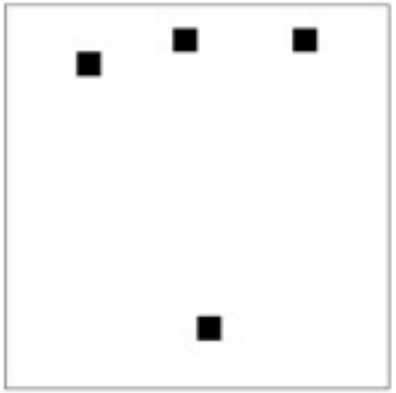
Learning data...



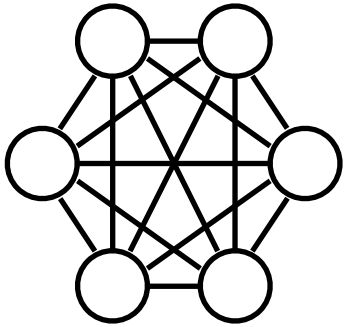


Learning data...

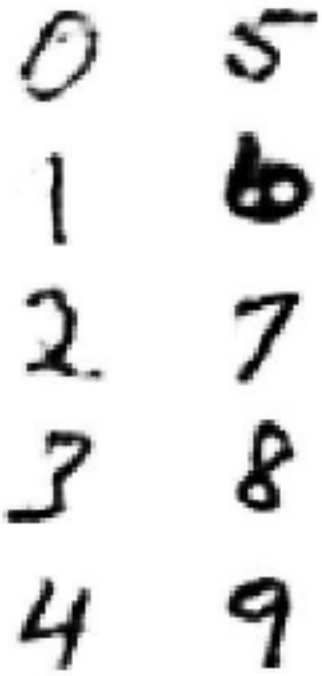


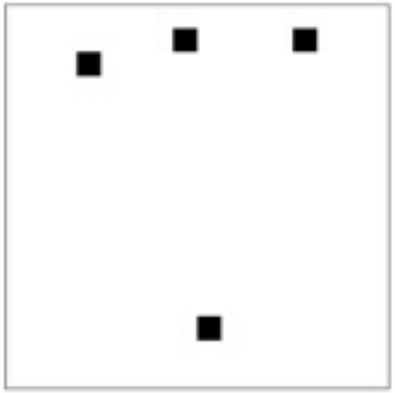


Hopfield/Boltman

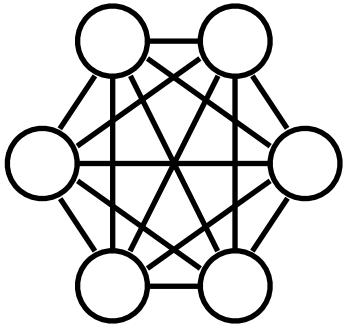


Learning data...

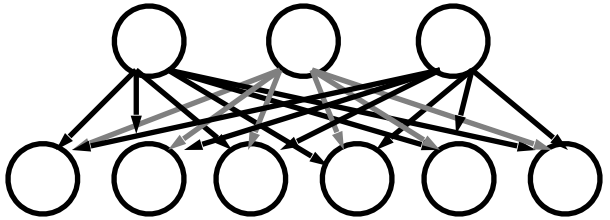




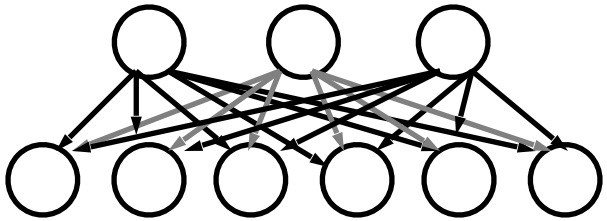
Hopfield/Boltman



?

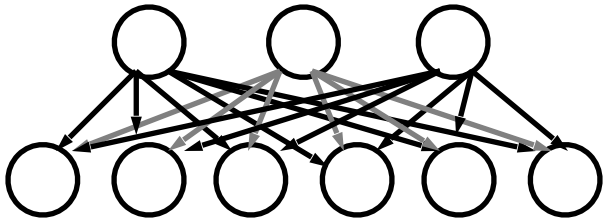


Principal Component Analysis



$$v = \mathbf{w} \cdot \mathbf{u}$$

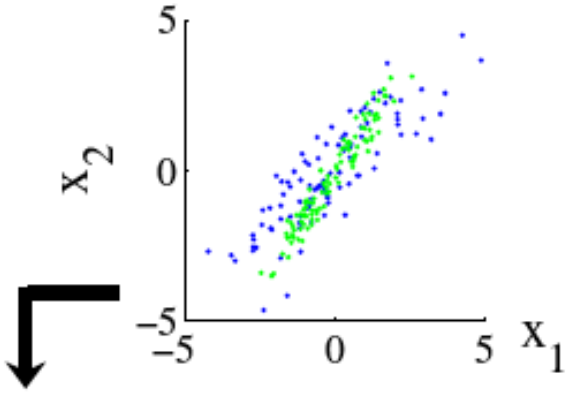
Principal Component Analysis



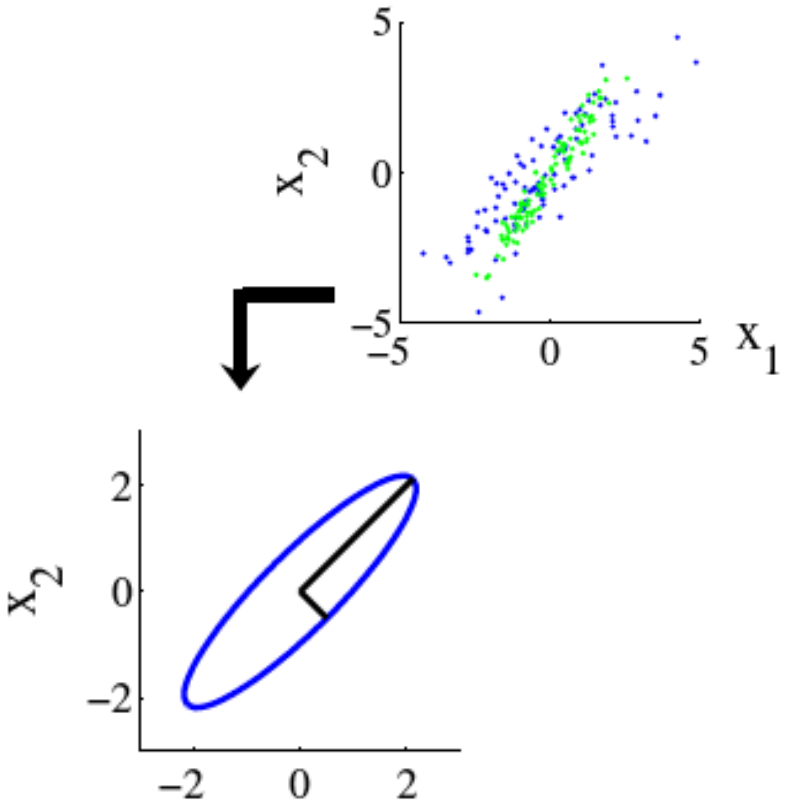
$$v = \mathbf{w} \cdot \mathbf{u}$$

Let's try to find linearly independent filters
Set the basis along the eigenvectors of the data

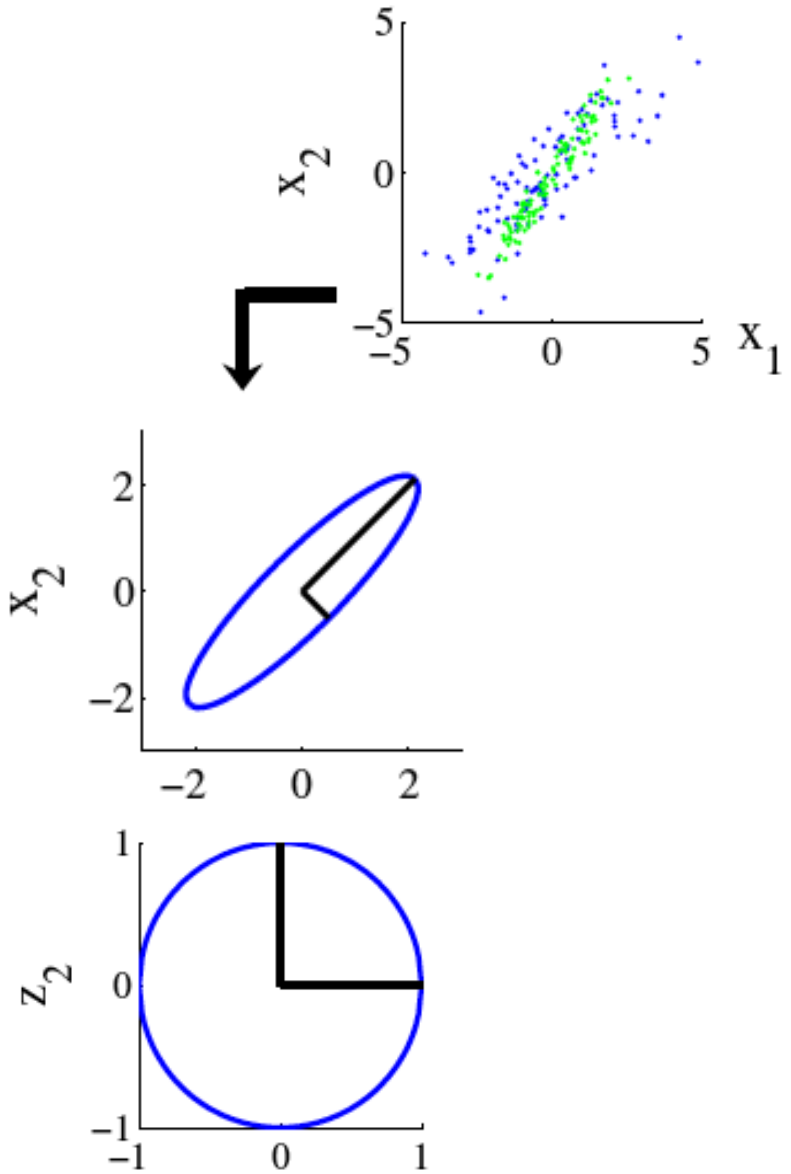
Principal Component Analysis



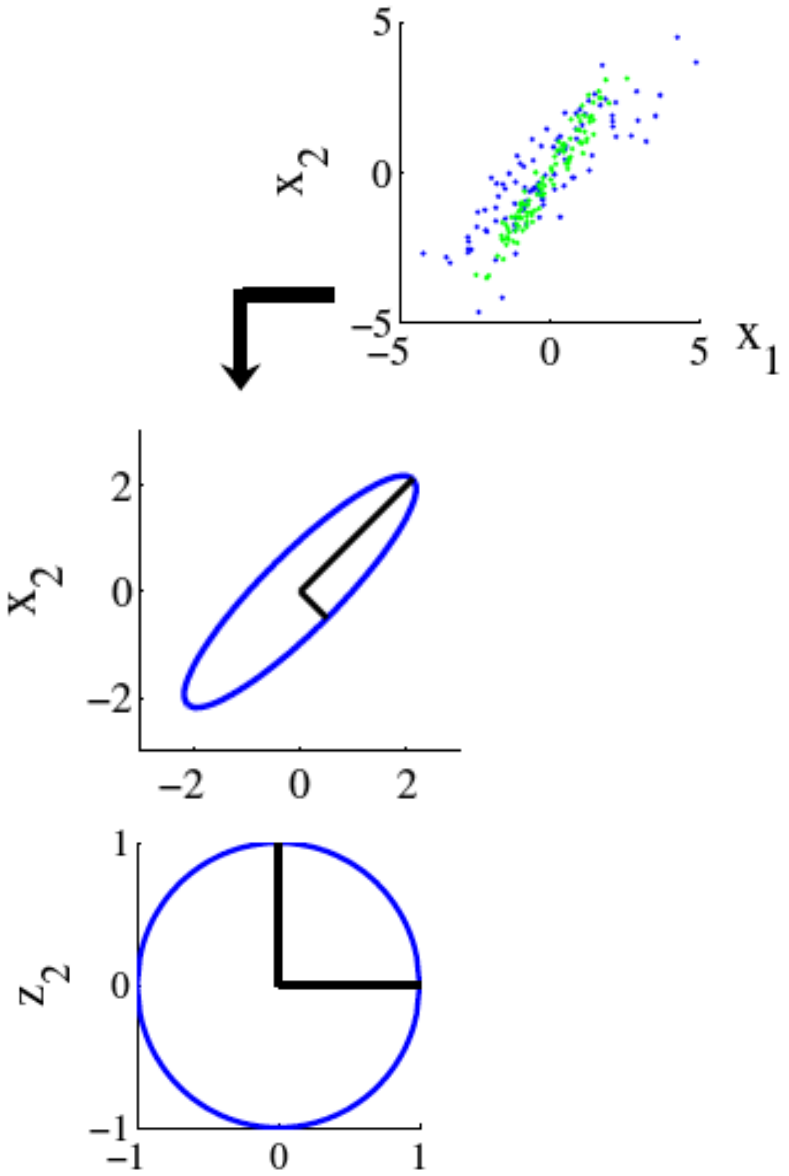
Principal Component Analysis

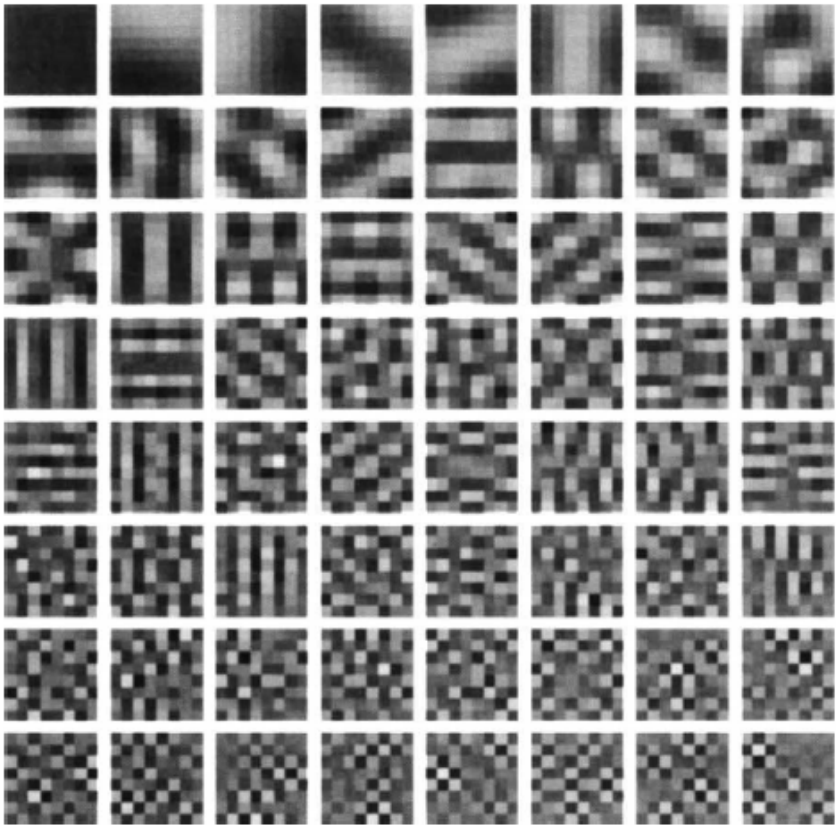


Principal Component Analysis



Principal Component Analysis





Olshausen & Field, Nature (1996)

Relation between PCA and learning rules

A single neuron driven by multiple inputs: $v = \mathbf{w} \cdot \mathbf{u}$

Basic Hebb rule: $\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u}$

Averaged Hebb rule: $\tau_w \frac{d\mathbf{w}}{dt} = \langle v\mathbf{u} \rangle$

Correlation based rule: $\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{Q} \cdot \mathbf{w}$ or $\tau_w \frac{dw_b}{dt} = \sum_{b'=1}^{N_u} Q_{bb'} w_{b'}$

note that $v = \mathbf{w} \cdot \mathbf{u}$

Relation between PCA and learning rules

Making possible both LTP and LTD

Postsynaptic threshold $\tau_w \frac{d\mathbf{w}}{dt} = (v - \theta_v)\mathbf{u}$

Postsynaptic threshold $\tau_w \frac{d\mathbf{w}}{dt} = v(\mathbf{u} - \boldsymbol{\theta}_u)$

Setting the threshold to average postsynaptic activity: $\theta_v = \langle v \rangle$

$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{C} \cdot \mathbf{w}$, where

$$\mathbf{C} = \langle (\mathbf{u} - \langle \mathbf{u} \rangle)(\mathbf{u} - \langle \mathbf{u} \rangle) \rangle = \langle \mathbf{uu} \rangle - \langle \mathbf{u} \rangle^2 = \langle (\mathbf{u} - \langle \mathbf{u} \rangle)\mathbf{u} \rangle$$

Relation between PCA and learning rules

Making possible both LTP and LTD

Postsynaptic threshold $\tau_w \frac{d\mathbf{w}}{dt} = (v - \theta_v)\mathbf{u}$

homosynaptic depression

Postsynaptic threshold $\tau_w \frac{d\mathbf{w}}{dt} = v(\mathbf{u} - \boldsymbol{\theta}_u)$

heterosynaptic depression

Setting the threshold to average postsynaptic activity: $\theta_v = \langle v \rangle$

$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{C} \cdot \mathbf{w}$, where

$$\mathbf{C} = \langle (\mathbf{u} - \langle \mathbf{u} \rangle)(\mathbf{u} - \langle \mathbf{u} \rangle) \rangle = \langle \mathbf{uu} \rangle - \langle \mathbf{u} \rangle^2 = \langle (\mathbf{u} - \langle \mathbf{u} \rangle)\mathbf{u} \rangle$$

Relation between PCA and learning rules

Making possible both LTP and LTD

Postsynaptic threshold $\tau_w \frac{d\mathbf{w}}{dt} = (v - \theta_v)\mathbf{u}$

homosynaptic depression

Postsynaptic threshold $\tau_w \frac{d\mathbf{w}}{dt} = v(\mathbf{u} - \boldsymbol{\theta}_u)$

heterosynaptic depression

Setting the threshold to average postsynaptic activity: $\theta_v = \langle v \rangle$

$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{C} \cdot \mathbf{w}$, where

$$\mathbf{C} = \langle (\mathbf{u} - \langle \mathbf{u} \rangle)(\mathbf{u} - \langle \mathbf{u} \rangle) \rangle = \langle \mathbf{uu} \rangle - \langle \mathbf{u} \rangle^2 = \langle (\mathbf{u} - \langle \mathbf{u} \rangle)\mathbf{u} \rangle$$

BCM rule: $\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u}(v - \theta_v)$

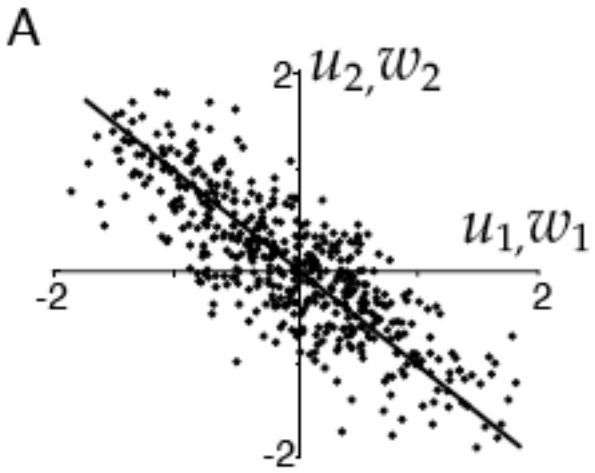
Relation between PCA and learning rules
Regularization again

BCM rule: $\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u}(v - \theta_v) \Longrightarrow \tau_\theta \frac{d\theta_v}{dt} = v^2 - \theta_v$

Hebb rule: $\tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u} \Longrightarrow \tau_w \frac{d\mathbf{w}}{dt} = v\mathbf{u} - \alpha v^2 \mathbf{w}$

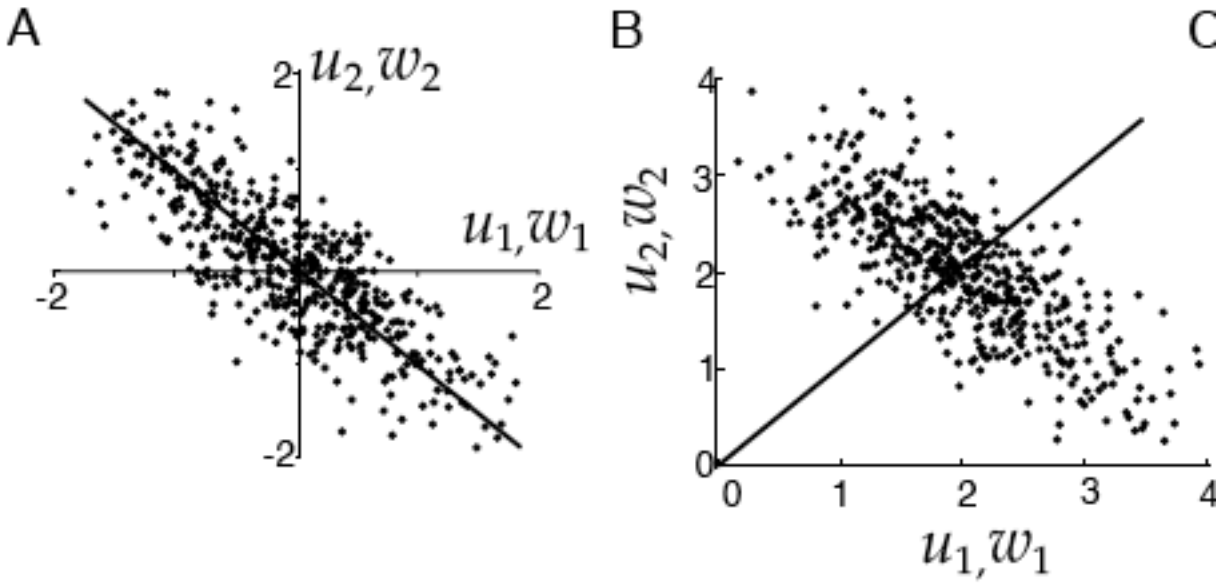
(Oja rule)

Relation between PCA and learning rules



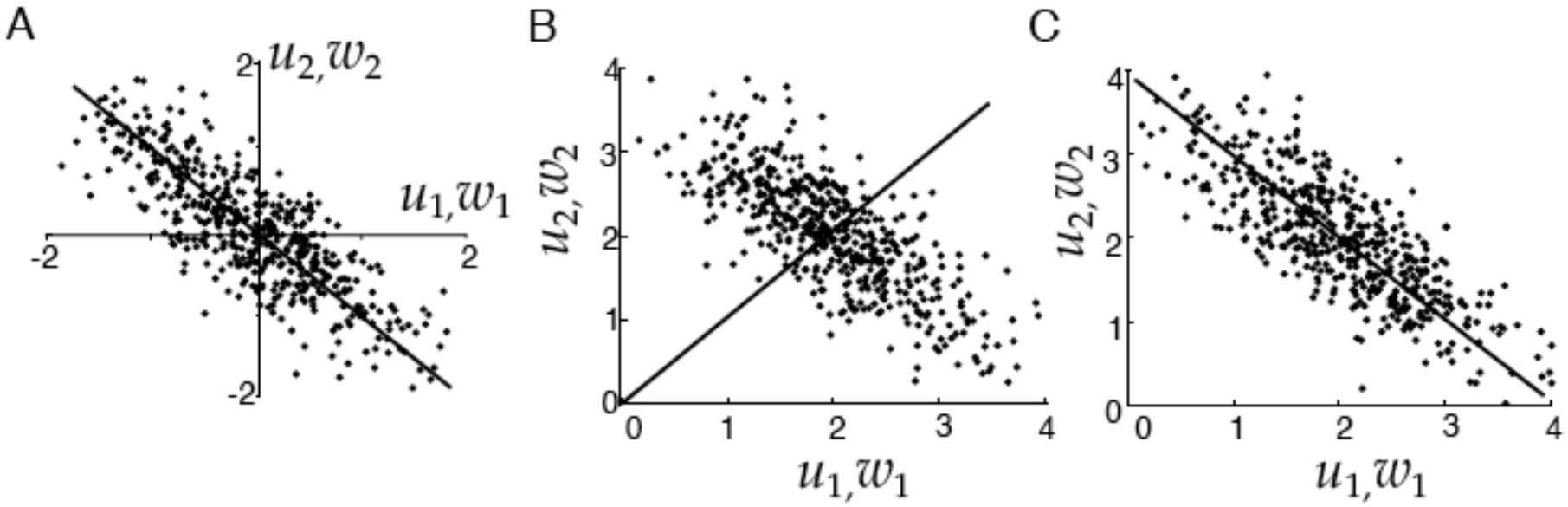
$$\mathbf{w}(t) = \sum_{\mu=1}^{N_u} \exp\left(\frac{\lambda_\mu t}{\tau_w}\right) (\mathbf{w}(0) \cdot \mathbf{e}_\mu)\, \mathbf{e}_\mu$$

Relation between PCA and learning rules



$$\mathbf{w}(t) = \sum_{\mu=1}^{N_u} \exp\left(\frac{\lambda_\mu t}{\tau_w}\right) (\mathbf{w}(0) \cdot \mathbf{e}_\mu)\, \mathbf{e}_\mu$$

Relation between PCA and learning rules



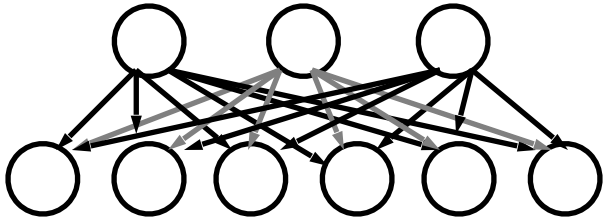
$$\mathbf{w}(t) = \sum_{\mu=1}^{N_u} \exp\left(\frac{\lambda_\mu t}{\tau_w}\right)(\mathbf{w}(0)\cdot\mathbf{e}_\mu)\,\mathbf{e}_\mu$$

The architecture of the network and **learning rule** hand-in-hand detemine the learned representation

Density estimation

Empirical distribution/ input distribution $p[\mathbf{u}]$

Latent variables: v



Recognition: $p[\, v \mid \mathbf{u}]$

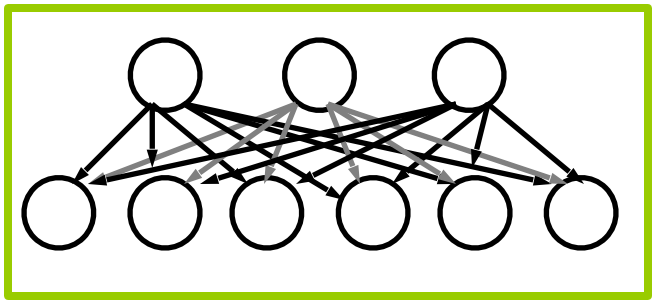
Generative distribution: $p[\mathbf{u}|v;\, \mathcal{G}\,]$

Kullback-Leibler divergence: $D_{\mathrm{KL}}(p[\mathbf{u}],\, p[\mathbf{u};\, \mathcal{G}\,]) = \int d\mathbf{u}\, p[\mathbf{u}] \ln \frac{p[\mathbf{u}]}{p[\mathbf{u};\, \mathcal{G}\,]}$

Density estimation

Empirical distribution/ input distribution $p[\mathbf{u}]$

Latent variables: v



Recognition: $p[\, v \mid \mathbf{u}]$

generative model

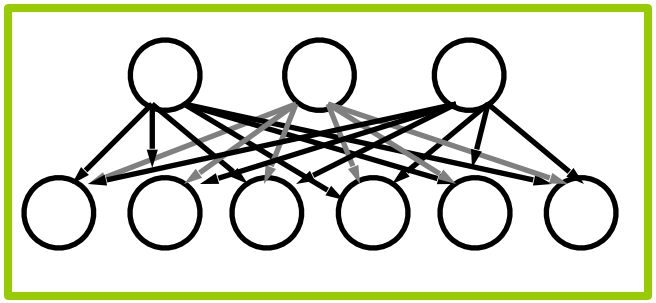
Generative distribution: $p[\mathbf{u}|v;\, \mathcal{G}\,]$

Kullback-Leibler divergence: $D_{\mathrm{KL}}(p[\mathbf{u}],\, p[\mathbf{u};\, \mathcal{G}\,]) = \int d\mathbf{u}\, p[\mathbf{u}] \ln \frac{p[\mathbf{u}]}{p[\mathbf{u};\, \mathcal{G}\,]}$

Density estimation

Empirical distribution/ input distribution $p[\mathbf{u}]$

Latent variables: v



Recognition: $p[\, v \mid \mathbf{u}]$

generative model

Generative distribution: $p[\mathbf{u}|v;\, \mathcal{G}\,]$

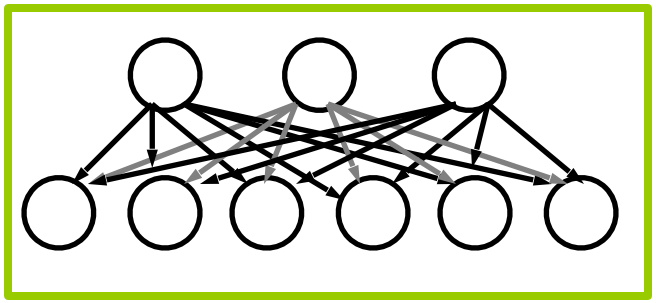
Recognition distribution: $P[v|\mathbf{u};\, \mathcal{G}\,] = \dfrac{p[\mathbf{u}|v;\, \mathcal{G}\,]P[v;\, \mathcal{G}\,]}{p[\mathbf{u};\, \mathcal{G}\,]}$

Kullback-Leibler divergence: $D_{\mathrm{KL}}(p[\mathbf{u}],\, p[\mathbf{u};\, \mathcal{G}\,]) = \int d\mathbf{u}\, p[\mathbf{u}] \ln \dfrac{p[\mathbf{u}]}{p[\mathbf{u};\, \mathcal{G}\,]}$

Density estimation

Empirical distribution/ input distribution $p[\mathbf{u}]$

Latent variables: v



Recognition: $p[\, v \mid \mathbf{u}]$

generative model

Generative distribution: $p[\mathbf{u}|v;\, \mathcal{G}\,]$

Recognition distribution: $P[v|\mathbf{u};\, \mathcal{G}\,] = \dfrac{p[\mathbf{u}|v;\, \mathcal{G}\,]P[v;\, \mathcal{G}\,]}{p[\mathbf{u};\, \mathcal{G}\,]}$

Kullback-Leibler divergence: $D_{\mathrm{KL}}(p[\mathbf{u}],\, p[\mathbf{u};\, \mathcal{G}\,]) = \int d\mathbf{u}\, p[\mathbf{u}] \ln \dfrac{p[\mathbf{u}]}{p[\mathbf{u};\, \mathcal{G}\,]}$

The match between our model distribution and input distributio

How to solve density estmation?
EM

$$\mathcal{F}(Q, \mathcal{G}) = \left\langle \sum_v Q[v; \mathbf{u}] \ln \frac{p[v, \mathbf{u}; \mathcal{G}]}{Q[v; \mathbf{u}]} \right\rangle$$

$$\sum_v Q[v; \mathbf{u}] = 1$$

$$\begin{aligned}
\mathcal{F}(Q, \mathcal{G}) &= \left\langle \sum_v Q[v; \mathbf{u}] \left(\ln p[\mathbf{u}; \mathcal{G}] + \ln \frac{P[v|\mathbf{u}; \mathcal{G}]}{Q[v; \mathbf{u}]} \right) \right\rangle \\
&= \langle \ln p[\mathbf{u}; \mathcal{G}] \rangle - \left\langle \sum_v Q[v; \mathbf{u}] \left(\ln \frac{Q[v; \mathbf{u}]}{P[v|\mathbf{u}; \mathcal{G}]} \right) \right\rangle \\
&= L(\mathcal{G}) - \langle D_{\mathrm{KL}}(Q[v; \mathbf{u}], P[v|\mathbf{u}; \mathcal{G}]) \rangle .
\end{aligned}$$

$$L(\mathcal{G}) \geq \mathcal{F}(Q, \mathcal{G})$$

Sparse coding

PCA:

Sparse coding: make the reconstruction faithful
keep the units/neurons quiet

$$E = -[\text{preserve information}] - \lambda[\text{sparseness of } a_i]$$

Sparse coding

PCA: trying to find linearly independent filters
setting the basis along the eigenvectors of the data

Sparse coding: make the reconstruction faithful
keep the units/neurons quiet

$$E = -[\text{preserve information}] - \lambda[\text{sparseness of } a_i]$$

Sparse coding

PCA: trying to find linearly independent filters
setting the basis along the eigenvectors of the data

Sparse coding: make the reconstruction faithful
keep the units/neurons quiet

$$E = -[\text{preserve information}] - \lambda[\text{sparseness of } a_i]$$

$$[\text{preserve information}] = -\sum_{x,y}\left[I(x,y) - \sum_i a_i \phi_i(x,y)\right]^2$$

$$[\text{sparseness of } a_i] = -\sum_i S\left(\frac{a_i}{\sigma}\right)$$

Sparse coding

Neural dynamics: gradient descent

$$\dot{a}_i = b_i - \sum_j C_{ij} a_j - \frac{\lambda}{\sigma} S'\left(\frac{a_i}{\sigma}\right)$$

$$\Delta\phi_i(x_m, y_n) = \eta\left\langle a_i \left[I(x_m, y_n) - \hat{I}(x_m, y_n) \right] \right\rangle$$

$$\hat{I}(x_m, y_n) = \Sigma_i a_i \phi_i(x_m, y_n),$$

Sparse coding