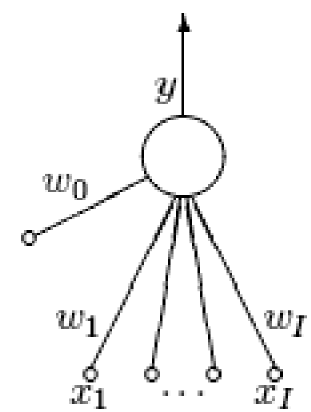
Neurons and neural networks II.
Hopfield network

Perceptron recap

- key ingredient: adaptivity of the system
- unsupervised vs supervised learning
- architecture for discrimination: single neuron — perceptron
- error function & learning rule
- gradient descent learning & divergence
- regularisation
- learning as inference



Interpreting learning as inference

So far: optimization wrt. an objective function

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

where

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

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

Interpreting learning as inference

So far: optimization wrt. an objective function

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

where

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

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

What’s this quirky regularizer, anyway?

Interpreting learning as inference

Let's interpret $y(\mathbf{x},\mathbf{w})$ as a probability:

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

Interpreting learning as inference

Let's interpret $y(\mathbf{x},\mathbf{w})$ as a probability:

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

in a compact form:

$$P(t\,|\,\mathbf{w},\mathbf{x}) = y^t(1-y)^{1-t} = \exp[t\ln y + (1-t)\ln(1-y)]$$

Interpreting learning as inference

Let's interpret *y*(**x,w**) as a probability:

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

in a compact form:

$$P(t\,|\,\mathbf{w},\mathbf{x}) = y^t(1-y)^{1-t} = \exp[t\ln y + (1-t)\ln(1-y)]$$

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

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

Interpreting learning as inference

Let's interpret $y(\mathbf{x},\mathbf{w})$ as a probability:

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

in a compact form:

$$P(t\,|\,\mathbf{w},\mathbf{x}) = y^t(1-y)^{1-t} = \exp[t\ln y + (1-t)\ln(1-y)]$$

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

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

$\Longrightarrow$ the regularizer has the form of a prior!

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

Interpreting learning as inference

Let's interpret *y*(**x,w**) as a probability:

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

in a compact form:

$$P(t\,|\,\mathbf{w},\mathbf{x}) = y^t(1-y)^{1-t} = \exp[t\ln y + (1-t)\ln(1-y)]$$

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

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

$\Longrightarrow$ 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**: $P(\mathbf{w}\,|\,D,\alpha) = \dfrac{P(D\,|\,\mathbf{w})P(\mathbf{w}\,|\,\alpha)}{P(D\,|\,\alpha)}$

Interpreting learning as inference Relationship between $M(\mathbf{w})$ and the posterior

$$
\begin{aligned}
P(\mathbf{w} \mid D, \alpha) &= \frac{P(D \mid \mathbf{w}) P(\mathbf{w} \mid \alpha)}{P(D \mid \alpha)} \\
&= \frac{e^{-G(\mathbf{w})} \, e^{-\alpha E_W(\mathbf{w})} / Z_W(\alpha)}{P(D \mid \alpha)} \\
&= \frac{1}{Z_M} \exp(-M(\mathbf{w})).
\end{aligned}
$$

interpretation: minimizing $M(\mathbf{w})$ leads to finding the *maximum a posteriori* estimate $\mathbf{w}_{\mathbf{MP}}$

The log probability interpretation of the objective function retains:

additivity of errors, while

keeping the multiplicativity of probabilities

Interpreting learning as inference
Properties of the Bayesian estimate

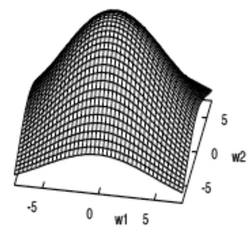
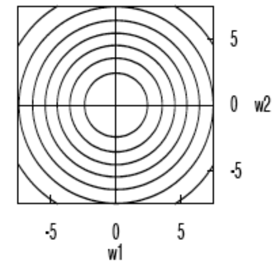
Interpreting learning as inference
Properties of the Bayesian estimate

- The probabilistic interpretation makes our assumptions explicit:

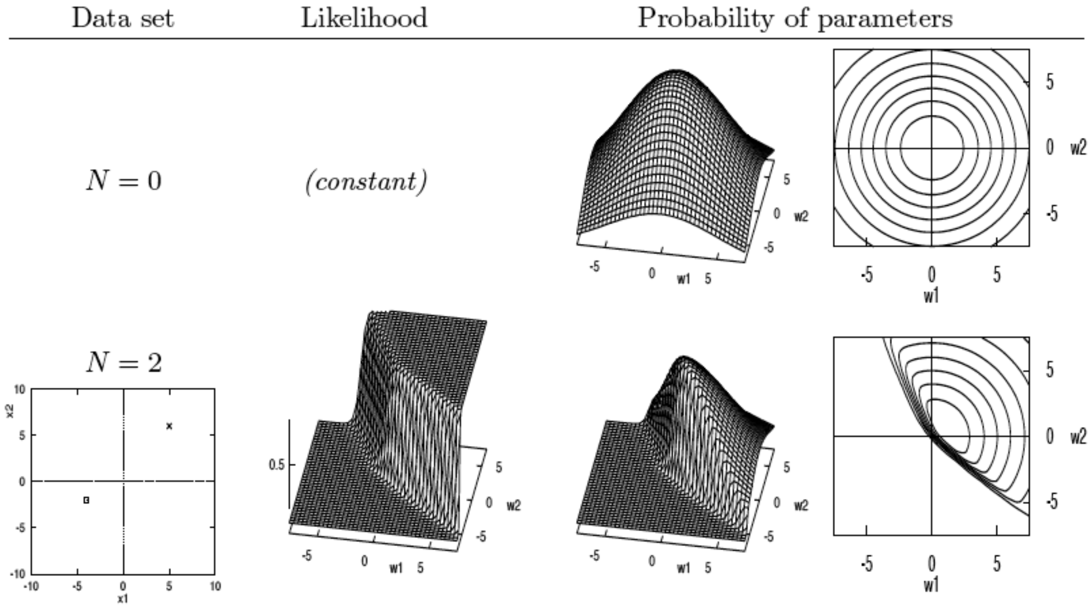
  by the regularizer we imposed a soft constraint on the learned parameters, which expresses our *prior expecations*.
- An additional plus:

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

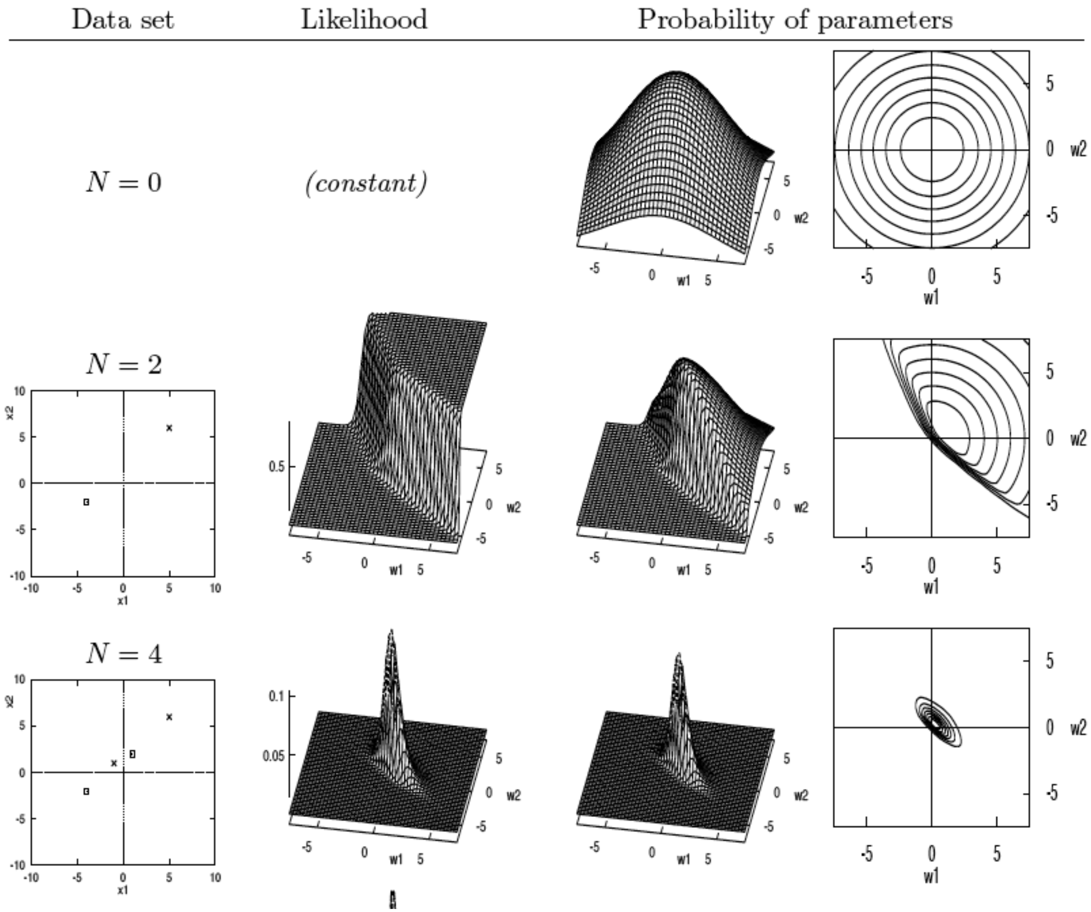
Interpreting learning as inference Demo

Data set	Likelihood	Probability of parameters	
$N = 0$	*(constant)*		

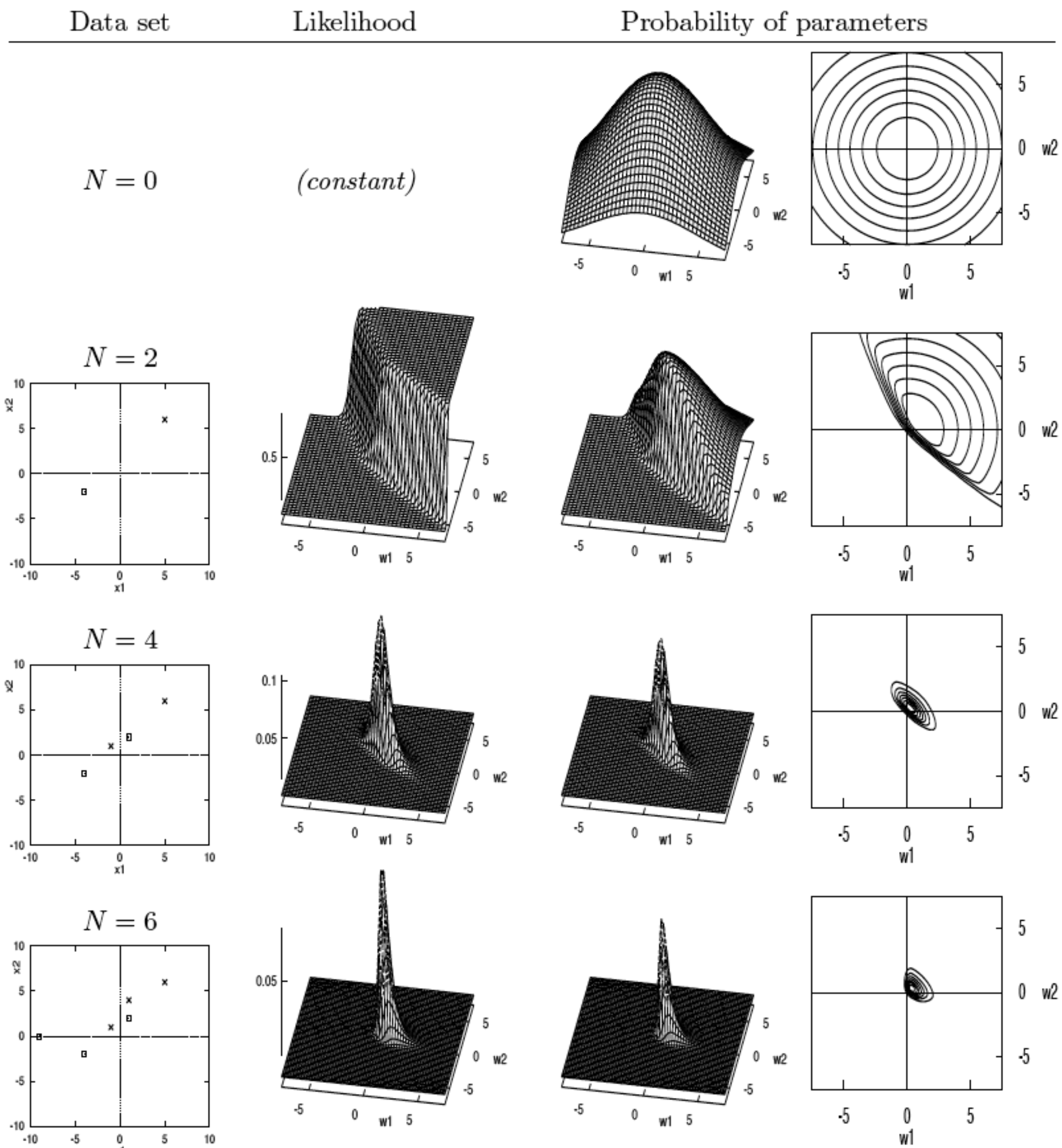
Interpreting learning as inference Demo



Interpreting learning as inference Demo

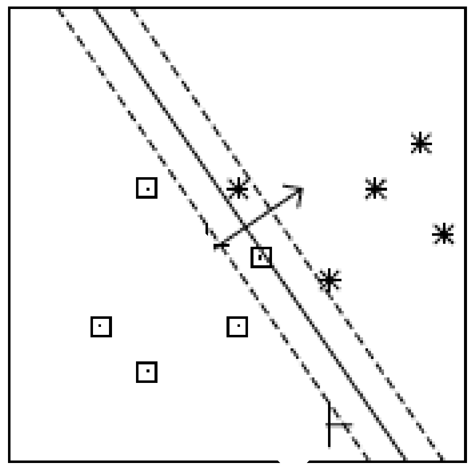


Interpreting learning as inference Demo



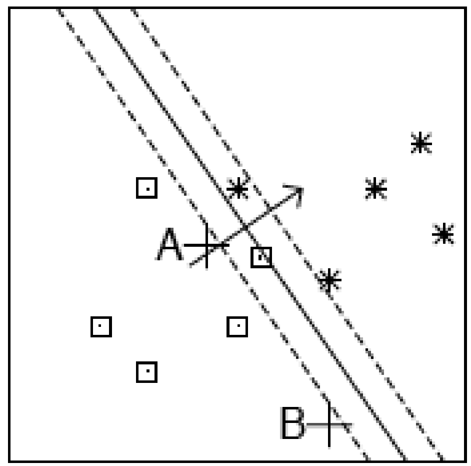
Interpreting learning as inference
Making predictions

Up to this point the goal was optimization: $M(\mathbf{w}) = G(\mathbf{w}) + \alpha E(\mathbf{w})$



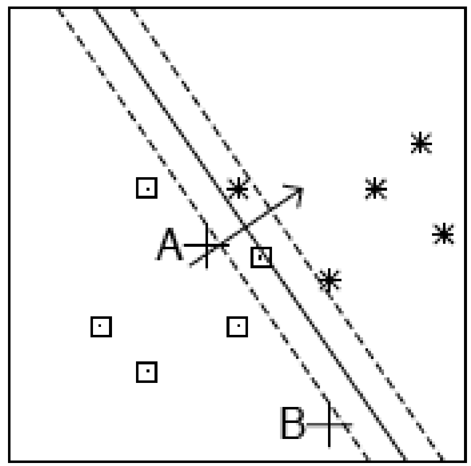
Interpreting learning as inference
Making predictions

Up to this point the goal was optimization: $M(\mathbf{w}) = G(\mathbf{w}) + \alpha E(\mathbf{w})$



Interpreting learning as inference
Making predictions

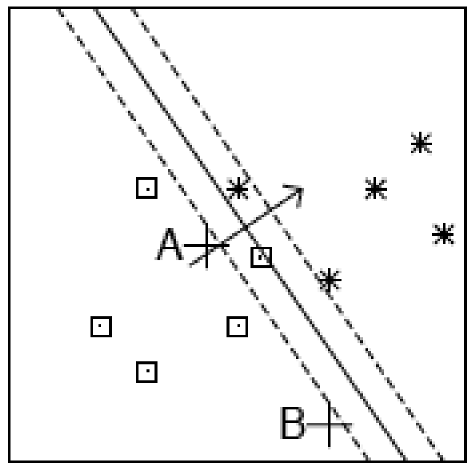
Up to this point the goal was optimization: $M(\mathbf{w}) = G(\mathbf{w}) + \alpha E(\mathbf{w})$



Are we equally confident in the two predictions?

Interpreting learning as inference
Making predictions

Up to this point the goal was optimization: $M(\mathbf{w}) = G(\mathbf{w}) + \alpha E(\mathbf{w})$



Are we equally confident in the two predictions?

The Bayesian answer exploits the probabilistic interpretation:

$$P(\mathrm{t}^{(N+1)} \mid \mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K \mathbf{w}\, P(\mathrm{t}^{(N+1)} \mid \mathbf{x}^{(N+1)}, \mathbf{w}, \alpha) P(\mathbf{w} \mid D, \alpha)$$

Interpreting learning as inference
Calculating Bayesian predictions

Predictive probability:

$$P(\mathrm{t}^{(N+1)} \mid \mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K \mathbf{w} \, P(\mathrm{t}^{(N+1)} \mid \mathbf{x}^{(N+1)}, \mathbf{w}, \alpha) P(\mathbf{w} \mid D, \alpha)$$

Interpreting learning as inference
Calculating Bayesian predictions

Predictive probability:

$$P(\mathrm{t}^{(N+1)} \mid \mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K \mathbf{w}\, P(\mathrm{t}^{(N+1)} \mid \mathbf{x}^{(N+1)}, \mathbf{w}, \alpha) P(\mathbf{w} \mid D, \alpha)$$

Likelihood:

$$\begin{aligned} P(\mathrm{t}^{(N+1)} = 1 \mid \mathbf{x}^{(N+1)}, \mathbf{w}, \alpha) &= y(\mathbf{x}^{(N+1)}; \mathbf{w}) \\ P(\mathrm{t}^{(N+1)} = 0 \mid \mathbf{x}^{(N+1)}, \mathbf{w}, \alpha) &= 1 - y(\mathbf{x}^{(N+1)}; \mathbf{w}) \end{aligned}$$

Weight posterior

$$P(\mathbf{w} \mid D, \alpha) = \frac{1}{Z_M} \exp(-M(\mathbf{w}))$$

Partition function:

$$Z_M = \int \mathrm{d}^K \mathbf{w} \exp(-M(\mathbf{w}))$$

Interpreting learning as inference
Calculating Bayesian predictions

Predictive probability:

$$P(\mathrm{t}^{(N+1)} \mid \mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K\mathbf{w}\, P(\mathrm{t}^{(N+1)} \mid \mathbf{x}^{(N+1)}, \mathbf{w}, \alpha) P(\mathbf{w} \mid D, \alpha)$$

Likelihood:

$$\begin{aligned} P(\mathrm{t}^{(N+1)} = 1 \mid \mathbf{x}^{(N+1)}, \mathbf{w}, \alpha) &= y(\mathbf{x}^{(N+1)}; \mathbf{w}) \\ P(\mathrm{t}^{(N+1)} = 0 \mid \mathbf{x}^{(N+1)}, \mathbf{w}, \alpha) &= 1 - y(\mathbf{x}^{(N+1)}; \mathbf{w}) \end{aligned}$$

Weight posterior

$$P(\mathbf{w} \mid D, \alpha) = \frac{1}{Z_M} \exp(-M(\mathbf{w}))$$

Partition function:

$$Z_M = \int \mathrm{d}^K\mathbf{w} \exp(-M(\mathbf{w}))$$

Finally: $$P(\mathrm{t}^{(N+1)} = 1 \mid \mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K\mathbf{w}\;\; y(\mathbf{x}^{(N+1)}; \mathbf{w})\, \frac{1}{Z_M} \exp(-M(\mathbf{w}))$$

Interpreting learning as inference
Calculating Bayesian predictions

$$P(\mathrm{t}^{(N+1)} = 1 \mid \mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K \mathbf{w} \;\; y(\mathbf{x}^{(N+1)}; \mathbf{w}) \, \frac{1}{Z_M} \exp(-M(\mathbf{w}))$$

How to solve the integral?

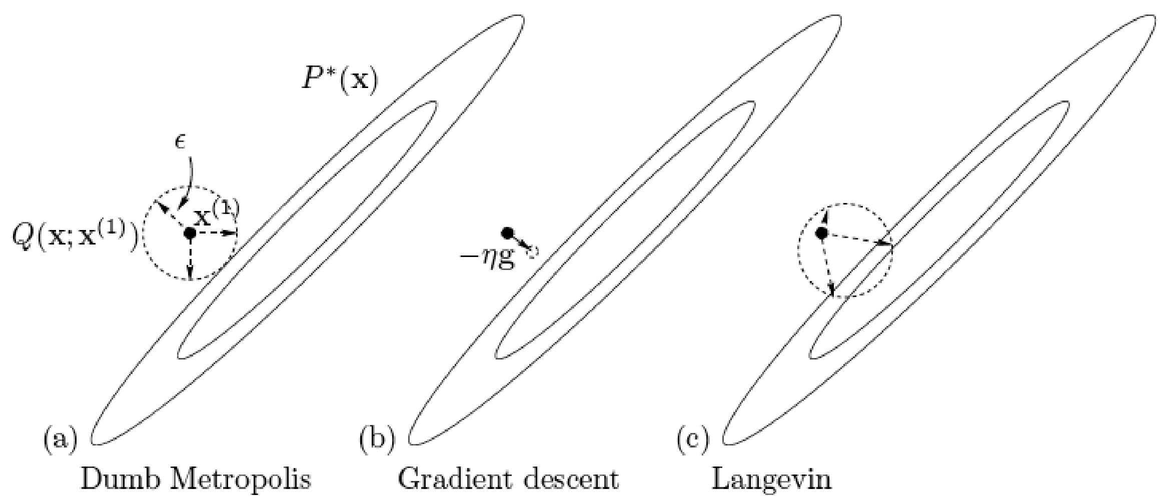
Interpreting learning as inference
Calculating Bayesian predictions

$$P(\mathrm{t}^{(N+1)} = 1 \mid \mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K \mathbf{w} \;\; y(\mathbf{x}^{(N+1)}; \mathbf{w}) \, \frac{1}{Z_M} \exp(-M(\mathbf{w}))$$

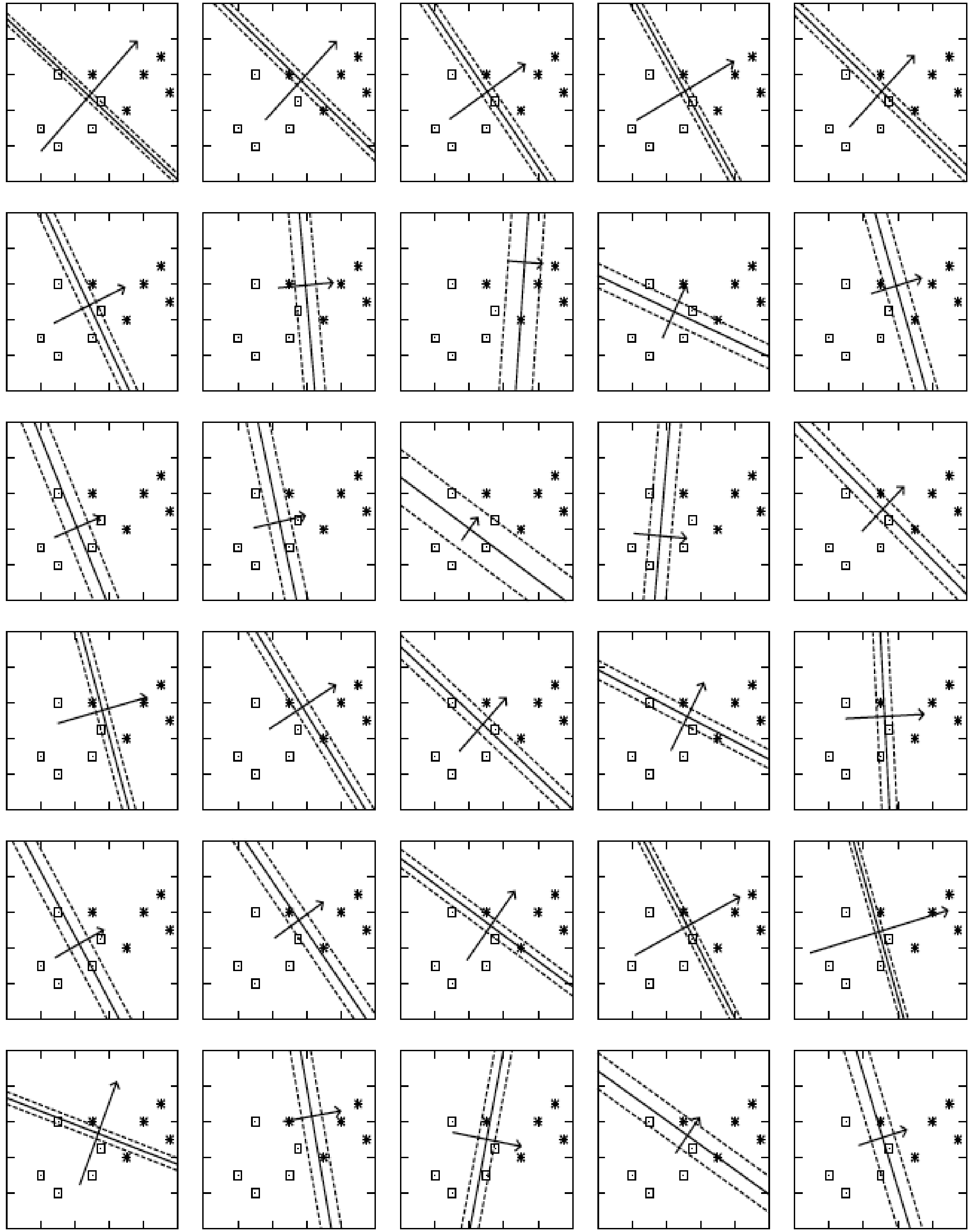
How to solve the integral?

Bad news: Monte Carlo integration is needed

$$\langle f(\mathbf{w}) \rangle \simeq \frac{1}{R} \sum_r f(\mathbf{w}^{(r)})$$



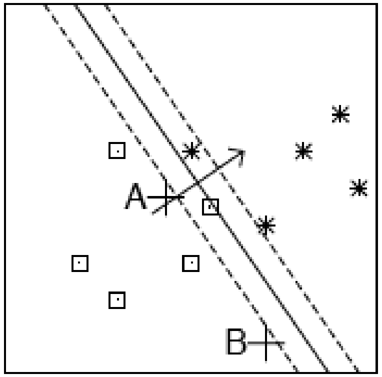
```
g = gradM ( w ) ;                            # set gradient using initial w
M = findM ( w ) ;                            # set objective function too

for l = 1:L                                  # loop L times
   p = randn ( size(w) ) ;                   # initial momentum is Normal(0,1)
   H = p' * p / 2 + M ;                      # evaluate H(w,p)

*  p = p - epsilon * g / 2 ;                 # make half-step in p
*  wnew = w + epsilon * p ;                  # make step in w
*  gnew = gradM ( wnew ) ;                   # find new gradient
*  p = p - epsilon * gnew / 2 ;              # make half-step in p

   Mnew = findM ( wnew ) ;                   # find new objective function
   Hnew = p' * p / 2 + Mnew ;                # evaluate new value of H
   dH = Hnew - H ;                           # decide whether to accept
   if ( dH < 0 )                    accept = 1 ;
   elseif ( rand() < exp(-dH) ) accept = 1 ;  # compare with a uniform
   else                             accept = 0 ;  #                variate
   endif
   if ( accept )      g = gnew ;   w = wnew ;    M = Mnew ;    endif
endfor

function gM = gradM ( w )                    # gradient of objective function
   a = x * w  ;                              # compute activations
   y = sigmoid(a) ;                          # compute outputs
   e = t - y   ;                             # compute errors
   g = - x' * e ;                            # compute the gradient of G(w)
   gM = alpha * w + g ;
endfunction

function M = findM ( w )                     # objective function
   G = - (t' * log(y) + (1-t') * log( 1-y )) ;
   EW = w' * w / 2 ;
   M = G + alpha * EW ;
endfunction
```



Interpreting learning as inference
Calculating Bayesian predictions

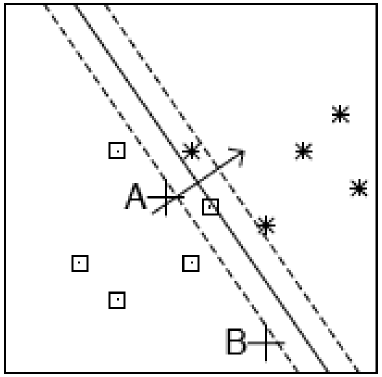
Interpreting learning as inference
Calculating Bayesian predictions

Original estimate

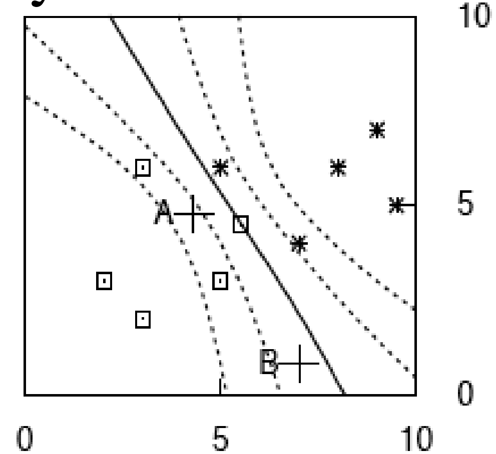


Interpreting learning as inference
Calculating Bayesian predictions

Original estimate



Bayesian estimate



Interpreting learning as inference Gaussian approximation

$$P(\mathrm{t}^{(N+1)} = 1 \mid \mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K\mathbf{w}\, y(\mathbf{x}^{(N+1)}; \mathbf{w}) \frac{1}{Z_M} \exp(-M(\mathbf{w})),$$

Taylor expansion around the MAP estimate

$$M(\mathbf{w}) \simeq M(\mathbf{w}_{\mathrm{MP}}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_{\mathrm{MP}})^{\mathsf{T}}\mathbf{A}(\mathbf{w} - \mathbf{w}_{\mathrm{MP}}) + \cdots,$$

$$A_{ij} \equiv \left.\frac{\partial^2}{\partial w_i \partial w_j} M(\mathbf{w})\right|_{\mathbf{w}=\mathbf{w}_{\mathrm{MP}}}$$

Interpreting learning as inference
Gaussian approximation

$$P(\mathrm{t}^{(N+1)}=1 \mid \mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K \mathbf{w}\, y(\mathbf{x}^{(N+1)}; \mathbf{w}) \frac{1}{Z_M} \exp(-M(\mathbf{w})),$$

Taylor expansion around the MAP estimate

$$M(\mathbf{w}) \simeq M(\mathbf{w}_{\mathrm{MP}}) + \frac{1}{2}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})^{\mathsf{T}}\mathbf{A}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}}) + \cdots,$$

$$A_{ij} \equiv \frac{\partial^2}{\partial w_i \partial w_j} M(\mathbf{w})\bigg|_{\mathbf{w}=\mathbf{w}_{\mathrm{MP}}}$$

The Gaussian approximation:

$$Q(\mathbf{w}; \mathbf{w}_{\mathrm{MP}}, \mathbf{A}) = [\det(\mathbf{A}/2\pi)]^{1/2} \exp\left[-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})^{\mathsf{T}}\mathbf{A}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})\right]$$

Interpreting learning as inference
Gaussian approximation

$$P(\mathrm{t}^{(N+1)}=1\,|\,\mathbf{x}^{(N+1)}, D, \alpha) = \int \mathrm{d}^K\mathbf{w}\; y(\mathbf{x}^{(N+1)};\mathbf{w})\,\frac{1}{Z_M}\exp(-M(\mathbf{w})),$$

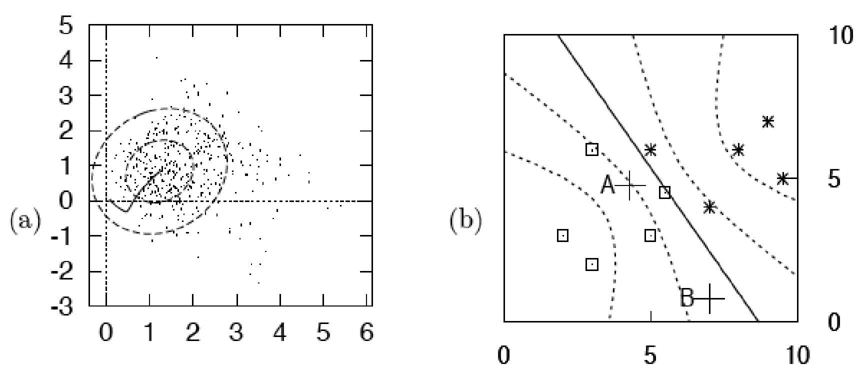
Taylor expansion around the MAP estimate

$$M(\mathbf{w}) \simeq M(\mathbf{w}_{\mathrm{MP}}) + \frac{1}{2}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})^{\mathsf{T}}\mathbf{A}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}}) + \cdots,$$

$$A_{ij} \equiv \left.\frac{\partial^2}{\partial w_i \partial w_j} M(\mathbf{w})\right|_{\mathbf{w}=\mathbf{w}_{\mathrm{MP}}}$$

The Gaussian approximation:

$$Q(\mathbf{w};\mathbf{w}_{\mathrm{MP}},\mathbf{A}) = [\det(\mathbf{A}/2\pi)]^{1/2}\exp\left[-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})^{\mathsf{T}}\mathbf{A}(\mathbf{w}-\mathbf{w}_{\mathrm{MP}})\right]$$



Neural networks
Unsupervised learning

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!

Neural networks
The Hopfield network

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

;

Neural networks
The Hopfield network

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:

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

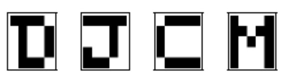
Neural networks
Stability of Hopfield network

Are the memories stable?

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

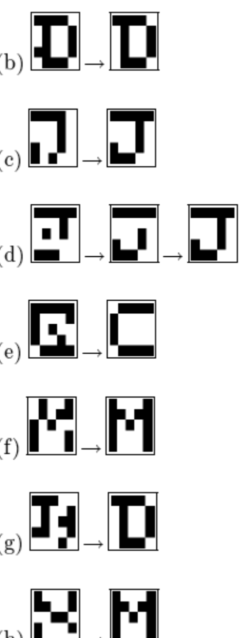
Necessary conditions: symmetric weights; asynchronous update

(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	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)

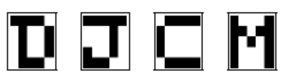


Neural networks
Stability of Hopfield network

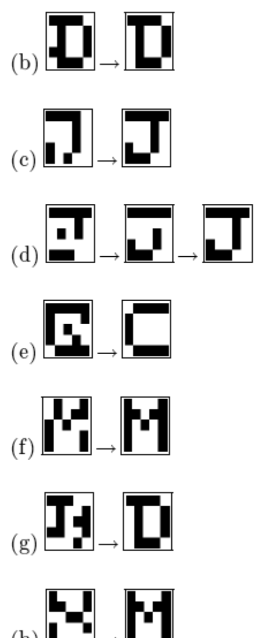
Are the memories stable?

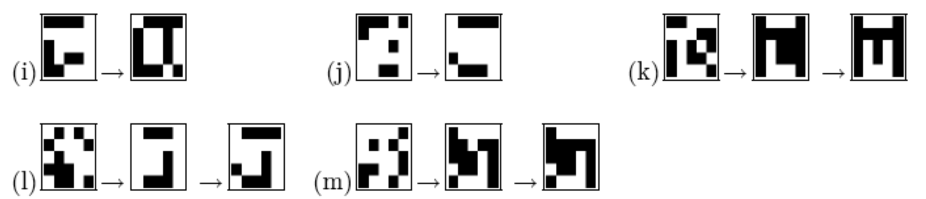
$$E(x, \mathbf{w}) = -\frac{1}{2}\sum_{m,n} w_{mn}x_m x_n - \sum_n w_{0n}x_n$$

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	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	



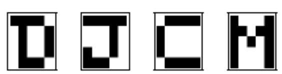


Neural networks
Stability of Hopfield network

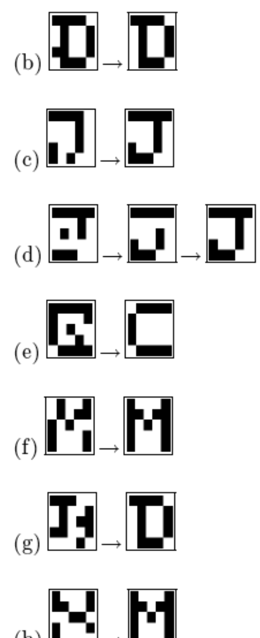
Are the memories stable?

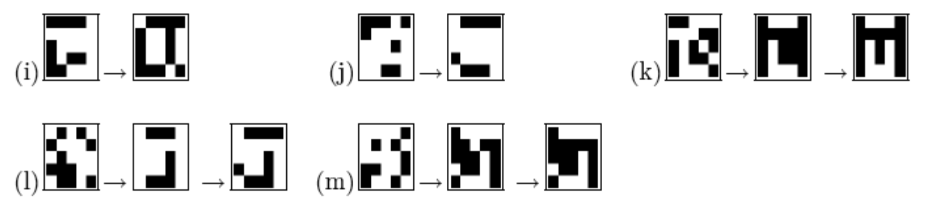
$$E(x, \mathbf{w}) = -\frac{1}{2}\sum_{m,n} w_{mn} x_m x_n - \sum_n w_{0n} x_n$$

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	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	

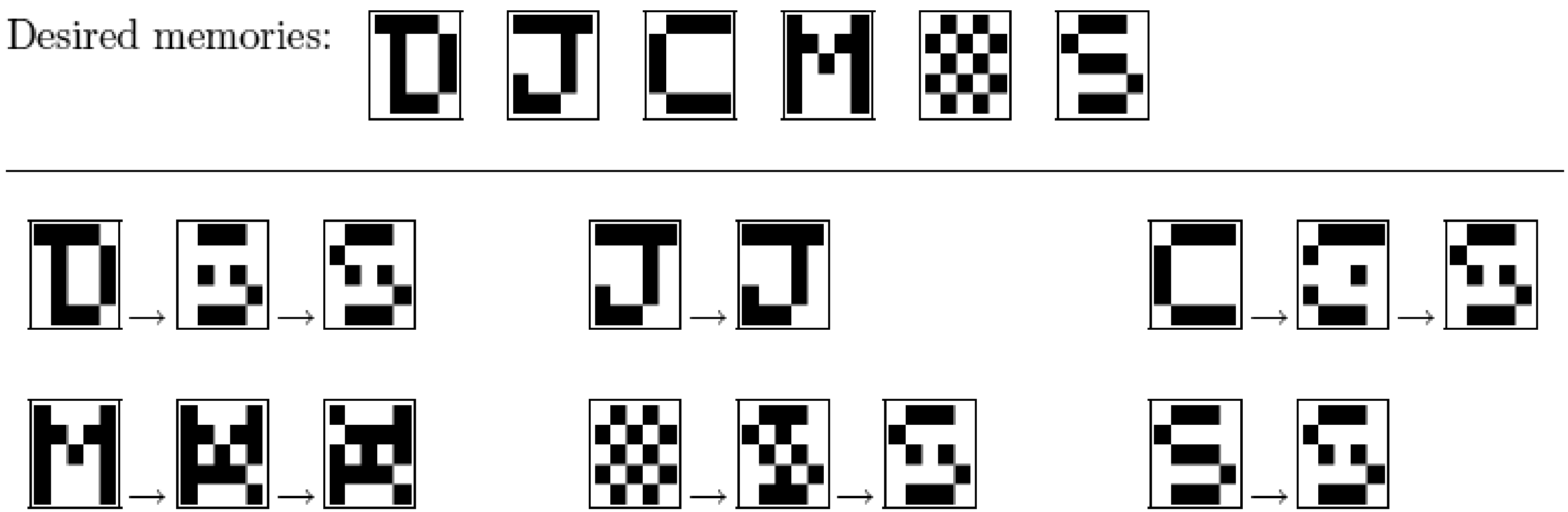




Robust against perturbation of a subset of weights

Neural networks
Capacity of Hopfield network

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



Neural networks
Capacity of Hopfield network

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

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

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

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