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

\[ M(w) = G(w) + \alpha E_W(w) \]

where

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

\[ E_W(w) = \frac{1}{2} \sum_i w_i^2. \]
Interpreting learning as inference

So far: optimization wrt. an objective function

\[ M(w) = G(w) + \alpha E_W(w) \]

where

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

\[ E_W(w) = \frac{1}{2} \sum_i w_i^2. \]

What’s this quirky regularizer, anyway?
Interpreting learning as inference

Let’s interpret $y(x,w)$ as a probability:

\[
P(t = 1 | w, x) = y \\
P(t = 0 | w, x) = 1 - y
\]
Interpreting learning as inference

Let’s interpret $y(x,w)$ as a probability:

$$
P(t = 1 | w, x) = y
$$

$$
P(t = 0 | w, x) = 1 - y
$$

in a compact form:

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

Let’s interpret \( y(x, w) \) as a probability:

\[
P(t = 1 \mid w, x) = y
\]
\[
P(t = 0 \mid w, x) = 1 - y
\]

in a compact form:

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

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

\[
P(D \mid w) = \exp[-G(w)]
\]
Interpreting learning as inference

Let’s interpret \( y(x,w) \) as a probability:

\[
P(t = 1 | w, x) = y \quad \quad P(t = 0 | w, x) = 1 - y
\]

in a compact form:

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

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

\[
P(D | w) = \exp[-G(w)]
\]

the regularizer has the form of a prior!

\[
P(w | \alpha) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W).
\]
Interpreting learning as inference

Let’s interpret \( y(x, w) \) as a probability:

\[
P(t = 1 \mid w, x) = y \\
P(t = 0 \mid w, x) = 1 - y
\]

in a compact form:

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

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

\[
P(D \mid w) = \exp[-G(w)]
\]

the regularizer has the form of a prior!

\[
P(w \mid \alpha) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W).
\]

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

the posterior distribution of \( w \):

\[
P(w \mid D, \alpha) = \frac{P(D \mid w)P(w \mid \alpha)}{P(D \mid \alpha)}
\]
Interpreting learning as inference
Relationship between $M(w)$ and the posterior

$$P(w \mid D, \alpha) = \frac{P(D \mid w)P(w \mid \alpha)}{P(D \mid \alpha)}$$

$$= \frac{e^{-G(w)}e^{-\alpha E_W(w)}/Z_W(\alpha)}{P(D \mid \alpha)}$$

$$= \frac{1}{Z_M} \exp(-M(w)).$$

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

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 expectations.

• An additional plus:
  beyond getting $w_{MP}$ we get a measure for learned parameter uncertainty
Interpreting learning as inference

Demo

<table>
<thead>
<tr>
<th>Data set</th>
<th>Likelihood</th>
<th>Probability of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 0$</td>
<td><em>(constant)</em></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

$N = 0$ *(constant)*
Interpreting learning as inference

Demo

<table>
<thead>
<tr>
<th>Data set</th>
<th>Likelihood</th>
<th>Probability of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 0$</td>
<td>(constant)</td>
<td><img src="image1.png" alt="Graph" /></td>
</tr>
<tr>
<td>$N = 2$</td>
<td><img src="image2.png" alt="Graph" /></td>
<td><img src="image3.png" alt="Graph" /></td>
</tr>
</tbody>
</table>
Interpreting learning as inference

Demo

<table>
<thead>
<tr>
<th>Data set</th>
<th>Likelihood</th>
<th>Probability of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 0$</td>
<td>(constant)</td>
<td></td>
</tr>
</tbody>
</table>

$N = 2$

$N = 4$
Interpreting learning as inference

Demo

<table>
<thead>
<tr>
<th>Data set</th>
<th>Likelihood</th>
<th>Probability of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N = 0$</td>
<td>(constant)</td>
<td></td>
</tr>
</tbody>
</table>

$N = 2$

$N = 4$

$N = 6$
Interpreting learning as inference
Making predictions

Up to this point the goal was optimization: \[ M(w) = G(w) + \alpha E(w) \]
Interpreting learning as inference
Making predictions

Up to this point the goal was optimization: \( M(w) = G(w) + \alpha E(w) \)
Interpreting learning as inference
Making predictions

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

Are we equally confident in the two predictions?
Interpreting learning as inference
Making predictions

Up to this point the goal was optimization: \[ M(w) = G(w) + \alpha E(w) \]

Are we equally confident in the two predictions?

The Bayesian answer exploits the probabilistic interpretation:

\[
P(t^{(N+1)} | x^{(N+1)}, D, \alpha) = \int d^K w \ P(t^{(N+1)} | x^{(N+1)}, w, \alpha) P(w | D, \alpha)
\]
Interpreting learning as inference
Calculating Bayesian predictions

Predictive probability:
\[ P(t^{(N+1)} | x^{(N+1)}, D, \alpha) = \int d^K w \ P(t^{(N+1)} | x^{(N+1)}, w, \alpha) P(w | D, \alpha) \]
Interpreting learning as inference
Calculating Bayesian predictions

Predictive probability:

\[ P(t^{(N+1)} | x^{(N+1)}, D, \alpha) = \int d^K w \ P(t^{(N+1)} | x^{(N+1)}, w, \alpha) P(w | D, \alpha) \]

Likelihood:

\[ P(t^{(N+1)} = 1 | x^{(N+1)}, w, \alpha) = y(x^{(N+1)}; w) \]
\[ P(t^{(N+1)} = 0 | x^{(N+1)}, w, \alpha) = 1 - y(x^{(N+1)}; w) \]

Weight posterior

\[ P(w | D, \alpha) = \frac{1}{Z_M} \exp(-M(w)) \]

Partition function:

\[ Z_M = \int d^K w \exp(-M(w)) \]
Interpreting learning as inference
Calculating Bayesian predictions

Predictive probability:

\[ P(t^{(N+1)} | x^{(N+1)}, D, \alpha) = \int d^K w \ P(t^{(N+1)} | x^{(N+1)}, w, \alpha)P(w | D, \alpha) \]

Likelihood:

\[ P(t^{(N+1)} = 1 | x^{(N+1)}, w, \alpha) = y(x^{(N+1)}; w) \]
\[ P(t^{(N+1)} = 0 | x^{(N+1)}, w, \alpha) = 1 - y(x^{(N+1)}; w) \]

Weight posterior

\[ P(w | D, \alpha) = \frac{1}{Z_M} \ \exp(-M(w)) \]

Partition function:

\[ Z_M = \int d^K w \ \exp(-M(w)) \]

Finally:

\[ P(t^{(N+1)} = 1 | x^{(N+1)}, D, \alpha) = \int d^K w \ y(x^{(N+1)}; w) \frac{1}{Z_M} \ \exp(-M(w)) \]
Interpreting learning as inference
Calculating Bayesian predictions

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

How to solve the integral?
Interpreting learning as inference
Calculating Bayesian predictions

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

How to solve the integral?
Bad news: Monte Carlo integration is needed

\[ \langle f(w) \rangle \approx \frac{1}{R} \sum_r f(w^{(r)}) \]
\[
g = \text{gradM}(w) \quad \text{# set gradient using initial } w
\]
\[
M = \text{findM}(w) \quad \text{# set objective function too}
\]

for \( l = 1:L \) \quad \text{# loop } L \text{ times}
\[
p = \text{randn(size(w))} \quad \text{# initial momentum is Normal(0,1)}
\]
\[
H = p' \cdot p / 2 + M \quad \text{# evaluate } H(w,p)
\]

* \( p = p - \epsilon \cdot g / 2 \) \quad \text{# make half-step in } p
* \( w_{new} = w + \epsilon \cdot p \) \quad \text{# make step in } w
* \( g_{new} = \text{gradM}(w_{new}) \) \quad \text{# find new gradient}
* \( p = p - \epsilon \cdot g_{new} / 2 \) \quad \text{# make half-step in } p

\[
M_{new} = \text{findM}(w_{new}) \quad \text{# find new objective function}
\]
\[
H_{new} = p' \cdot p / 2 + M_{new} \quad \text{# evaluate new value of } H
\]
\[
dH = H_{new} - H \quad \text{# decide whether to accept}
\]
if \( dH < 0 \)
\[
\quad \text{accept} = 1
\]
elseif \( \text{rand()} < \exp(-dH) \)
\[
\quad \text{accept} = 1 \quad \text{# compare with a uniform}
\]
else
\[
\quad \text{accept} = 0 \quad \text{# variate}
\]
endif
if \( \text{accept} \)
\[
g = g_{new} \quad w = w_{new} \quad M = M_{new}
\]
endif
endfor

function \( \text{gM} = \text{gradM}(w) \) \quad \text{# gradient of objective function}
\[
a = x \cdot w \quad \text{# compute activations}
\]
\[
y = \text{sigmoid}(a) \quad \text{# compute outputs}
\]
\[
e = t - y \quad \text{# compute errors}
\]
\[
g = -x' \cdot e \quad \text{# compute the gradient of } G(w)
\]
\[
gM = \alpha \cdot w + g
\]
endfunction

function \( M = \text{findM}(w) \) \quad \text{# objective function}
\[
G = -(t' \cdot \log(y) + (1-t') \cdot \log(1-y))
\]
\[
EW = w' \cdot w / 2
\]
\[
M = G + \alpha \cdot EW
\]
endfunction
Interpreting learning as inference
Calculating Bayesian predictions
Interpreting learning as inference
Calculating Bayesian predictions

Original estimate

[Diagram showing original estimate]
Interpreting learning as inference
Calculating Bayesian predictions

Original estimate

Bayesian estimate
Interpreting learning as inference

Gaussian approximation

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

Taylor expansion around the MAP estimate

\[ M(w) \approx M(w_{\text{MP}}) + \frac{1}{2}(w - w_{\text{MP}})^T A(w - w_{\text{MP}}) + \cdots, \]

\[ A_{ij} \equiv \frac{\partial^2}{\partial w_i \partial w_j} M(w) \bigg|_{w = w_{\text{MP}}} \]
Interpreting learning as inference

Gaussian approximation

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

Taylor expansion around the MAP estimate

\[ M(w) \approx M(w_{\text{MP}}) + \frac{1}{2}(w - w_{\text{MP}})^\top A(w - w_{\text{MP}}) + \cdots, \]

\[ A_{ij} \equiv \left. \frac{\partial^2}{\partial w_i \partial w_j} M(w) \right|_{w = w_{\text{MP}}} \]

The Gaussian approximation:

\[ Q(w; w_{\text{MP}}, A) = [\det(A/2\pi)]^{1/2} \exp \left[ -\frac{1}{2}(w - w_{\text{MP}})^\top A(w - w_{\text{MP}}) \right] \]
Interpreting learning as inference

Gaussian approximation

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

Taylor expansion around the MAP estimate

\[
M(w) \simeq M(w_{MP}) + \frac{1}{2}(w - w_{MP})^\top A(w - w_{MP}) + \cdots,
\]

\[
A_{ij} \equiv \left. \frac{\partial^2}{\partial w_i \partial w_j} M(w) \right|_{w = w_{MP}}
\]

The Gaussian approximation:

\[
Q(w; w_{MP}, A) = [\det(A/2\pi)]^{1/2} \exp \left[ -\frac{1}{2}(w - w_{MP})^\top A(w - w_{MP}) \right]
\]
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:

\[ \frac{dw_{ij}}{dt} \sim Correlation(x_i, x_j) \]

This simple rule realizes an associative memory!
Reasoning, deduction & the nervous system

Turing machine

- there are systems performing simple computations
- universality can be reached by combining these computations
- all universal system can perform the same computations
there are systems performing simple computations

universality can be reached by combining these computations

all universal system can perform the same computations

arbitrary programming language can be used to code all the programs
Reasoning, deduction &
the nervous system

Walter Pitts
1923 - 1969
Reasoning, deduction &
the nervous system
Reasoning, deduction & the nervous system
Reasoning, deduction &
the nervous system

Two states
• on
• off
Reasoning, deduction & the nervous system
Reasoning, deduction & the nervous system
Reasoning, deduction & the nervous system
Reasoning, deduction & the nervous system

\[ \sum \leq \]

large $w$

small $w$
Reasoning, deduction & the nervous system

logical calculus of the brain
Sweet child of ours

Perceptron
Frank Rosenblatt

“remarkable machine...[was] capable of what amounts to thought.”
Sweet child of ours

Perceptron
Frank Rosenblatt

“remarkable machine...[was] capable of what amounts to thought.”

*The New Yorker*, December 6, 1958 P. 44
Linear discrimination

- The perceptron can learn a linear subspace for discrimination.

(a) Illustration of object tangling. In a neuronal population space, each cardinal axis is one neuron's activity (e.g. firing rate over a 200 ms interval) and the dimensionality of the space is equal to the number of neurons. Although such high-dimensional spaces cannot be visualized, the three-dimensional views portrayed here provide fundamental insight.

(b) The manifolds of two objects (two faces, red and blue) are shown in a common neuronal population space. In this case, a decision (hyper-)plane can be drawn cleanly between them. If the world only consisted of this set of images, this neuronal representation would be 'good' for supporting visual recognition.

(c) In this case, the two object manifolds are intertwined, or tangled. A decision plane can no longer separate the manifolds, no matter how it is tipped or translated.

(d) Pixel (retina-like) manifolds generated from actual models of faces (14,400-dimensional data; 120 images) for two face objects were generated from mild variation in their pose, position, scale and lighting (for clarity, only the pose-induced portion of the manifold is displayed). The three-dimensional display axes were chosen to be the projections that best separate identity, pose azimuth and pose elevation. Even though this simple example only exercises a fraction of typical real-world variation, the object manifolds are hopelessly tangled. Although the manifolds appear to cross in this three-dimensional projection, they do not cross in the high-dimensional space in which they live.
Linear discrimination

- The perceptron can learn a linear subspace for discrimination
Linear discrimination

- The perceptron can learn a linear subspace for discrimination
Universal function approximation

- Multi-layer neural network can combine discrimination subspaces

- Multi-layer perceptron is a universal function approximator — albeit not necessarily effective
Deep networks as universal functional approximators

The figure illustrates the architecture of different types of neural networks.

- **Shallow feedforward** (1 hidden layer): This type of network has one hidden layer and can approximate functions with varying precision depending on the number of hidden units.

- **Deep feedforward** (>1 hidden layer): These networks have more than one hidden layer and can approximate complex functions.

- **Recurrent**: This network allows connections to form cycles, which is useful for processing temporal sequences of inputs.

Each network type consists of input, hidden, and output layers. The arrows represent the flow of information, with each layer's output being the input to the next layer.
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?

\[ E(x, w) = -\frac{1}{2} \sum_{m,n} w_{mn}x_mx_n - \sum_n w_{0n}x_n \]

Necessary conditions: symmetric weights; asynchronous update
Neural networks

Stability of Hopfield network

Are the memories stable?

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

Necessary conditions: symmetric weights; asynchronous update
Neural networks

Stability of Hopfield network

Are the memories stable?

\[ E(x, w) = -\frac{1}{2} \sum_{m,n} w_{mn} x_m x_n \]

Necessary conditions: symmetric weights; asynchronous update

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?

Desired memories: $D J C M O S S$
Neural networks
Capacity of Hopfield network

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

\[
\omega_{ij} = x_i^{(n)} x_j^{(n)} + \sum_{m \neq n} x_i^{(m)} x_j^{(m)}.
\]

\[
a_i = \sum_{j \neq i} x_i^{(n)} x_j^{(n)} x_j^{(n)} + \sum_{j \neq i} \sum_{m \neq n} x_i^{(m)} x_j^{(m)} x_j^{(n)}
\]

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

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

goo.gl/R0OT6n