Computational Neuroscience

Structure – Dynamics – Implementation – Algorithm – Computation - Function
Learning at psychological level

- Classical conditioning
Hebb's rule

When an axon of cell A is near enough to excite cell B, and repeatedly or consistently takes part in firing it, some growth process or metabolic change takes part in one or both cells such that A's efficiency, as one of cell firing B, is increased" (Hebb, The Organization of Behavior, 1949)
The mathematical formalization

- Tuning of the model parameters based on data
- Two level dynamics
  - Variables (input-output transformation) - fast
  - Parameters - slow
- Memory vs learning
  - Memory is a simple recall, without change of the representation
  - Learning, continuous refinement of the representation accompanied by output generation
- Main task: prediction of the future, based on the past
Three basic types of learning

- **Supervised**
  - data: input output pairs
  - Aim, function approximation, classification

- **Reinforcement**
  - data: observed states, rewards
  - Aim: optimal strategy stratégia for maximization of reward

- **Unsupervised, representational**
  - Data: set of input
  - Aim: optimal representation / model finding

- Combination of them
Learning in neural systems

• Single neuron
• Feedforward network
• Recurrent network
• today: rate model
  • Parameters: weights, thresholds
  • Transfer functions
    – Step function: H (Heavyside)
    – Sigmoid
    – Linear neuron

\[ y = f(xw - \theta) \]
Perceptron

- Binary neurons: linear separation
  - In two dimension, the separator line:

\[
\theta = x_1 w_1 + x_2 w_2 \quad \rightarrow \quad x_2 = \frac{-w_1}{w_2} x_1 + \frac{\theta}{w_2}
\]

- Logical functions
Error-correcting learning rules

Rosenblatt-algorithm – a binary neuron

\[ y = H(w_1 + w_2 - \Theta) \]

The threshold is transformed out by introducing a constant 1 input \( b = -\Theta \) (bias)

Learning the **bias** is equivalent to the learning of a weight.
Error-correcting learning rules

- Using the actual and the correct answer, and the distance between them
- Rosenblatt-algorithm – binary neurons

\[ w(t+1) = w(t) + \epsilon (t^m - y(x^m)) x^m \]

- Delta-rule
  - Continuous activity – gradient-method
    \[ w_b(t+1) = w_b(t) - \epsilon \frac{\partial E}{\partial w_b} \]
    \[ E = \frac{1}{2} \sum_{m}^{N_s} (t^m - y(x^m))^2 \]
    \[ \frac{\partial E}{\partial w_b} = - \sum_{m}^{N_s} (t^m - y(x^m)) x^m \]

  approximation for linear neurons:

\[ w(t+1) = w + \epsilon (t^m - y(x^m)) x^m \]

- Minsky-paper 1969: the neural networks can solve only linear problems
Error-correcting learning rules

Rosenblatt-algorithm – binary neurons

\[ w \rightarrow w + \epsilon (v^m - \nu(u^m)) u^m \]

Input \( u^m \)

Required output:

- \( v^m = 1 \)
- \( v^m = 0 \)

Good W

Wrong W

Wrong W

good W

Input \( u^m \)

Required output:

- \( v^m = 1 \)
Error-correcting learning rules

Rosenblatt-algorithm – a binary neuron

\[ w \rightarrow w + \epsilon (v^m - \nu(u^m)) u^m \]

Cone of solutions, Konvex problem

Input \( u^m \)

Required output:
\[ v^m = 1 \]

Good \( W \)

Wrong \( W \)

Input \( u^m \)

Required output:
\[ v^m = 0 \]

Wrong \( W \)
Multi-Layer Perceptron

- Nonlinear separation
- Regression
- Dense in $l_2$ with only 1 hidden layer
- Its representational capabilities is increasing by the number of hidden layers
- Feedforward structures in the neural system, example: visual system
What could be represented by a simple, one layered, feed forward network called perceptron?

It is able to learn many functions, but there are some exceptions such as XOR.

**Fig. 5.2. Architecture of the XOR gate.**

**Fig. 5.3. Disconnected areas of equal output value in the space of input variables.**

Problem:
The linearly inseparable functions are more numerous as the dimension of the problem increases.
In two dimensions the problem can be transformed: this requires a two layered network.

With this two layered network, all the two dimensional Boolean-functions can be learned.

But in higher dimensions?

The weights and the thresholds appropriate to the XOR solution:
A possible solution: increasing the embedding dimension

In three dimension the XOR problem is linearly separable. As the embedding dimension increases, the fraction of linearly inseparable logical functions vanishes.
Soft-max output layer, to represent probabilities

The logistic function maps the activation onto the \([0, 1]\) interval:

\[
y = f(z) = \frac{1}{1 + e^{-z}}
\]

A softmax layer is a generalization of the logistic function for multiple variables:

\[
p_i = y_i = \frac{e^{z_i/T}}{\sum e^{z_i/T}}
\]

Non local, the activity of the neurons depends on the input of all neurons, not only its own input

\[
\frac{\partial y_i}{\partial z_i} = y_i(1 - y_i)
\]

T: temperature parameter
T\(\to\)inf: uniform distribution
T\(\to\)0: converge to Max function

Its derivative is smooth and local,
Simmilar to the derivaive of the logistic function
Error backpropagation

- Input: $z$
- Required output (target): $t_n$
- Actual output: $y$
- Partial derivatives of the error function:
- Gradient method, which converges to a local minimum of the error function.

$$z = xw + b$$

$$y = f(z) = \frac{1}{1 + e^{-z}}$$

$$\frac{\partial E}{\partial w_{bi}} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial z} \frac{\partial z}{\partial w_{bi}}$$

$$\frac{\partial E}{\partial y} = t^n - y(x)$$

$$\frac{dy}{dz} = y(1 - y)$$

$$\frac{\partial z}{\partial w_i} = x_i$$

$$w_i(t+1) = w_i(t) + (y - t^n) y(1-y) x_i$$

$$\delta_j = y_j (1 - y_j) \sum w_{jq} \delta_q$$
Slow convergence along highly correlated variables

Problem:
In case of strongly correlated variables, the gradient is many times almost perpendicular to the direction of the minimum.

Possible solutions:

- Momentum method,
- Hessian matrix
- Hessian free optimizer
- Conjugate gradient method
- Adaptive step length
Recurrent Networks

**Reservoir computing:**
- Context reverberation
- Echo state network
- Fluid state networks

**Possible learning techniques:**
- Error-back propagation in time
- Fixed reservoir, trainable perceptron
- (Long short term memory: Edge-of-chaos)
Hebb's rule

When an axon of cell A is near enough to excite cell B, and repeatedly or consistently takes part in firing it, some growth process or metabolic change takes part in one or both cells such that A's efficiency, as one of cell firing B, is increased" (Hebb, The Organization of Behavior, 1949)

\[
\tau \frac{d w_i}{d t} = v \cdot u_i
\]
Hebb's rule (?) in an experiment at population level

LTP – long term potentiation  LTD – long term depression
Plasticity in the neural system

- Position of the plasticity: synapses, firing thresholds of the postsynaptic neurons (excitability)
- Potentiation, depression
- STP: Calcium dynamics, transmitter depletion, range < 1 minute
- LTP: genexpression (induction, expression, maintenance), NMDA magnesium-block, range > 1 minute
- Correlation between the molecular and the psychological level
Spike-time dependent plasticity (STDP)

- Timing-dependent plasticity:
  - If the postsynaptic spike follows the presinaptic in short time window (causal order) the connection strength increases
  - Spiking in the anticausal order decreases the connection strength
  - Many more variables

- A Hebb-szabály formalizációja:
  lineáris ráta-modellben

\[ \tau_w \frac{d w}{d t} = \nu u \]

Bi & Poo, 1998

Stabilized Hebb rules

- Problems with the Hebb rule:
  - Weights can only increase
  - No competition between the synapses – input selectivity can not be implemented
- Simple solution: ceiling to the weights
- BCM: stabilizing with postsynaptic excitability

\[
\tau_w \frac{d \mathbf{w}}{dt} = v \mathbf{u} (v - \theta_u) \\
\tau_\theta \frac{d \theta_u}{dt} = v^2 - \theta_u
\]

- Sinaptic normalization
  - Subtractive normalization
  
  Global rule, but results in observed connection patterns (Ocular dominance)
- Oja-rule
  
  Local rule, but can generate the observed patterns
Rate-based and spike-time dependent learning rules
Principal component analysis

\[ x_k = \frac{1}{K} \sum_{k=1}^{K} x_{k}\]  
\[ \sigma_k = \sqrt{\frac{1}{K-1} \sum_{k=1}^{K} (x_{k} - \bar{x})^2} \]

\[ S = \text{covar}[y_i,y_j]_{i,j=1}^{N} \approx \left( \frac{1}{K-1} \sum_{k=1}^{K} y_{ki} y_{kj} \right)_{i,j=1}^{N} = \frac{1}{K-1} Y^T \cdot Y \]

\[ S \cdot X' = \lambda \cdot X' \]

\[ Z = X \cdot A \]
Principal component network, derivation of Oja's rule:

\[ w_i(n + 1) = w_i(n) + \eta y(x(n))x_i(n) \]

\[ w_i(n + 1) = \frac{w_i(n) + \eta y(x(n))x_i(n)}{\left(\sum_{j=1}^{m}[w_j(n) + \eta y(x(n))x_j(n)]^p\right)^{1/p}} \]

\[ w_i(n+1) = \frac{w_i(n)}{\left(\sum_j w_j^p\right)^{1/p}} + \eta \left( \frac{y(n)x_i(n)}{\left(\sum_j w_j^p\right)^{1/p}} - \frac{w_i(n)\sum_j y(n)x_j(n)w_j(n)}{\left(\sum_j w_j^p\right)^{1+1/p}} \right) + O(\eta^2) \]

\[ y(x(n)) = \sum_{j=1}^{m} x_j(n)w_j(n) \quad \|w\| = \left(\sum_{j=1}^{m} w_j^p\right)^{1/p} = 1 \]

\[ w_i(n + 1) = w_i(n) + \eta y(n)(x_i(n) - w_i(n)y(n)) \]
The Oja-rule and the principal component analysis

y output
x input
w weight matrix

\[ y = w^T x = x^T w \]

Oja's rule
\[ \Delta w = \alpha (xy - y^2w) \]

Substituting y:
\[ \Delta w = \alpha (xx^T w - w^T xx^T ww) \]

Assuming \( \text{mean}(x) = 0 \) & averaging over the input \( \Rightarrow xx^T = C \)

The convergence point: \( \Delta w = 0 \)
\[ Cw - w^T Cww = 0 \]

This is an eigenvector equation!

E modification for Independent Component analysis
\[ \Delta w = \alpha (xy^3 - w) \]

If \( C = 1 \)
Independent component analysis (ICA)
Independent component analysis (ICA)
The somatosensory map

[Image of a brain diagram showing somatosensory, visual, auditory, gustatory, olfactory, and motor areas.]
Kohonen's self-organizing map

\[ D_{lk} = \sqrt{\sum_{n=1}^{N} (x_{kn} - w_{n}^{l})^2} \]

Winner take all

\[ \overline{W}^{j}(t + 1) = \overline{W}^{j}(t) + \eta(t) N(c,r)[X_k - \overline{W}^{j}(t)] \]

Iteration 0  Iteration 200  Iteration 600  Iteration 1900
Kohonen's self-organizing map

Generates feature maps  Captures the structure in the inputs
Associative memory

- Heteroassociation
  - Example: place-object
- Autoassociation
  - From partial pattern the original
- Difference between the memory of a computer and an AM is the addressing

- Capacity: how many patterns can be stored and retrieved (non-unique definition)
- Stability: for each patterns, we want the most similar (non-unique definition)
Attractor networks

- Types of attractors
  - Fixed points
  - Periodic
  - Chaotic
- Basin of attractions
- Implementation: recurrent neural networks
- Attractor formation: synaptic weights
  - Offline learning
  - Online learning
  - One-shot learning
- Retrieval: convergence from an initial condition
MODIFICATIONS OF THE CLASSICAL SCENARIO

Fixed points vs. strange attractors

Not only fixed points
but also limit cycle
or strange attractors
might be involved

- neurophysiological experiments,
  theoretical studies
- structural conditions for the possible mechanism
  for the generation of rhythms and chaos can be given
  based on the notions of qualitative stability and instability

W. J. Freeman
Hopfield-networks

- Associative memory
- Binary MCP-neurons
- Patterns: binary vectors
- Symmetric weight matrix
- Dale's law is violated: a cell can be excitatory and inhibitory parallel
- Recurrent networks in the brain (dominantly): hippocampus CA3 region, ...

Offline learning patterns to learn:
\[ \{ s^1 \ldots s^N \} \]

\[
W_{ij} = \frac{1}{N} \sum_n^{N} s_i^n s_j^n \quad \text{Hebb's rule}
\]

\[
x^{t+1} = sgn(Wx^t - \theta) \quad x_k^{t+1} = sgn(\sum_i^K W_{ik}x_i^t - \theta_k)
\]

- Iterations: synchronous or sequential
The dynamics of the Hopfield network

- Stability-analysis of a nonlinear system: The definition of the „energy“ of a state by introducing a Lyapunov-function:
  - Bounded
  - The iterative dynamics always decreases (or increases) it.

It it exists, that we showed, that the system will converge to a fixed point for every input patterns

- The Lyapunov-function of the Hopfield network:

\[ E = -\frac{1}{2} x^T W x - \theta x \]

- The learning builds attractors at the stored patterns, but not only there (spurious attractors)
- The HN can be used to optimize probles with quadratic forms
The capacity of the HN

• Information theoretical capacity
  • The patterns can be considered as a set of random variables from Bernoulli-distribution
    \[ P(s^n_i = 1) = P(s^n_i = 0) = 0.5 \]
  • The convergence is required for all patterns
    \[ \lim_{n \to \infty} P(s^a = \text{sgn}(Ws^a)) = 1 \quad \forall \ a = 1 \ldots M \]
  • Then it can be shown (with many approximation) that
    \[ M \approx \frac{N}{2 \log_2 N} \]
  • In case of the CA3 hippocampal subregion
    • Cc. 200000 neurons, cc. 6000 patterns to store
  • Different estimations
    • Let us consider the sparsity of the patterns
      \[ P(s^n_i = 1) = \alpha \]
      \[ M \approx N \frac{1}{\alpha \log_2 \frac{1}{\alpha}} \]
Boltzmann machine

- To represent probability distributions – statistical interdependences between variables
- Stochastic state transition
  \[ I = W u + M v \]
  \[ P(v_{a}^{t+1} = 1) = \frac{1}{1 + e^{-I_{a}}} \]
- The limit distribution
  Energy:
  \[ E(v) = -v^{T}Wu + \frac{1}{2}v^{T}Mv \]
  Boltzmann-distribution:
  \[ P(v) = \frac{e^{-E(v)}}{\sum_{v} e^{-E(v)}} \]
Learning with Boltzmann machine

- Supervised learning, only for W, similarly M
- Error: Kullback-Leibler-divergency between the targeted and actualy generated distribution

\[ D_{KL}[P(v|u), P(v|u, W)] = \sum_v P(v|u) \ln \frac{P(v|u)}{P(v|u, W)} \]

averagon the the inputs: (in stead of the \( P(v|u) \) weighted outputs)

\[ \langle D_{KL} \rangle = -\frac{1}{N_s} \sum \ln P(v^m|u^m, W) - K \]

- Gradient descent — for only one input

\[ \frac{\partial \ln P(v^m|u^m, W)}{\partial W_{ij}} = v_i^m u_j^m - \sum_v P(v|u^m, W) v_i u_j^m \]

from the Boltzmann-distribution

- Delta-rule — the average for the all possible outputs is approximated with the actual one

\[ W_{ij} \rightarrow W_{ij} + \epsilon_w (v_i^m u_j^m - v_i(u^m) u_j^m) \]

two phases: Hebbian, anti-Hebbian

- Unsupervised

\[ D_{KL}[P(u), P(u, W)] \]
Reinforcement Learning

- State space: the possible values of the sensory (or other input) variables
- Reward: in certain states we get information from the success
- Actions: the agent realizes a state transition (at least tries)
- aim: to maximize the reward in long run
- Value function: the utility of the states
- Representation of the value function:
  - Table (machine learning)
  - General function approximator, for example a feed-forward neural network
    - (embedden supervised learning)
Reinforcement learning

Solutions

- Model-based: state and state-transition representation value function and action policy

  Types: Direct utility estimation (DUE)
  Adaptive dynamic programming (ADP)
  Temporal difference (TD)

- Model-free:
  Q-learning: state-action pair value association
  Strategy searching
Temporal difference learning

- Using the prediction error for the learning
- Updating the value function in a neural representation:
  \[ w(\tau) \rightarrow w(\tau) + \epsilon \delta(t)u(t-\tau) \quad \delta(t) = \sum_\tau r(t+\tau) - v(t) \]

- Calculation of the prediction error:
  - (IN principle we need to know the full reward in the future)
  - On step local approximation
    \[ \sum_\tau r(t+\tau) - v(t) \approx r(t) + v(t+1) \]

- If the environment is suitable, it converges to an optimal strategy
- The error can be back propagated to the previous states as well (eligibility trace, similar to the error back-propagation)
- Action selection: exploration vs. exploitation
TD with feed-forward neural network

- Gerald Tesauro: TD-Gammon
  - Feedforward network
  - Input: the states which can be achieved by the possible actions
  - Output: values (winning probabilities)
- In each step, the error is calculated
  - Based on the reward signal
- Result: comparable with the best human players
The effect of reward in dopaminergic cell of basal ganglia

An interpretation:

Dopamine cells signal the difference between the expected and received reward.
Problems to solve in learning systems

- Bias-variance dilemma
  - Structural error: the model (even with optimal parameters) can differ from the function to approximate (e.g., Linear model fitting for cubic data)
  - Approximation error: infinite datapoints are needed for precise parameter tuning

- Accuracy vs. Predictive power
  - The models with too much parameters fits well, but generalize poorly
  - May have lower explanatory ability