Neural networks an introduction

Akito Sakurai

Contents

- Overview
  - Neurons and models
- Implementation in early days
  - Perceptrons
- An introduction to MLP

What are neurons?

- "There is no such thing as a typical neuron", Arbib, 1997

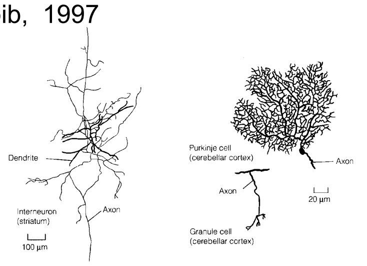


Fig. 1. The morphologies of three common types of neuron. The full length of the axons is not shown. The bifurcating axon of the granule cell extends for several millimeters in each direction. Note how the axon of the interneuron branches extensively.

A typical(?) neuron

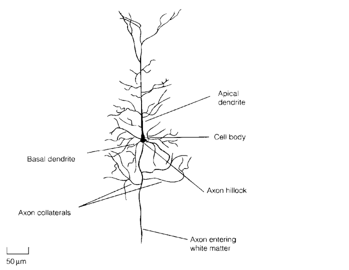
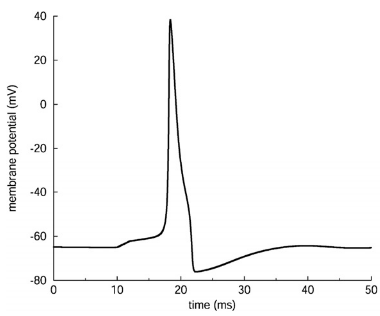


Fig. 1. Key features of a neuron. A drawing of a pyramidal cell showing the distribution of neurites (dendrites and axon).

Dynamics of neuron activity



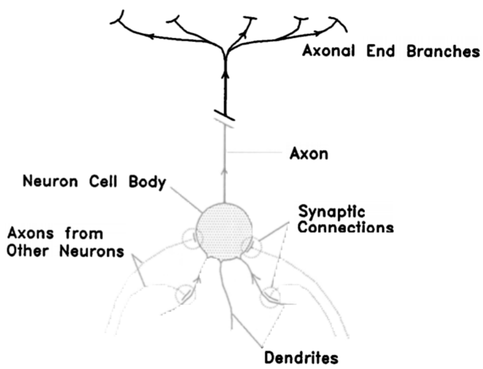
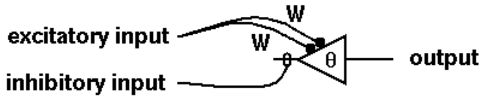


Figure 1.1 A typical neuron. It receives excitatory and inhibitory signals from other neurons by way of the many synaptic connections (circled) they make onto the neuron's cell body and its extended tree of dendritic branches. It sums those various incoming signals and emits an appropriate signal down its own axon, to make contact with further neurons.

Paul M. Churchland (1996) The Engine of Reason, The Seat of the Soul

McCulloch and Pitts

- Warren S. McCulloch and Walter Pitts (1943) ``A logical calculus of the ideas immanent in nervous activity'', ***Bulletin of Mathematical Biophysics*, 5**: 115-133.



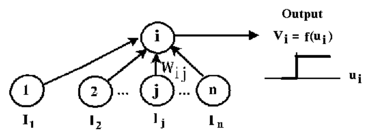
- A very simplified (mathematical, computational) model
- Any logical function can be realized by connecting the model neurons.

A model neuron

- A model neuron receives an input vector, processes it, and outputs a value.

  $output = actiavtion\_function(W \cdot Input)$

- The model in early ages use the step function as an activation function. Currently sigmoid or rectified linear function are commonly used.



Perceptron: its structure

- Rosenblatt, F. (1957). “The perceptron: A perceiving and recognizing automaton (project PARA).”, Technical Report 85-460-1, Cornell Aeronautical Laboratory.
- Rosenblatt, F. (1962). “Principles of Neurodynamics.”, Spartan Books, New York.

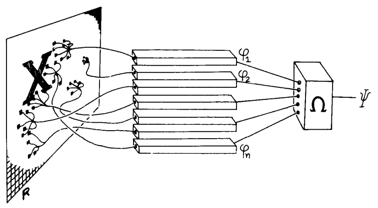


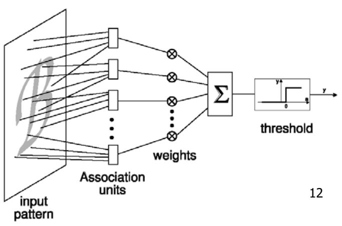
FIGURE 1. The one-layer perceptron analyzed by Minsky and Papert. (From *Perceptrons* by M. L. Minsky and S. Papert, 1969, Cambridge, MA: MIT Press. Copyright 1969 by MIT Press. Reprinted by permission.)

Contents

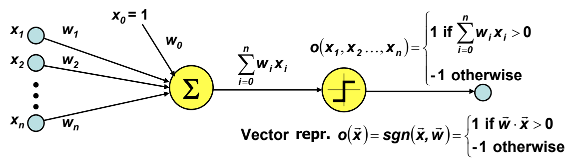
- Overview
  - Neurons and models
- Implementation in early days
  - Perceptrons
- An introduction to MLP

Perceptron: a polysemic word

- Perceptron: used to mean varying concepts
  - Linear threshold unit: the next slide
  - Original perceptron: as follows
  - Sigmoid unit or any other similar units
  - Network of sigmoidal units: called multi-layer perceptron or MLP
  - Network of linear threshold units: MLP but quite rarely used in this meaning
- In this class, we assume it means a sigmoidal network
- Original perceptron
  - Rosenblatt 1962
  - Minsky and Papert 1969

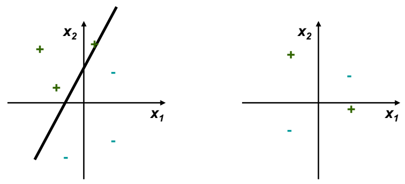


Perceptron



- Perceptron: a single neuron unit model
  - Alias: Linear Threshold Unit (LTU) or Linear Threshold Gate (LTG)
  - Net input: a value of a linear function applied to inputs $net = \sum_{i=0}^{n} w_i x_i$
  - Net output: a value of the threshold function applied to the net input (threshold $\theta = w_0$)
    - A function to obtain the net output by applying it to net input is called activation function
- Perceptron Networks
  - A network of perceptrons connected through weighted links w_i
  - Multi-Layer Perceptron (MLP):

Decision boundary of perceptron



Example A Example B

- Perceptron: can easily represent many important functions.
  - Logical functions (McCulloch and Pitts, 1943)
  - e.g., with simple integer weights $AND(x_1, x_2)$, $OR(x_1, x_2)$, $NOT(x)$
- Some functions are not representable
  - e.g., linearly non-separable functions (just a paraphrase)
  - To circumvent: construct a network of perceptrons

Perceptron learning algorithm

- Perceptron Learning Rule (Rosenblatt, 1959)
  - Idea: suppose that for each input vector, an output value is given. Then by updating weights, the perceptron will become able to output the proper values.
  - The unit assumes binary value; for a perceptron unit, the weight update formula is
  - $$w_i \leftarrow w_i + \Delta w_i$$ $$\Delta w_i = (t - o)x_i$$

    where $t = c(x)$ is a target value for x, o is a current perceptron output value. The second formula is sometimes expressed as $\Delta w_i = r(t-o)x_i$ where r is called a learning rate. Because in the case of perceptron, r could be any positive value, giving equivalent results, r =1 is preferred in the formula.
  - When the training set D is linearly separable , the algorithm converges in finite time. Some literature requires r to be small enough, but it is wrong for the perceptron.

In a different way

Initialization: $\vec{w}$ is any vector. $\vec{x} \in F = F^+ \cup F^-$

Repeat

Select all $\vec{x} \in F$ in sequence in arbitrary order

If $\vec{w} \cdot \vec{x} > 0$ and $\vec{x} \in F^+$ then continue;

If $\vec{w} \cdot \vec{x} \leq 0$ and $\vec{x} \in F^+$ then FixPlus and continue;

If $\vec{w} \cdot \vec{x} \leq 0$ and $\vec{x} \in F^-$ then continue;

If $\vec{w} \cdot \vec{x} > 0$ and $\vec{x} \in F^-$ then FixMinus and continue;

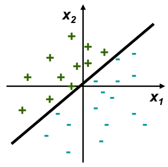
until no errors (neither FixPlus nor FixMinus is called)

FixPlus: $\vec{w} := \vec{w} + \vec{x}$

FixMinus: $\vec{w} := \vec{w} - \vec{x}$

Linearly separable?

- Definition
  - Suppose 0 or 1 is a label $f(x)$ of x in D. if there exists w and θ s.t.
  - $f(x) = 1$ if $w_1x_1 + w_2x_2 + \ldots + w_nx_n \geq \theta$, 0 otherwise
  - D is called linearly separable.θ is called a threshold.
- Linearly separable?
  - Note: D being linearly separable does not mean its population is so.
  - disjunction: $c(x) = x_1' \vee x_2' \vee \ldots \vee x_m'$
  - m of n: $c(x)$ = at least 3 of $(x_1', x_2', \ldots, x_m')$
  - exclusive OR (XOR): $c(x) = x_1 \oplus x_2$
  - DNF: $c(x) = T_1 \vee T_2 \vee \ldots \vee T_m$; $T_i = l_1 \wedge l_1 \wedge \ldots \wedge l_k$
- Transformation of expression
  - Can we transform a linearly non-separable problem to a linearly separable one?
  - If it is possible, is it meaningful? Realistic?
  - Is it an important problem?



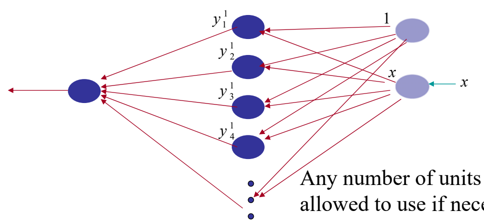
Linearly Separable (LS) Data Set

Convergence of the perceptron algorithm

- Perceptron convergence theorem
  - Claim: if there exists a weight vector w which is consistent with the training dataset (i.e., linearly separable), the perceptron learning algorithm converges.
  - Proof: Searching spaces are in order with a limit (width of wedge (searching space) decrease) – Ref. Minsky and Papert, 11.2-11.3
  - Note 1: how many repetitions are necessary until convergence?
  - Note 2: what happens if it is not linearly separable?
- Perceptron cycling theorem
  - Claim: If a training dataset is not linearly separable, weight vectors obtained during the perceptron algorithm form a bounded set. If the weights are integers, the set is finite.
  - Proof: If the initial weight vector is long enough, its length is shown to be unable to become longer; proven by a mathematical induction with the dimension n. – Minsky and Papert, 11.10
- How to make it robust? Or to make it more expressive?
  - Goal 1: to develop an algorithm which finds a better approximation
  - Goal 2: to develop a new architecture to go beyond the restrictions

Universal approximation theorem

- A neural network with a single hidden layer can approximate any continuous function within a required accuracy if any finite number of hidden units are allowed to use

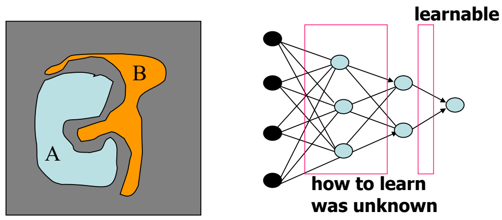


Any number of units are allowed to use if necessary.

Perceptron's ability

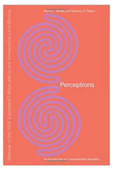
- Perceptrons can
  - Recognize characters (alphabets)
  - Recognize types of patterns (forms etc.)
  - Learn with a splendid learning algorithm
    - Perceptron learning algorithm, as was seen, is able to find a solution of any problems that has solutions by perceptrons.
    - Note: Existence of solutions does not help us to find a solution.

Unfortunately



When an output is not as is required, some weights must be updated, which is easily inferred. But how much they are changed was not known at the time.

In short



- There exist problems that cannot be solved, although a solution exists in a form of networks.
  - Parity or XOR problem
  - Connectivity of patterns
  - In general, linearly non-separable problems
- Marvin L. Minsky and Seymour Papert (1969), "Perceptrons", Cambridge, MA: MIT Press
  - Proved that perceptron is not capable to solve many problems.
  - We can construct a network! Yes, but "we" must do it.
- McCulloch & Pitts neuron network is equivalent to Turing machine (i.e. "universal"). OK but does it help us?
  - If we do not know how to make it learn, it is useless.
    - Does a learning algorithm of the network exist?

Exception at the time

Cognitron

Sophisticated structure
Specific learning algorithm
Too advanced to be popular

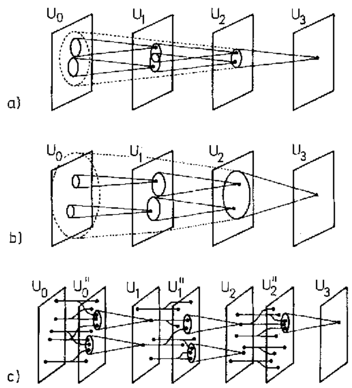


Fig. 4 a-c. Three possible methods for interconnecting layers. The connectable area of each cell is differently chosen in these three methods. Method c is adopted for the cognitron discussed in this paper

K.Fukushima, Cognitron: A self-organizing multilayered neural network, Biological Cybernetics 1975

Exception at the time

Neocognitron

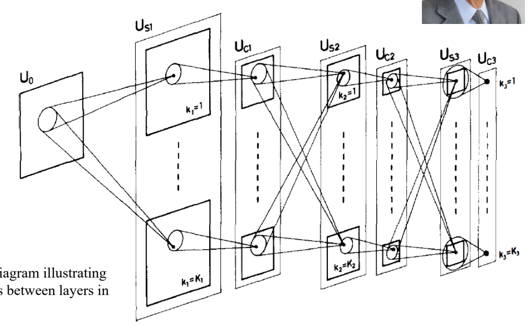


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

K. Fukushima, Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position, Biological Cybernetics (1980)

S-cell in Neocognitron

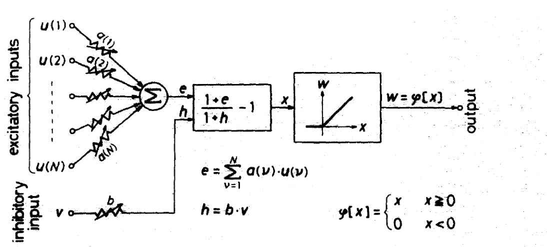


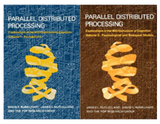
FIGURE 1. Input-to-output characteristics of an S-cell: A typical example of the cells employed in the neocognitron.

K. Fukushima, Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition, NeuralNetworks, Vol. 1, pp. 119-130, 1988

Contents

- Overview
  - Neurons and models
- Implementation in early days
  - Perceptrons
- MLP appears

26



PDP appeared

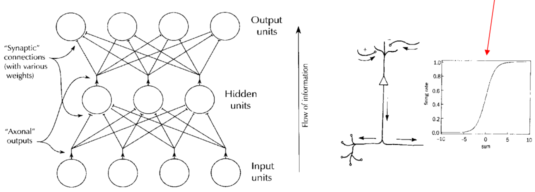
- The book "Perceptrons" was said to have delayed researches in the field by two decades (pros and cons exist)
- A turning point: D.E. Rumelhart, J.L. McClelland, eds., "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", MIT Press, 1986.
  - A compilation of articles: from mathematics to philosophy
  - Many successful experiments on multi-layer networks
  - Proposed "error back propagation algorithm": unexpected influence on learning algorithms.
  - [Because similar techniques to BP were found before PDP (Amari 1967; Werbos, 1974, "dynamic feedback"; Parker, 1982, "learning logic"), reinvention/rediscovery would be a better word to describe it. But it has had a profound influence.]

27

Two important inventions

Error Back Propagation

- Basically it is for multi-layer feed-forward networks but could be applied to other types of networks.



- Weights are updated in proportion to backpropagated errors

28

Reasons why PDP succeeded

- Changed the activation function of units from the threshold function to the sigmoid function
- Formulated the learning problem in the error minimization problem. E.g.,

$$E(w) = \sum_{\text{all samples: } x_k} \left(f(x_k; w) - \text{target value for } x_k\right)^2$$

- Solved the (nonlinear multivariate) problem by a naïve method (steepest descent)

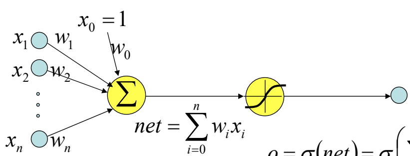
29

Error Minimization

- Do not pursue a complete solution (error=0)
  - Our ability may be limited
  - Samples may contain errors
  - Samples may be of probabilistic events
- Consider (too naively) the sum of squares of the difference between targets and current outputs as the error.
- Find out weights that minimize the error

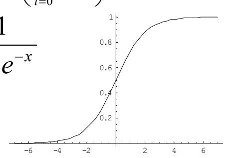
30

Sigmoid function



$$o = \sigma(net) = \sigma\left(\sum_{i=0}^{n} w_i x_i\right)$$

Quite often use σ : $\sigma(x) \equiv \dfrac{1}{1+e^{-x}}$



A method of minimization

- Solve equations obtained by equating the gradients to be 0

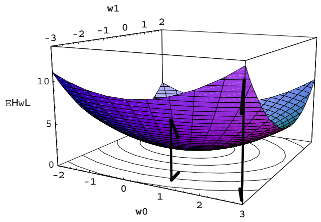
$$E(w) = \sum_{\text{for all saples } x_k} \left(f(x_k; w) - \text{target of } x_k\right)^2$$

$$\frac{\partial E}{\partial w} = 0$$

- Because f is non-linear, we cannot solve it.
- An iterative method is to be sought, i.e., a method that gives us $w_1, w_2, w_3 \cdots$ such that $E(w_1) > E(w_2) > E(w_3) > \cdots$

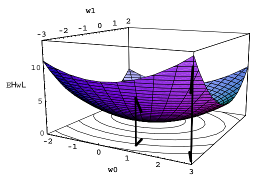
Iterative method for minimization

- Many efficient methods have been proposed
- The simplest one is the steepest descnet
  - Steepest ascent method give us a maximum



Direction of the steepest descent is normal to the contour of the error

Mathematically



- Steepest descent

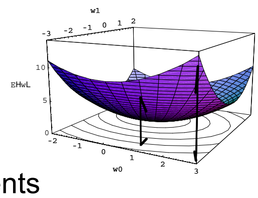
$$-\frac{\partial E}{\partial w}$$

- Update w a bit along the direction

$$w^{new} \leftarrow w^{current} - \eta \frac{\partial E}{\partial w}$$

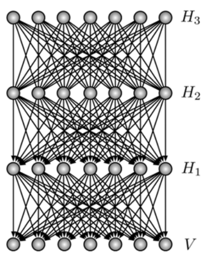
- The learning rate η>0 is to be defined appropriately

Actually

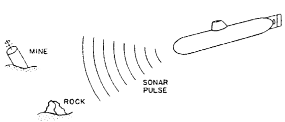


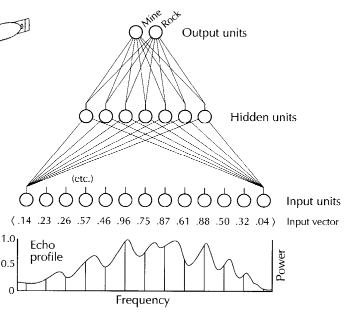
- A method to simplify the calculation of gradients is needed, because there are hundreds (at the time) or hundreds of thousands variables exist.

$$w^{new} \leftarrow w^{current} - \eta \frac{\partial E}{\partial w}$$



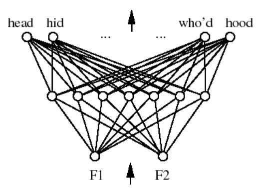
What was done: sound classification

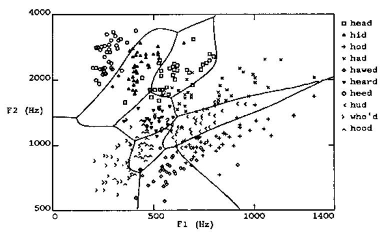




(Gorman & Sejnowski, NC 1(1), 75-89 (1988))

What was done: speech recognition



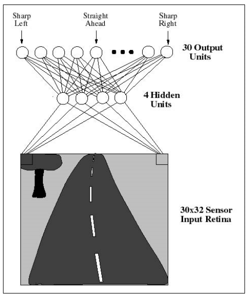


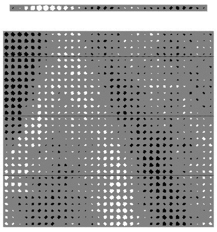
http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-3/www/ml.html

ALVINN :
An Autonomous land vehicle in a neural network

- Made a tour on the interstate

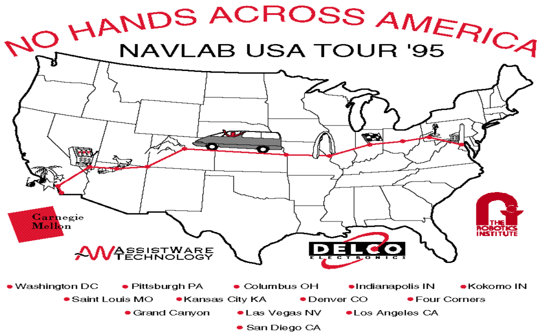
(Dean Pomerleau 1995)





http://web.mit.edu/6.034/wwwbob/L9/node10.html

Navlab



Summary and a bit beyond it

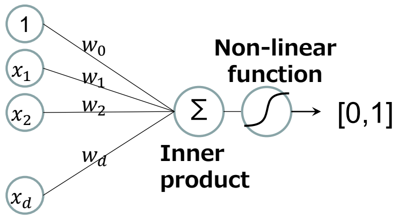
Problems to be solved

- Inputs : *N* labeled samples $(\mathbf{x}_i, \mathbf{b}_i),\ i = 1, \dots, N$
  - $\mathbf{x}_i \in \boldsymbol{R}^d$ is a d-dimensional feature vector
  - $\mathbf{b}_i \in \boldsymbol{R}^c$ is a *c*-dimensional label
  - $\mathbf{x}_i$ belongs to a class k → $\mathbf{b}_i = (0, \dots, 1, \dots, 0)$ where only k-th element is 1.
- Outputs: a function $\boldsymbol{g}(\mathbf{x})$
  - $\mathbf{x}_i \in \boldsymbol{R}^d$, $\boldsymbol{g}(\mathbf{x}) \in \boldsymbol{R}^c$
  - It classifies the training samples as correctly as possible
  - $g(\mathbf{x}_i) = (b_1, b_2, \dots, b_c)$
    - $\mathbf{x}_i$ belongs to k-th class

      $\rightarrow b_k > b_i \quad i \neq k$

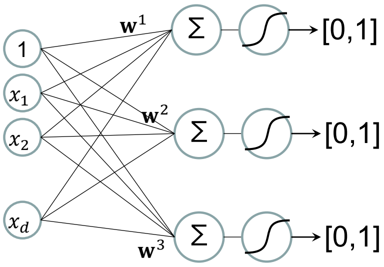
A unit in neural networks

- A unit or a neuron is a basic construct of neural networks.
- It has a weight vector $\mathbf{w}$ and a nonlinear function f.
- It returns a value in [0,1] by first obtaining an input vector $\mathbf{x}$, calculating $\mathbf{w}^T\mathbf{x}$ the inner product with the weight, and applying it to the non-linear function f
- **Note: the returned value becomes larger when the input vector $\mathbf{x}$ is more similar to the weight vector $\mathbf{w}$.**



Connecting units in parallel

- Each unit returns large value when its input vector is similar to its weight vector.
- When multiple units run in parallel, multiple class classification becomes possible.



The input vector is similar to the weight vector $\mathbf{w}^1$, the output value is large

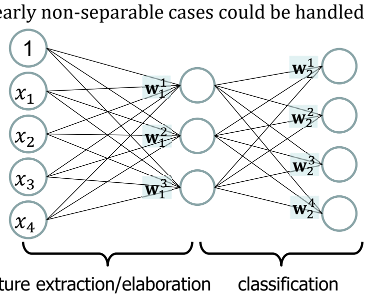
The input vector is similar to the weight vector $\mathbf{w}^2$, the output value is large

The input vector is similar to the weight vector $\mathbf{w}^3$, the output value is large

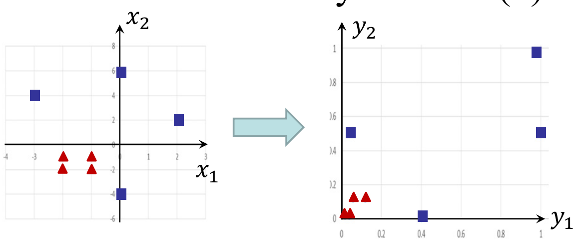
Connecting units in series

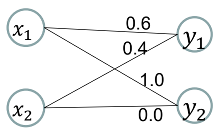
- Insert intermediate layers between input and output layers.
- Input vector is transformed such that it could be discriminated well

→ linearly non-separable cases could be handled



Intermediate layer's role (1)





$\mathbf{w}_1 = (0.6, 0.4)$

$\mathbf{w}_2 = (1.0, 0.0)$

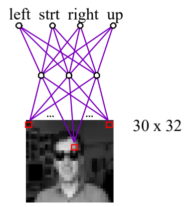
$f(x) = \frac{1}{1+\exp(-2x)}$

Class 1
(-2,-2) → (0.18, 0.18)
(-1,-1) → (0.12, 0.12)
(-1,-2) → (0.06, 0.12)
(-2,-1) → (0.04, 0.02)

Class 2
(0,-4) → (0.04, 0.50)
(2, 2) → (0.98, 0.98)
(0, 6) → (0.99, 0.50)
(-3, 4) → (0.40 , 0.00)

Very old story

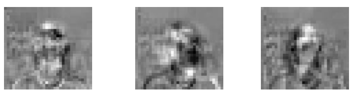
Intermediate layer's role (2) from Face direction recognition

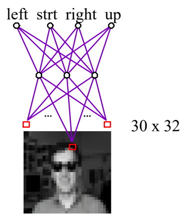


http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-3/www/ml.html

A result in intermediate layer

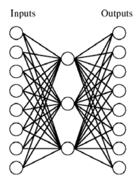
- Weights to intermediate layer





Intermediate layer's role (3)

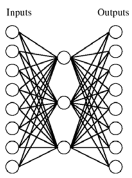
- Can this be learned?



Input		Output
10000000	→	10000000
01000000	→	01000000
00100000	→	00100000
00010000	→	00010000
00001000	→	00001000
00000100	→	00000100
00000010	→	00000010
00000001	→	00000001

Result of learning

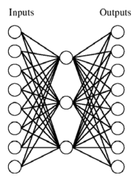
Information compression – the root of encoder network



Input						Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

Result of learning

Information compression – the root of encoder network



Input						Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

Intermediate layer's role (4) from NETTalk

- NETTalk: pronunciation of words
  - Many rules available
  - Many exceptions

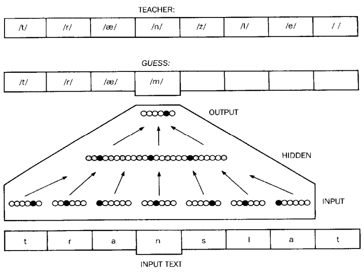


Table 1 Symbols for phonemes used in NETtalk.

Symbol	*Phoneme*	*Symbol*	*Phoneme*
/a/	father	/C/	chin
/b/	bet	/D/	this
/c/	bought	/E/	bet
/d/	debt	/G/	sing
/e/	bake	/I/	bit
/f/	fin	/K/	sexual
/g/	guess	/L/	bottle
/h/	head	/M/	absym
/i/	Pete	/N/	button
/k/	Ken	/O/	boy
/l/	let	/Q/	quest
/m/	met	/R/	bird
/n/	net	/S/	shin
/o/	boat	/T/	thin
/p/	pet	/U/	book
/r/	red	/W/	bout
/s/	sit	/X/	excess
/t/	test	/Y/	cute
/u/	lute	/Z/	leisure
/v/	vest	/@/	bat
/w/	wet	/!/	Nazi
/x/	about	/#/	examine
/y/	yet	/*/	one
/z/	zoo	/|/	logic
/A/	bite	/^/	but

NETTalk: cluster analysis

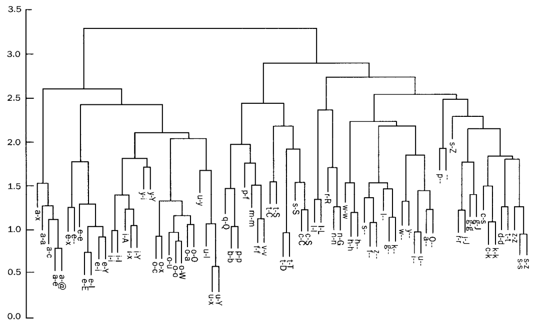


Figure 7.6 Hierarchical clustering of hidden units for letter-to-sound correspondences. The vectors of average hidden unit activity for each correspondence ("l-p" for letter l and phoneme p) were successively merging from right to left in the binary tree. The scale at the top indicates the Euclidean distance between the clusters. (From Sejnowski and Rosenberg 1987.)

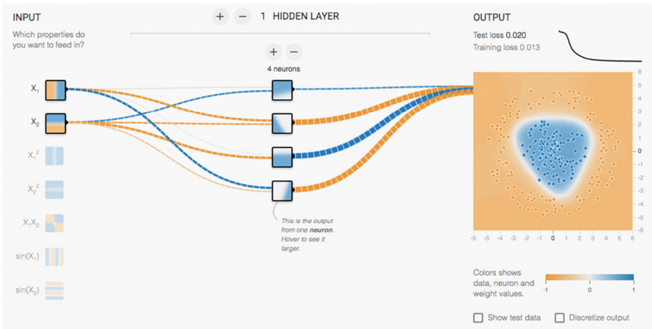
Interesting findings

Interesting things were found, i.e.,

In intermediate layers, some representation which were not imagined by researchers was observed.

- The representations are meaningful.
- They are information compression and extraction.
- This will give again profound influence in the future (now current) research.

MLP: a demo



http://playground.tensorflow.org/