NUS AI SUMMER EXPERIENCE 2019

Artificial Neural Network

Li Xinke

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

NUS AI SUMMER EXPERIENCE 2019

Department of ISEM, NUS
Xinke.li@u.nus.edu

Computer vision, 3D CNN, Video Analysis

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

1

Pigeons as art experts

Follow the story line of AI

Pigeons as art experts (Watanabe. et al. 1995)

Pigeon in a skinner box

Present paintings of two different artists (e.g., Chagall/Van Gogh)
Reward the pigeon for pecking when it is presented a particular artist (e.g., Van Gogh)

Marc Chagall (1887-1985)

Russian Jewish modernism artist

Marc Chagall

Vincent Willem van Gogh (1853-1890)

Dutch Post-Impressionist artist

Vincent Van Gogh

Can pigeon recognize new paintings?

The most remarkable thing is that it is still 85% successful!

Can a computer achieve the same feat on the new paintings?

Impossible at this moment!

Can pigeon recognize new paintings?

The most remarkable thing is that it is still 85% successful!

Can a computer achieve the same feat on the new paintings?

Impossible at this moment!

Even the Pigeons can beat the computer!

So the pigeons do not simply memorize the pictures.

They can extract and recognize patterns (the styles of the two artists);

They learn from their training process and make predictions on the new ones.

Can pigeon recognize new paintings?

The most remarkable thing is that it is still 85% successful!

Can a computer achieve the same feat on the new paintings?

Impossible at this moment!

Even the Pigeons can beat the computer!

So the pigeons do not simply memorize the pictures.

They can extract and recognize patterns (the styles of the two artists);

They learn from their training process and make predictions on the new ones.

MOTIVATION

So the human or the pigeon brains must do something very different from the computer!

NATIONAL UNIVERSITY OF SINGAPORE
DEPARTMENT OF ISEM

A biological neural network is a *massively parallel distributed processor* made up of simple processing unit, which has a natural propensity for *storing experiential knowledge* and making it available for use.

01	02	03
It employs a massive inter-connection of "simple" computing units - *neurons*.	It is capable of organizing its structure consists of many neurons, to perform tasks that are many times faster than the fastest digital computers nowadays.	It can learn Knowledge obtained from the data/input signals provided.

The artificial neural networks are largely inspired by the biological neural networks, and the ultimate goal of building an intelligent machine which can mimic the human brain.

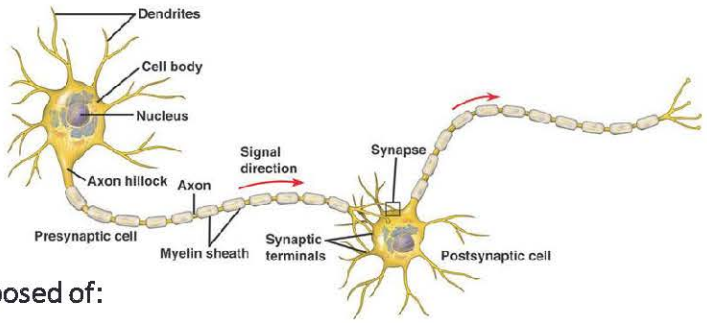
01	02	03
It employs a massive inter-connection of "simple" computing units - *neurons*.	It is capable of organizing its structure consists of many neurons, to perform tasks that are many times faster than the fastest digital computers nowadays.	It can learn Knowledge obtained from the data/input signals provided.

2

AI Story 1: Motivation

Story 1 Motivation

Biological Neurons: What do they look like?



A typical biological neuron is composed of:

A cell body;

Hair-like dendrites=> input channels

Axon=> output cable; it usually branches

Story 1 Motivation

Biological Neurons: Behavior of Neurons

The neuron responds to many sources of electric impulses in three ways:

Some inputs excite the neuron;

Some inhibit it;

Some modulate its behaviour.

If the neuron becomes sufficiently excited, it responds ("fires") by sending an electric pulse--(a spike)-down its output cable—(its axon).

The spikes travel down each branch and subbranches until eventually the axon contacts many other neurons and so influences their behaviors.

Story 1 Motivation

Biological Neurons: Behavior of Neurons

The neuron responds to many sources of electric impulses in three ways:

Some inputs excite the neuron;

Some inhibit it;

Some modulate its behaviour.

If the neuron becomes sufficiently excited, it responds ("fires") by sending an electric pulse--(a spike)-down its output cable—(its axon).

The spikes travel down each branch and subbranches until eventually the axon contacts many other neurons and so influences their behaviors.

When you are learning something, what have changed in your brain? The neurons or the connections (synapses)?

Synaptic plasticity is the ability of the connection, or synapse, between two neurons to change in strength

Story 1 Motivation

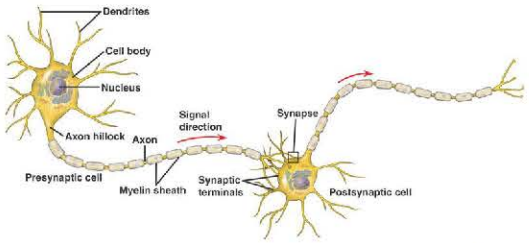
Biological Neurons: Capability of Neurons

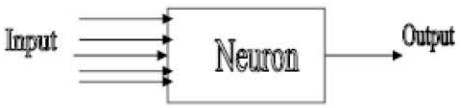
The major job of neuron

1. It **receives information**, usually in the form of electrical pulses, from many other neurons.
2. It does what is, in effect, a complex dynamic **sum of these inputs**.
3. It **sends out information** in the form of a stream of electrical impulses (action potential) down its axon and on to many other neurons.
4. The connections (synapses) are crucial for **excitation, inhibition or modulation** of the cells.
5. **Learning** is possible by adjusting the synapses!

Story 1 Motivation

let us do some math modelling!





Simplest model:

$$y = \sum_{i=1}^{m} x_i$$

So the neuron is always on fire!
The real neurons only fire when they are sufficiently excited!
When m is large, the output can be huge!
It means that the firing rate could be very big. Is this reasonable?
There is also an upper bound on the firing rate!

Modified model:

$$y = \varphi(\sum_{i=1}^{m} x_i - b)$$

Story 1 Motivation

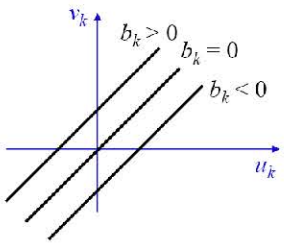
Why do we need the threshold (bias) b?

The neuron will not fire till it is "high" enough!

Actually, the addition of bias means more flexibility of model.

Learning process could be faster with bias. (Later)

By the way, we can also initialize bias properly to train model efficiently.



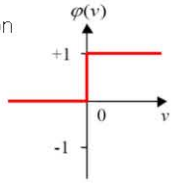
$$y = \varphi(\sum_{i=1}^{m} x_i - b)$$

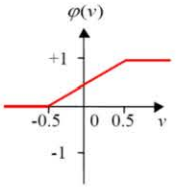
What is the simplest function which has a lower bound and an upper bound in the output?

The Squash Function

Also called activation function

It must be non-linear





Squash Function

3

AI Story 2: Beginning

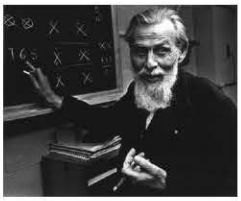
Story 2 Beginning

The beginning of the artificial neural networks: *McCulloch and Pitts*,1943

1.McCulloch was an American neuro-physiologist. He studied philosophy and psychology at Yale(B.A. 1921) and Columbia(M.A. 1923). Receiving his MD in 1927, he was an intern at a Hospital before returning to academia in 1934.

2.Pitts was a logician whose life was more colorful

3. In early 1942,McCulloch invited Pitts, who was homeless, to live with his family. In the evenings McCulloch and Pitts collaborated. This led to their famous paper in 1943.



Warren Sturgis McCulloch (1898-1969)

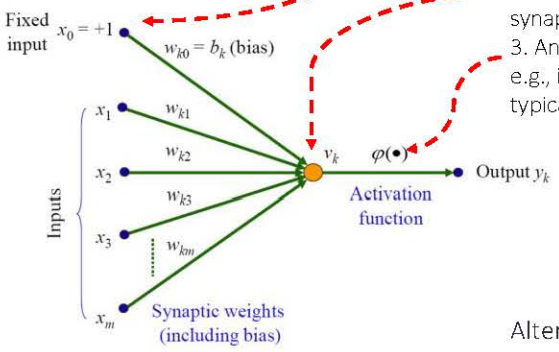
Walter Pitts (1923-1969)

4. In 1951, Wiener set up a neuro-group at MIT with Pitts and McCulloch. Pitts wrote a thesis on the properties of neural nets connected in three dimensions. Pitts was described as an eccentric, refusing to allow his name to be made publicly available. He refused all offers of advanced degrees or official positions at MIT as he would have to sign his name.

5. Wiener suddenly turned against McCulloch because his wife Margaret Wiener hated McCulloch. He broke off relations with anyone connected to him including Pitts. This sent Walter Pitts into 'cognitive suicide'. He burnt the manuscript on three dimensional networks and took little further interest in work.

Story 2 Beginning

M-C Model



1. A set of synapses or connecting links: characterized by a weight or strength of its own.
2. An adder for summing the input signals, weighted by the respective synapses of the **neuron** (a linear combiner).
3. An activation function for limiting the amplitude of the neuron output, e.g., it limits the permissible amplitude range of the output signal, typically [0, 1] or [-1, 1].

$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$y_k = \varphi(\overbrace{u_k + b_k}^{v_k})$$

Where $x_1 \ldots x_k$ are the input signals;
$w_{k1} \ldots w_{km}$ are the synaptic weights of neuron k;
u_k is the linear combiner output due to the input signals;
b_k is the bias;
$\varphi()$ is the activation function; and y_k is the output signal of the neuron.

Alternatively, we may reformulate the model as follows:

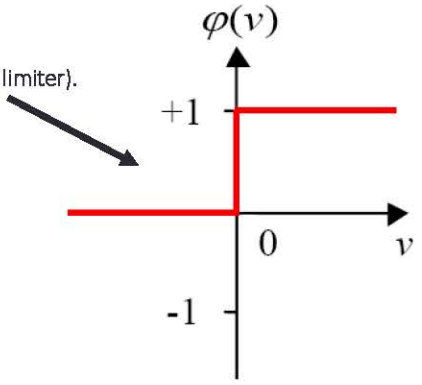
$$v_k = \sum_{j=0}^{m} w_{kj} x_j \qquad \text{and} \qquad y_k = \varphi(v_k)$$

Note: we have added a new input: $x_0 = 1$ and its synapse weight is $w_{k0} = b_k$.

Story 2 Beginning

EXAMPLE: A neuron *j* receives inputs from four other neurons whose activity levels are 10, -20, 4 and -2. The respective synaptic weights of neuron *j* are 0.8, 0.2, -1.0, and -0.9. Calculate the output of neuron *j* for the following two situations

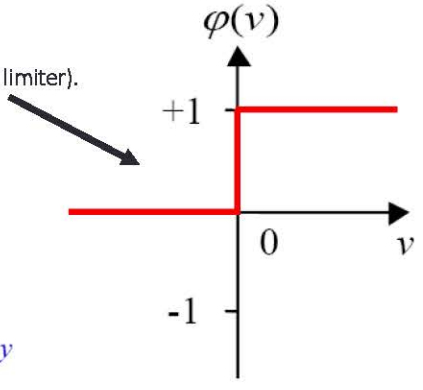
(a) The neuron is purely linear.
(b) The neuron is represented by a McCulloch-Pitts model (hard limiter).



Story 2 Beginning

EXAMPLE: A neuron *j* receives inputs from four other neurons whose activity levels are 10, -20, 4 and -2. The respective synaptic weights of neuron *j* are 0.8, 0.2, -1.0, and -0.9. Calculate the output of neuron *j* for the following two situations

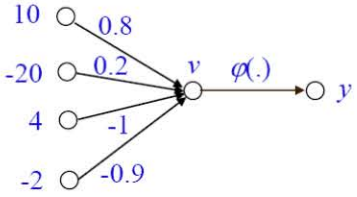
(a) The neuron is purely linear.
(b) The neuron is represented by a McCulloch-Pitts model (hard limiter).



Assume that the bias applied to the neuron is zero.
Solution: The induced local field of neuron *j* is:

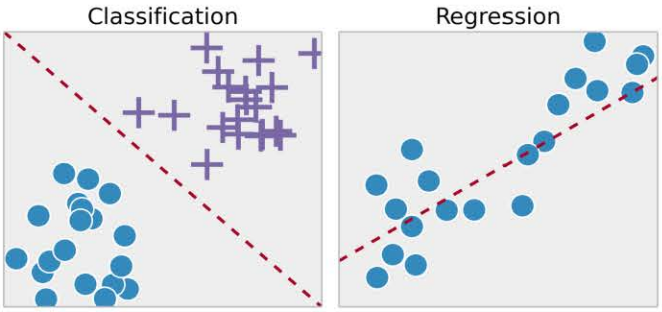
$$v_j = \sum_{i=0}^{4} w_{ji} x_i$$

where $x_0 = 1$, $w_{j0} = b_j = 0$ (no bias)
$x_1 = 10$, $w_{j1} = 0.8$
$x_2 = -20$, $w_{j2} = 0.2$
$x_3 = 4$, $w_{j3} = -1.0$
$x_4 = -2$, $w_{j4} = -0.9$
Hence, $v_j = 1.8$



Story 2 Beginning

NNs are mainly used for solving two types of problems:
Pattern Recognition (Pattern Classification)
Regression (Data Fitting, Function Approximation)



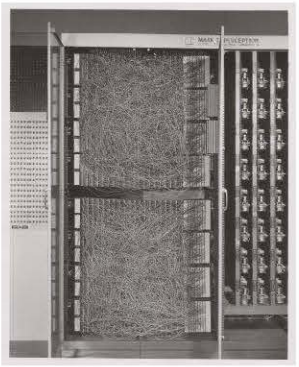
4 AI Story 3: Popular

Story 3 Popular

Perceptron—single layer neural networks
Frank Rosenblatt (1928-1971), at Cornell University in 1958.

Perceptron was the first computer that could learn new skills by trial and error. Rosenblatt was a colorful character at Cornell in the early 1960s. A handsome bachelor, he drove a classic MGA sports car and was often seen with his cat named Tobermory. He enjoyed mixing with undergraduates, and for several years taught an interdisciplinary undergraduate honors course entitled "Theory of Brain Mechanisms" that drew students equally from Cornell's Engineering and Liberal Arts colleges.

By the study of neural networks such as the Perceptron, Rosenblatt hoped that "the fundamental laws of organization which are common to all information handling systems, machines and men included, may eventually be understood."

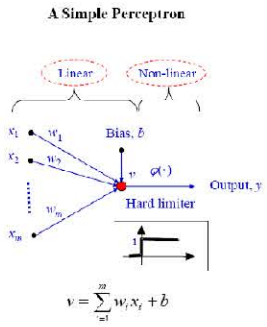


First hardware perceptron in Cornell

Everyone talks about AI at that time!

Story 3 Popular

A Perceptron is a classification hyper-plane



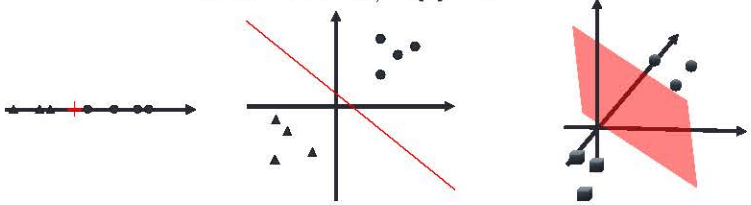
$$v = \sum_{i=1}^{m} w_i x_i + b$$

$$v(n) = \sum_{i=1}^{m} w_i(n)x_i(n) + b(n) = \sum_{i=0}^{m} w_i(n)x_i(n) = w^T(n)x(n)$$

if $v(n) = w^T(n)x(n) > 0$, $y(n) = 1$

if $v(n) = w^T(n)x(n) < 0$, $y(n) = 0$

Decision boundary: $v(n) = 0$



A point on a line | A line in a plane | A plane in the 3-dimensional space

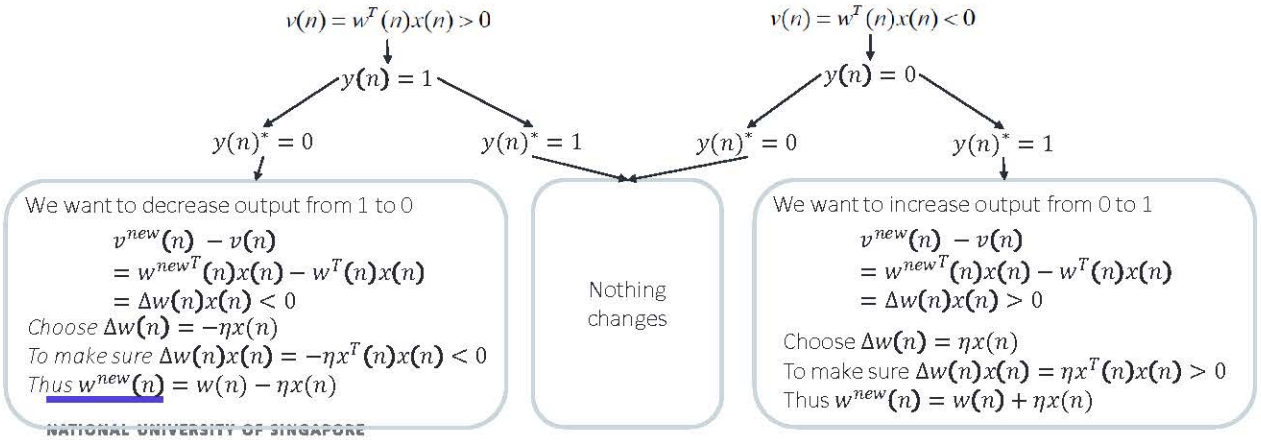
A hyper-plane in the m-dimensional space (input vector space)

Story 3 Popular

Perceptron classification learn

Feed a pattern x to the perceptron with weight vector w, it will produce a binary output y (1 or 0, **i.e. Fire or Not Fire**).

We use labelled and correct data to do **supervised learning**.



Story 3 Popular

Perceptron classification learn

We choose $e = y^* - y$ *or* $d - y$ to replace the sign in learning equation:

$$w^{new}(n) = w(n) \pm \eta x(n) \Rightarrow w^{new}(n) = w(n) + \eta e x(n)$$

Perceptron Learning Algorithm

while *there exist input vectors that are misclassified by* ***w(n)***
Do *Let* ***x(n)*** *be a misclassified input vector;*
Update the weight vector to

$$w(n+1) = w(n) + \eta e(n)x(n)$$

$$e(n) = d(n) - y(n)$$

$$\text{Where } \eta > 0 \text{ and d} = \begin{cases} 1 \text{ if } x \text{ belongs to class } 1 \\ 0 \text{ if } x \text{ belongs to class } 2 \end{cases}$$

Increment n
End while

Can we conclude that choosing a large learning rate would speed up the convergence?

If it is chosen very large and applied to the example x(n), then learning is excellent as far as the present example is concerned, but at the cost of spoiling the learning that has taken place earlier with respect to other examples. Thus, a large value is not necessarily good. Called over-good for one sample.

An extremely small value is chosen , that also leads to slow learning. Some intermediate value is the best. Usually the choice is problem dependent.

Story 3 Popular

Perceptron classification learn

Perceptron Convergence Theorem: (Rosenblatt, 1962)

If *C*1 and *C*2 are linearly separable, then the perceptron training algorithm "converges" in the sense that after a *finite number* of steps, *the synaptic weights* remain unchanged and the perceptron correctly classifies all elements of the training set.

If samples are linearly separable, we assume w_0^T is the optimal weights can classify 2 classes.

Then there is $\alpha = min\{|w_0^T x(i)|\}$

We will only prove the case when w(1)=0, and η=1.

Story 3 Popular

Perceptron classification learn

Perceptron Convergence Theorem: (Rosenblatt, 1962)

If samples are linearly separable, we assume w_0^T is the optimal weights can classify 2 classes.
Then there is $\alpha = min\{|w_0^T x(i)|\}$
We will only prove the case when w(1)=0, and η=1.

$$w(n+1) = w(n) + e(n)x(n) = e(1)x(1) + \cdots + e(n)x(n)$$

Multiply w_0^T

$$w_0^T w(n+1) = e(1)w_0^T x(1) + \cdots + e(n)w_0^T x(n)$$

Check terms on right:

$e(k)w_0^T x(k) = |w_0^T x(k)|$ *or* 0 $(can\ be\ skipped, only\ consider\ about\ error\ samples)$

α must be positive if no samples lie on boundary.

$$w_0^T w(n+1) = \sum_{i=1}^{n} |w_0^T x(i)| \geq n\alpha$$

Using the Cauchy-Schwarz inequality: $\|x\|\,\|y\| \geq |x^T y|$:

$$\|w_0\|\,\|w(n+1)\| \geq |w_0^T w(n+1)| \geq n\alpha$$

$$\|w(n+1)\| \geq \frac{n\alpha}{\|w_0\|}$$

SO we get lower bound

Story 3 Popular

Perceptron classification learn

Perceptron Convergence Theorem: (Rosenblatt, 1962)

We need to figure out how the magnitudes of the weights change with time

$$\| w(n+1) \|^2 = w^T(n+1)w(n+1) = (w(n)+e(n)x(n))^T(w(n)+e(n)x(n))$$
$$= w^T(n)w(n) + 2e(n)w^T(n)x(n) + e^2(n)x^T(n)x(n)$$

Let's check out the middle term

$$e(n)w(n)^T x(n) = (d(n) - y(n))w^T(n)x(n)$$

Story 3 Popular

Perceptron classification learn

Perceptron Convergence Theorem: (Rosenblatt, 1962)

We need to figure out how the magnitudes of the weights change with time

$$\| w(n+1) \|^2 = w^T(n+1)w(n+1) = (w(n)+e(n)x(n))^T(w(n)+e(n)x(n))$$
$$= w^T(n)w(n) + 2e(n)w^T(n)x(n) + e^2(n)x^T(n)x(n)$$

Let's check out the middle term

$$e(n)w(n)^T x(n) = (d(n) - y(n))w^T(n)x(n)$$

$$w^T(n)x(n) > 0 \Longrightarrow y(n) = 1, d(n) = 0 \Longrightarrow e(n)w^T(n)x(n) < 0$$

$$w^T(n)x(n) < 0 \Longrightarrow y(n) = 0, d(n) = 1 \Longrightarrow e(n)w^T(n)x(n) < 0$$

$$\| w(n+1) \|^2 - \| w(n) \|^2 = 2e(n)w^T(n)x(n) + \| x(n) \|^2 \le \| x(n) \|^2$$

$$\|w(2)\|^2 - \|w(1)\|^2 + \cdots + \|w(n+1)\|^2 - \|w(n)\|^2 = \|w(n+1)\|^2 - \|w(1)\|^2 \le \sum_{i=1}^{n} \|x(i)\|^2$$

Let $\beta = max\{\|x(i)\|^2\}$, so we get $\|w(n+1)\|^2 \le n\beta$ We get the upper bound

Story 3 Popular

Perceptron classification learn

Perceptron Convergence Theorem: (Rosenblatt, 1962)

$$\|w(n+1)\|^2 \le n\beta \qquad \|w(n+1)\| \ge \frac{n\alpha}{\|w_0\|}$$

$$\frac{n^2\alpha^2}{\|w_0\|^2} \le \|w(n+1)\|^2 \le n\beta$$

FASTER

$$n \to \infty$$

$$\frac{n^2\alpha^2}{\|w_0\|^2} = n\beta$$

So we know the convergence number of n:

$$n_{max} = \frac{\alpha^2}{\|w_0\|^2\beta}$$

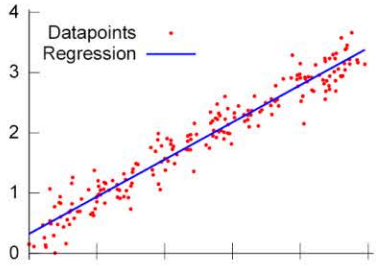
Story 3 Popular

Perceptron regression learn

Regression Problem

Things goes to continuous from discrete

How the error signal $e(i)$ is used to adjust the synaptic weights in the model for the unknown system is determined mainly by the ***cost function*** used.
It can be formulated as an ***optimization problem***,



What is the most common cost function to evaluate how good the model is?

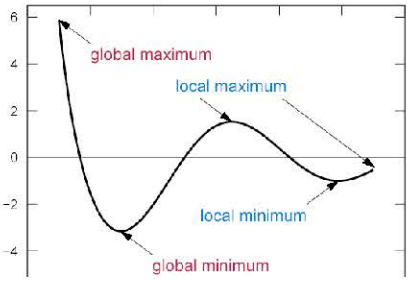
Summation of squares of errors: $E(w) = \sum_{i=1}^{n} e(i)^2 = \sum_{i=1}^{n} (d(i) - y(i))^2$

$E(w) = \sum_{i=1}^{n} |e(i)|$ Not smooth to take differentiate

Story 3 Popular

Perceptron regression learn

Regression Problem



Where to find the minimal or maximal points?

$\frac{df(x)}{dx} = 0$ and the boundary

For multiple parameters: $\nabla\left(E(\boldsymbol{w}^*)\right) = 0$ $\nabla = \left[\frac{\partial}{\partial w_0}, \frac{\partial}{\partial w_1}, \ldots\ldots, \frac{\partial}{\partial w_m}\right]^T$

Usually it is not easy to solve $\nabla\left(E(\boldsymbol{w}^*)\right) = 0$

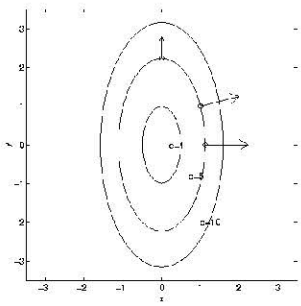
Iterative descent algorithm: Starting with an initial guess denoted by w(0):

$w(n+1) = w(n) + \Delta w(n)$

How to define to make sure $E(w(n+1)) < E(w(n))$

Story 3 Popular

Perceptron regression learn : Method of Steepest Descent (Gradient Descent)



Gradient is the direction along which the function value rises most quickly
If you want the cost E(w) to decrease, u should let w move along opposite the direction of the gradient.

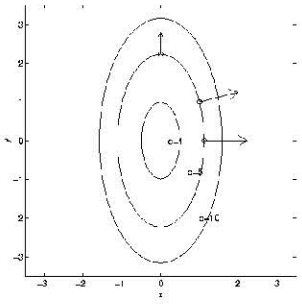
Let $\boldsymbol{g(n)} = \nabla E(\boldsymbol{w(n)})$, steepest descent algorithm is formally described by

$$\Delta w(n+1) = w(n) - \eta g(n)$$

where η is a positive constant called the step size or learning-rate parameter, $\boldsymbol{g}(n)$ is the gradient vector evaluated at the point $\boldsymbol{w}(n)$.

Story 3 Popular

Perceptron regression learn : Method of Steepest Descent (Gradient Descent)



How to guarantee $E(w(n+1)) < E(w(n))$ **in every iteration**

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots$$

$$E(w(n+1)) = E(w(n) + \Delta w(n)) \approx E(w(n)) + \frac{\partial E}{\partial w}\Delta w(n)$$

$$E(w(n+1)) \approx E(w(n)) + g^T(n)\Delta w(n) \qquad \text{holds for small } \eta$$
$$\approx E(w(n)) - \eta g^T(n) g(n)$$
$$\approx E(w(n)) - \eta \|g(n)\|^2 \qquad \text{DECREASING!}$$

Story 3 Popular

Perceptron regression learn : Least-Mean-Square Algorithm

Proposed by Widrow and Hoff at Stanford University in 1960

Based on the instantaneous cost function at step n

$$E(w) = \frac{1}{2}e^2(n)$$

where e(n) is the error signal measured at step n. $e(n) = d(n) - x^T(n)w(n)$

$$\frac{\partial E}{\partial e} = e(n) \qquad \frac{\partial e(n)}{\partial w(n)} = -x^T(n) \longrightarrow \frac{\partial E(w)}{\partial w(n)} = -e(n)x^T(n)$$

The gradient $g(n) = (\frac{\partial E(w)}{\partial w(n)})^T = -e(n)x(n)$

Applying steepest descent method for iteration, we have

$$w(n+1) = w(n) - \eta g(n) = w(n) + \eta e(n)x(n)$$

5

AI Story 4: Winter

Story 4 Winter
The Tragical ending of the Perceptron

Frank was overly optimistic about the power of the perceptron and predicted that "perceptron may eventually be able to learn, make decisions, and translate languages."

In 1969, Marvin Minskyand Seymour Papert showed that it was impossible for perceptron to learn an XOR function. They conjectured (incorrectly)that a similar result would hold for a perceptron with three or more layers.

Frank (1946) and Marvin(1945) were schoolmates at the Bronx High School of Science in New York City

Rosenblatt died tragically in a boating accident in 1969, shortly after Minsky's book was published.

In 2004 the IEEE established the Frank Rosenblatt Award.

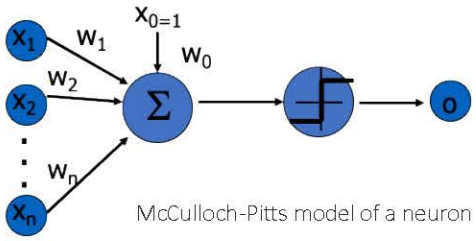
And then, AI goes into long winter.

Story 4 Winter

41

limitation of perceptron which caused the winter of AI

Perceptron

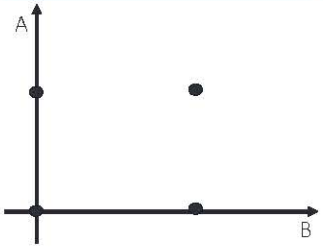


McCulloch-Pitts model of a neuron

$$net = \sum_{i=0}^{n} w_i \cdot x_i$$

$$o = \mathrm{sgn}(net) = \begin{cases} 1 & if \quad net \geq 0 \\ -1 & if \quad net < 0 \end{cases}$$

		Linear separable				Linear inseparable	
INPUTS		OUTPUTS					
A	B	AND	NAND	OR	NOR	EXOR	EXNOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

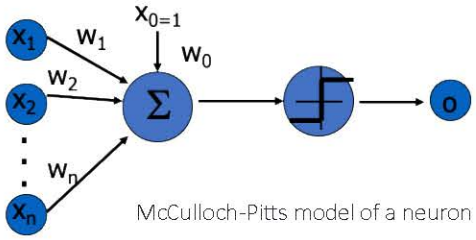


Story 4 Winter

42

limitation of perceptron which caused the winter of AI

Perceptron

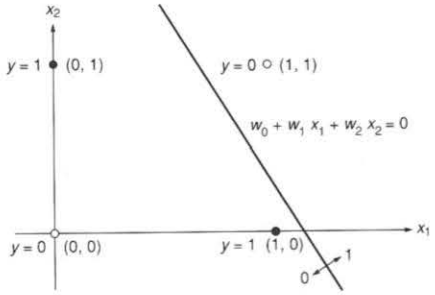


McCulloch-Pitts model of a neuron

$$net = \sum_{i=0}^{n} w_i \cdot x_i$$

$$o = \mathrm{sgn}(net) = \begin{cases} 1 & if \quad net \geq 0 \\ -1 & if \quad net < 0 \end{cases}$$

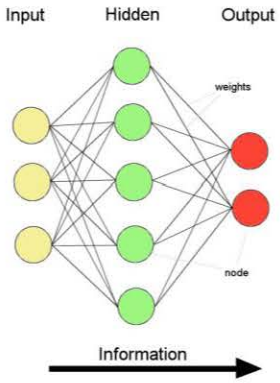
BUT in XOR problem



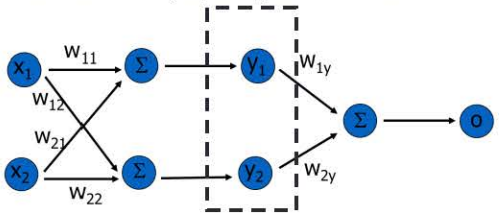
By Minsky and Papert in mid 1960

It will not converge!

Story 4 Winter
Neural Networks (Hidden layer)

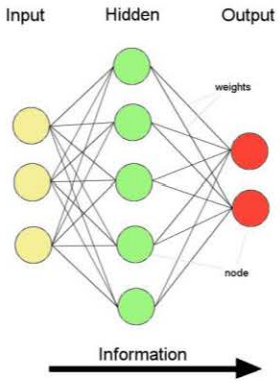


With hidden layer, XOR problem solved!

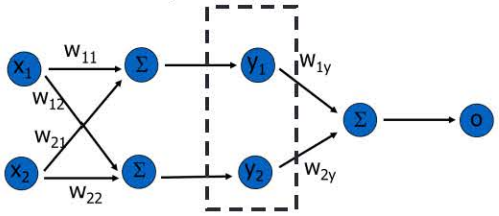


Can you design proper weights?

Story 4 Winter
Neural Networks (Hidden layer)



With hidden layer, XOR problem solved!



We design:
$w_{11} = 1$
$w_{12} = -1$
$w_{21} = -1$
$w_{22} = 1$
$w_{y1} = 0.5$
$w_{y2} = 0.5$

When $x_1 = 1, x_2 = 1$
$y_1 = \text{sgn}(w_{11} x_1 + w_{21} x_2) = \text{sgn}(0) = 1$
$y_2 = \text{sgn}(w_{12} x_1 + w_{22} x_2) = \text{sgn}(0) = 1$
$o = (w_{1y} y_1 + w_{2y} y_2) = 1$

When $x_1 = 0, x_2 = 1$
$y_1 = \text{sgn}(w_{11} x_1 + w_{21} x_2) = \text{sgn}(-1) = 1$
$y_2 = \text{sgn}(w_{12} x_1 + w_{22} x_2) = \text{sgn}(-1) = -1$
$o = (w_{1y} y_1 + w_{2y} y_2) = 0$

6 AI Story 5: New spring

Story 5 New spring

New beginning of Artificial Neural Network

Multilayer Perceptron (MLP) and Back Propagation Algorithm

David Rumelhart and the PDP (Parallel Distributed Processing) group, 1986

David Rumelhart
1942 – 2011

He obtained his B.A. in psychology and mathematics in 1963 at the University of South Dakota. He received his Ph. D. in mathematical psychology at Stanford University in 1967. From 1967 to 1987 he served on the faculty of the Department of Psychology at the University of California, San Diego.

The PDP group was led by David Rumelhart and Jay McClelland at UCSD. They became dissatisfied with symbol-processing machines, and embarked on a more ambitious "connectionist" program.

The 1986 PDP book was a big success. The book was read eagerly not only by brain theorists and psychologists but by mathematicians, physicists, engineers and even by people working in Artificial Intelligence.

In 1987, Rumelhart moved to Stanford University, serving as Professor there until 1998.

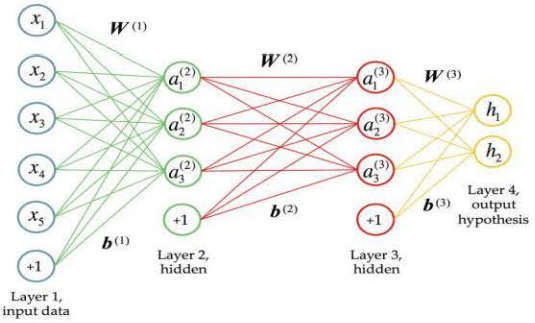
Francis Crick was also a member of the PDP group. He joked later, "Almost my only contribution to their efforts was to insist that they stop using the word *neurons* for the units of their networks."

Story 5 New spring

Multilayer perceptrons (MLPs)

- Generalization of the single-layer perceptron Consists of An input layer
- One or more hidden layers of computation nodes
- An output layer of computation nodes

Architectural graph of a multilayer perceptron with two hidden layers:



Story 5 New spring

Universal Approximation Theorem:

Question: Can Multi-layer Perceptrons approximate any functions?

Universal Approximation Theorem:

Let $\varphi(\cdot)$ be a non-constant, bounded, and monotone-increasing continuous function. Let I_{m_0} denote the m_o -dimensional unit hypercube $[0, 1]^{m_0}$. Then, given any continuous function f on I_{m_0} and $\varepsilon > 0$, there exist an integer m_1 and sets of real constant α_i, b_i and w_{ij}, where $i = 1, \ldots, m_1$ and $j = 1, \ldots, m_0$ such that we may define,

$$F(x_1,\ldots,x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i\right)$$

as an approximate realization of the function $f(\cdot)$; that is,

$$\left|F(x_1,\ldots,x_{m_0}) - f(x_1,\ldots,x_{m_0})\right| < \varepsilon$$

for all $x_1, x_2, \ldots, x_{m_0}$ that lie in the input space.

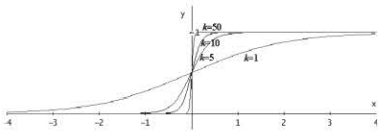
Story 5 New spring

Activation function

MLP generally adopts a smooth nonlinear activation function, such as the following logistic function:

$$y_j = \frac{1}{1+\exp(-v_j)}$$

Where v_j is the induced local field (weighted sum of all synaptic inputs plus the bias) of neuron j , y_j is the output of the neuron.



Name	Plot	Equation	Range
Binary Step X		$f(x) = \begin{cases} 0 & ,x<0 \\ 1 & ,x\geq 0 \end{cases}$	$\{0,1\}$
Logistic		$f(x) = \frac{1}{1+e^{-x}}$	$(0,1)$
TanH		$f(x) = \tanh(x)$	$(-1,1)$
Arctan		$f(x) = \tan^{-1}(x)$	$\left(-\frac{\pi}{2},\frac{\pi}{2}\right)$
Relu		$f(x) = \begin{cases} 0 & ,x<0 \\ x & ,x\geq 0 \end{cases}$	$[0,\infty)$

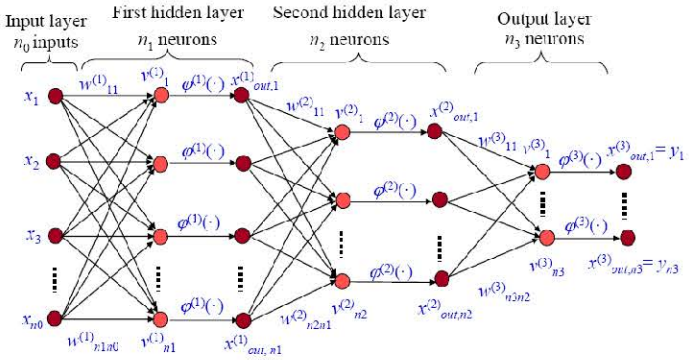
Activation Functions

What would happen if all the neurons are linear neurons? Would it behave differently from single layer perceptrons?

Story 5 New spring

Back-Propagation Algorithm

Consider a multilayer perceptron neural network having three layers of neurons (one output layer and two hidden layers).



Story 5 New spring

Back-Propagation Algorithm

Let ***d(n)*** denote the desired network output, and the LMS error is then

$$E(n) = \frac{1}{2}\sum_{j=1}^{n_3} e_j(n)^2 = \frac{1}{2}\sum_{j=1}^{n_3}(d_j(n) - x_{out,j}^{(3)}(n))^2$$

Gradient Descend $\quad \Delta w(n) = -\eta g(n)$, $\quad g^T(n) = \frac{\partial E(n)}{\partial w(n)}$ $\quad \Longrightarrow \quad \Delta w_{ji}^{(s)}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}^{(s)}(n)}$

where $s = 1, 2, 3$ designates the appropriate network layer, eta > 0 is the corresponding learning rate parameter

Story 5 New spring

Back-Propagation Algorithm

For Output Layer *(neuron j for output layer):*

We get $\frac{\partial E(n)}{\partial x_{out,j}^{(3)}(n)} = (d_j(n) - x_{out,j}^{(3)}(n)) \bullet (-1) = -e_j(n)$ $\qquad \frac{\partial x_{out,j}^{(3)}(n)}{\partial v_j^{(3)}(n)} = \varphi^{(3)\prime}(v_j^{(3)}(n))$

We also have $v_j^{(3)}(n) = \sum_{i=1}^{n_2} w_{ji}^{(3)}(n) x_{out,i}^{(2)}(n)$ $\qquad \frac{\partial v_j^{(3)}(n)}{\partial w_{ji}^{(3)}(n)} = x_{out,i}^{(2)}(n)$

To calculate $\frac{\partial E(n)}{\partial w_{ji}^{(3)}(n)}$, we need to use chain rule $g(f(x))' = g'(f(x))f'(x)$.

$$\frac{\partial E(n)}{\partial w_{ji}^{(3)}(n)} = \frac{\partial E(n)}{\partial x_{out,j}^{(3)}(n)} \frac{\partial x_{out,j}^{(3)}(n)}{\partial v_j^{(3)}(n)} \frac{\partial v_j^{(3)}(n)}{\partial w_{ji}^{(3)}(n)}$$

$$\Rightarrow \quad \frac{\partial E(n)}{\partial w_{ji}^{(3)}(n)} = -e_j(n)\varphi_j^{(3)\prime}(v_j^{(3)}(n)) x_{out,i}^{(2)}(n)$$

If we define $\delta_j^{(3)}(n) = e_j(n)\varphi_j^{(3)\prime}(v_j^{(3)}(n))$

$$\Rightarrow \quad \frac{\partial E(n)}{\partial w_{ji}^{(3)}(n)} = -\delta_j^{(3)}(n) x_{out,i}^{(2)}(n)$$

Story 5 New spring

Back-Propagation Algorithm

In the second hidden Layer:

no directly relation

$$\frac{\partial E(n)}{\partial w_{ji}^{(2)}(n)} = \frac{\partial E(n)}{\partial v_j^{(2)}(n)} \frac{\partial v_j^{(2)}(n)}{\partial w_{ji}^{(2)}(n)}$$

$$\frac{\partial E(n)}{\partial v_j^{(2)}(n)} = \frac{\partial E(n)}{\partial x_{out,j}^{(2)}(n)} \frac{\partial x_{out,j}^{(2)}(n)}{\partial v_j^{(2)}(n)}$$

$$\frac{\partial v_j^{(2)}(n)}{\partial w_{ji}^{(2)}(n)} = \frac{\partial (\sum_{k=1}^{n_1} w_{jk}^{(2)}(n) x_{out,k}^{(1)}(n))}{\partial w_{ji}^{(2)}(n)} = x_{out,i}^{(1)}(n)$$

$$\frac{\partial E(n)}{\partial x_{out,j}^{(2)}(n)} = \sum_{k=1}^{n_3} \frac{\partial E(n)}{\partial x_{out,k}^{(3)}} \frac{\partial x_{out,k}^{(3)}}{\partial x_{out,j}^{(2)}(n)} = \sum_{k=1}^{n_3} e_k(n) \varphi^{(3)\prime}(v_k^{(3)}(n)) w_{kj}^{(3)}(n)$$

$$\frac{\partial x_{out,j}^{(2)}(n)}{\partial v_j^{(2)}(n)} = \frac{\partial \varphi^{(2)}(v_j^{(2)}(n))}{\partial v_j^{(2)}(n)} = \varphi^{(2)\prime}(v_j^{(2)}(n))$$

Use chain rule, **the error can influence second layer, even no directly relation**

$$\delta_k^{(3)}(n) = e_k(n) \varphi^{(3)\prime}(v_k^{(3)}(n)) \Longrightarrow \frac{\partial E(n)}{\partial x_{out,j}^{(2)}(n)} = -\sum_{k=1}^{n_3} w_{kj}^{(3)}(n) \delta_k^{(3)}(n)$$

If we define $\delta_j^{(2)}(n) = (\sum_{k=1}^{n_3} w_{kj}^{(3)}(n) \delta_k^{(3)}(n)) \varphi^{(2)\prime}(v_j^{(2)}(n))$ $\Delta w_{ji}^{(2)}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}^{(2)}(n)} = -\eta \frac{\partial E(n)}{\partial v_j^{(2)}(n)} \frac{\partial v_j^{(2)}(n)}{\partial w_{ji}^{(2)}(n)} = \eta \delta_j^{(2)}(n) x_{out,i}^{(1)}(n)$

Story 5 New spring

Back-Propagation Algorithm

Summary,

$$w_{ji}^{(s)}(n+1) - w_{ji}^{(s)}(n) + \eta \delta_j^{(s)}(n) x_{out,i}^{(s-1)}(n)$$

where

$$\delta_j^{(s)}(n) = (d(n) - x_{out,j}^{(s)}(n)) \varphi^{(s)\prime}(v_j^{(s)}(n)) \quad \text{for output layer}$$

or

$$\delta_j^{(s)}(n) = (\sum_{k=1}^{n_{s+1}} \delta_k^{(s+1)}(n) w_{kj}^{(s+1)}(n)) \varphi^{(s)\prime}(v_j^{(s)}(n)) \quad \text{for hidden layer}$$

It can be viewed as the output of the neuron of another network where all the connections are **"backwards"** now!

$$\delta_j^{(3)}(n) = e_j(n) \varphi_j^{(3)\prime}(v_j^{(3)}(n))$$

$$\delta_j^{(2)}(n) = (\sum_{k=1}^{n_3} w_{kj}^{(3)}(n) \delta_k^{(3)}(n)) \varphi^{(2)\prime}(v_j^{(2)}(n))$$

Output Error — Gradient of the activation function

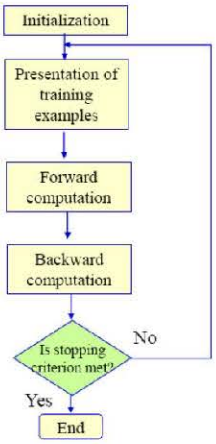
Story 5 New spring

Back-Propagation Algorithm

Back-propagation algorithm - 2 passes of computation:

1. Forward pass: Computation of function signals for each neuron.

2. Backward pass: Starts at the output layer, backwardly compute δ for each neuron from output layer towards the first hidden layer. At each layer, the synaptic weights are changed accordingly to the above delta rule.



Back-Propagation Algorithm – Stopping Criteria

Epoch: A complete presentation of the entire training set during the learning process.

For single layer perceptron, does the LMS converge?

Similarly, BP would not converge.

When to stop training? Any suggestions

Possible criteria:

Absolute rate of change in mean squared error per epoch is sufficiently small.

Synaptic weights and bias level stabilized.

Mean squared error over the entire training set is less than some threshold value.

The total number of epochs reaches a threshold.

7 Design issues

Design issues

Input format

Consider the following regression example: we try to build a map which uses the person's age, gender and height to predict the average weight.

Gender: 1 or -1, 1 or 0
Age: 0--120
Height: 0---2.5m

The neural network would treat all the input variables equally as **simple numbers**!
If the input is [1 30 2], the neural network would recognize that the second input value is much higher than others.

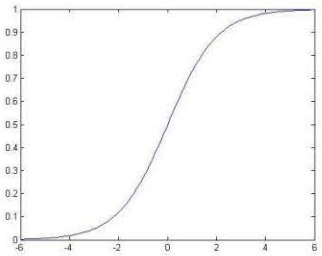
Consider the situation that you want **to normalize** all the inputs to the same range of either [0,1] or [-1, 1].

How to normalize the input variables?
There are many ways:

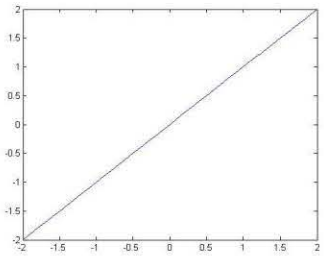
$$\bar{x}_i = \frac{\sum_{n=1}^{N} x_i(n)}{N} \qquad \sigma = \sqrt{\frac{\sum_{n=1}^{N} (x_i(n) - \bar{x}_i)^2}{N}} \qquad x_i'(n) = \frac{(x_i(n) - \bar{x}_i)}{\sigma}$$

Design issues

Output scale



$$\log sig(x) = \frac{1}{1+e^{-x}}$$



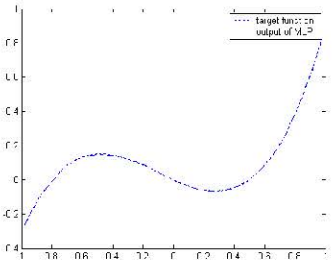
$$purelin(x) = x$$

In the regression problem, the outputs may not necessarily lie in the range of (0, 1) Or (-1, 1). Using linear neuron at the output layer would be more flexible.

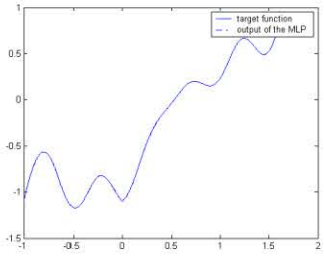
Design issues

Number of neurons

Guideline: Estimate the minimal number of line segments (or hyperplanes in high dimensional cases) that can construct the basic geometrical shape of the target function, and use this number as the first trial for the number of hidden neurons of the three-layered MLP.



1-3-1



1-9-1

Hidden neurons mean the flexibility of a MLP

However, when too many neurons are not necessary, will be easy to go overfitting

Design issues

Number of layers

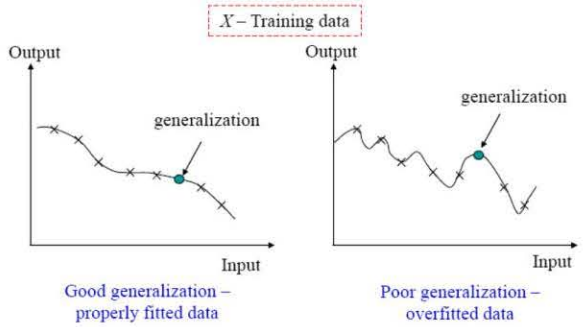
MLPs	Total Parameters
1-9-1	28
1-3-3-1	22

The total number of free parameters can be slightly decreased!

However, adding more layers may make the MLP more prone to **local minima traps** because of its more complicate structure. Also, it is harder to train.

Design issues
Overfitting

A network is said to generalize well when the input-output mapping computed by the network is correct (or near so) for test data never used in creating or training the network.

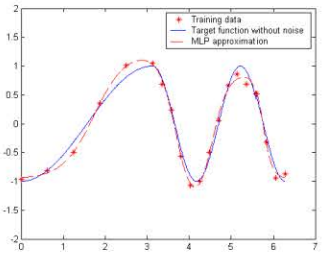


Overfitting:
Fit the training set too well,
But poor in test set.

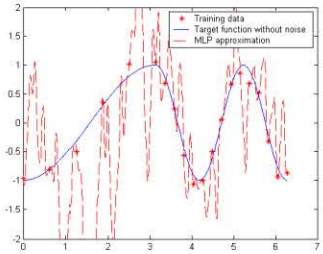
What are the factors that can influence generalization?

- Size and accuracy of training set;
- Architecture of neural network---number of free parameters.

Design issues
Regularization



1-4-1 MLP



1-100-1 MLP

Regularization Methods: $Enew = E + \lambda Ew$

L1-norm regularization: $Ew = |w|$ (more sparse, eliminate useless weights, more robust)

L2-norm regularization: $Ew = \|w\|_2^2$ (easier to calculate)

NUS National University of Singapore

NUS AI SUMMER EXPERIENCE 2019

Experiment:

https://playground.tensorflow.org/

1. How many neurons need for classification problems (No.2, No.3) at least? What are they like(value)?
2. When we use L1 regularization? What happen to number of learned weights?
3. For harder classification problem, how many neurons we need if we add one hidden layer?
4. For harder classification problem, please compare the speed of training(how many steps for a standard loss):
   a. Least number of neurons, one layers
   b. All neurons, one layers
   c. Least number of neurons, two layers
   d. All neurons, two layers
5. For regression problem 1, we set noise to be largest, and training set only 10%, figure out the test loss/train loss value for one neuron and 8 neurons, what about two hidden layers?

Thank you!

Li Xinke

Xinke.li@u.nus.edu