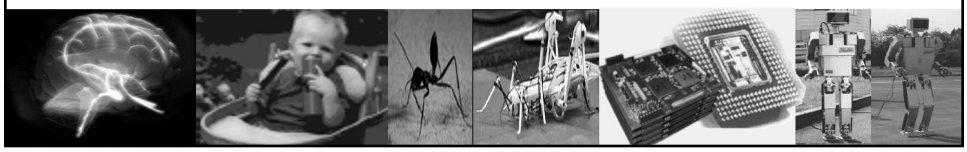
Neural Networks and Deep Neural Networks

Prof. Xiangnan Zhong
Department of Electrical Engineering and Computer Science
Florida Atlantic University
Boca Raton, FL 33431



1

Note

Most of the lecture slides are adopted from the following resources:

[1] T. M. Mitchell, Machine Learning, McGraw Hill, 1997. ISBN: 978-0-07-042807-2
[2] Weifeng Li, Victor Benjamin, Xiao Liu, and Hsinchun Chen, University of Arizona (2015)
[3] Qiang Yang, Introduction to Deep Learning, HK Univ. of S&T

2

2

What are Neural Networks?

- Simple **computational elements** forming a large **network**
  - Emphasis on learning (pattern recognition)
  - Local computation (neurons)
- Definition of NNs is vague
  - Often | but not always | inspired by biological brain

3

Source: http://www.cs.vu.nl/~elena/nn.html

3

History

- Roots of work on NN are in:
- **Neurobiological studies** (more than one century ago):
  - How do nerves behave when stimulated by different magnitudes of electric current? Is there a minimal threshold needed for nerves to be activated? Given that no single nerve cel is long enough, how do different nerve cells communicate among each other?
- **Psychological studies**:
  - How do animals learn, forget, recognize and perform other types of tasks?
- **Psycho-physical** experiments helped to understand how individual neurons and groups of neurons work.
- **McCulloch and Pitts** introduced the first mathematical model of single neuron, widely applied in subsequent work.

4

Source: http://www.cs.vu.nl/~elena/nn.html

4

History

Prehistory:

- Golgi and Ramony Cajal study the nerve system and **discover neurons** (end of 19th century)

History (brief):

- McCulloch and Pitts (1943): the first artificial neural network with binary neurons
- Hebb (1949): learning = neurons that are together wire together
- Minsky (1954): neural networks for reinforcement learning
- Taylor (1956): associative memory
- Rosenblatt (1958): perceptron, a single neuron for supervised learning

5

Source: http://www.cs.vu.nl/~elena/nn.html

5

History

- Widrow and Hoff (1960): Adaline (Adaptive Linear Neuron or later Adaptive Linear Element)
- Minsky and Papert (1969): limitations of single-layer perceptrons (and they erroneously claimed that the limitations hold for multi-layer perceptrons)

Stagnation in the 70's:

- Individual researchers continue laying foundations
- von der Marlsburg (1973): competitive learning and self-organization

Big neural-nets boom in the 80's

- Grossberg: adaptive resonance theory (ART)
- Hopfield: Hopfield network
- Kohonen: self-organising map (SOM)

6

Source: http://www.cs.vu.nl/~elena/nn.html

6

History

- Oja: neural principal component analysis (PCA)
- Ackley, Hinton and Sejnowski: Boltzmann machine
- Rumelhart, Hinton and Williams: backpropagation

Diversification during the 90's:

- Machine learning: mathematical rigor, Bayesian methods, information theory, support vector machines, ...
- Computational neurosciences: workings of most subsystems of the brain are understood at some level; research ranges from low-level compartmental models of individual neurons to large-scale brain models

7

Source: http://www.cs.vu.nl/~elena/nn.html

7

NNs: goal and design

– Knowledge about the learning task is given in the form of a set of examples (dataset) called training examples.

– A NN is specified by:

- an architecture: a set of neurons and links connecting neurons. Each link has a weight,
- a neuron model: the information processing unit of the NN,
- a learning algorithm: used for training the NN by modifying the weights in order to solve the particular learning task correctly on the training examples.

The aim is to obtain a NN that generalizes well, that is, that behaves correctly on new examples of the learning task.

8

Source: http://www.cs.vu.nl/~elena/nn.html

8

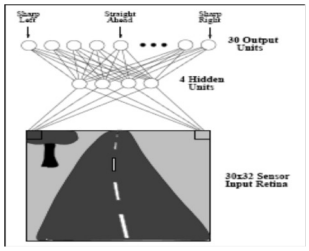
Applications of NNs

- **classification**
  - in marketing: consumer spending pattern classification
  - In defence: radar and sonar image classification
  - In agriculture & fishing: fruit and catch grading
  - In medicine: ultrasound and electrocardiogram image classification, EEGs, medical diagnosis
- **recognition and identification**
  - In general computing and telecommunications: speech, vision and handwriting recognition
  - In finance: signature verification and bank note verification
- **assessment**
  - In engineering: product inspection monitoring and control
  - In defence: target tracking
  - In security: motion detection, surveillance image analysis and fingerprint matching
- **forecasting and prediction**
  - In finance: foreign exchange rate and stock market forecasting
  - In agriculture: crop yield forecasting
  - In marketing: sales forecasting
  - In meteorology: weather prediction

Source: Dr. George Papadourakis, Technological Educational Institute Of Crete, Neural Networks Laboratory

9

NN applications….



ALVINN [Pomerleau' 93] drives 70 mph in highway

10

Who is concerned with NNs?

- Computer scientists want to find out about the properties of non-symbolic information processing with neural nets and about learning systems in general.
- Statisticians use neural nets as flexible, nonlinear regression and classification models.
- Engineers of many kinds exploit the capabilities of neural networks in many areas, such as signal processing and automatic control.
- Cognitive scientists view neural networks as a possible apparatus to describe models of thinking and consciousness (High-level brain function).
- Neuro-physiologists use neural networks to describe and explore medium-level brain function (e.g. memory, sensory system, motorics).
- Physicists use neural networks to model phenomena in statistical mechanics and for a lot of other tasks.
- Biologists use Neural Networks to interpret nucleotide sequences.
- Philosophers and some other people may also be interested in Neural Networks for various reasons

11

Source: Dr. George Papadourakis, Technological Educational Institute Of Crete, Neural Networks Laboratory

11

Neural Networks

- Analogy to biological neural systems, the most robust learning systems we know.
- Attempt to understand natural biological systems through computational modeling.
- Massive parallelism allows for computational efficiency.
- Help understand "distributed" nature of neural representations (rather than "localist" representation) that allow robustness and graceful degradation.
- Intelligent behavior as an "emergent" property of large number of simple units rather than from explicitly encoded symbolic rules and algorithms.

12

Source: Raymond J. Mooney, University of Texas at Austin, CS 391L: Machine Learning Neural Networks

12

Neural Speed Constraints

- Neurons have a "switching time" on the order of a few milliseconds, compared to nanoseconds for current computing hardware.
- However, neural systems can perform complex cognitive tasks (vision, speech understanding) in tenths of a second.
- Only time for performing 100 serial steps in this time frame, compared to orders of magnitude more for current computers.
- Must be exploiting "massive parallelism."
- Human brain has about 10^{11} neurons with an average of 10^4 connections each.

13

Source: Raymond J. Mooney, University of Texas at Austin, CS 391L: Machine Learning Neural Networks

13

Neural Network Learning

- Learning approach based on modeling adaptation in biological neural systems.
- Perceptron: Initial algorithm for learning simple neural networks (single layer) developed in the 1950's.
- Backpropagation: More complex algorithm for learning multi-layer neural networks developed in the 1980's.

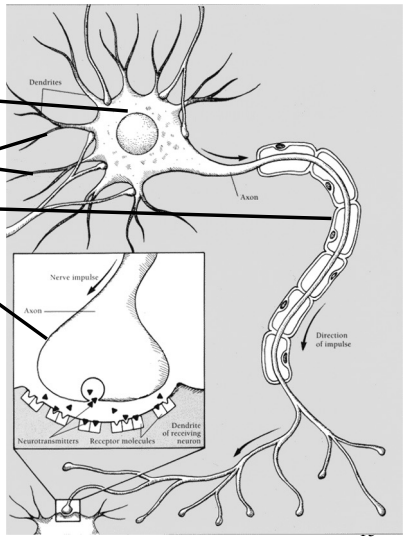
14

Source: Raymond J. Mooney, University of Texas at Austin, CS 391L: Machine Learning Neural Networks

14

Real Neurons

- Cell structures
  - Cell body
  - Dendrites
  - Axon
  - Synaptic terminals

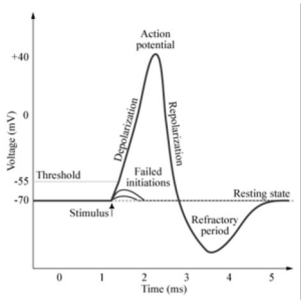


Source: Raymond J. Mooney, University of Texas at Austin, CS 391L: Machine Learning Neural Networks

15

Neural Communication

- Electrical potential across cell membrane exhibits spikes called action potentials.
- Spike originates in cell body, travels down axon, and causes synaptic terminals to release neurotransmitters.
- Chemical diffuses across synapse to dendrites of other neurons.
- Neurotransmitters can be excitatory or inhibitory.
- If net input of neurotransmitters to a neuron from other neurons is excitatory and exceeds some threshold, it fires an action potential.



16

Source: Raymond J. Mooney, University of Texas at Austin, CS 391L: Machine Learning Neural Networks

16

Real Neural Learning

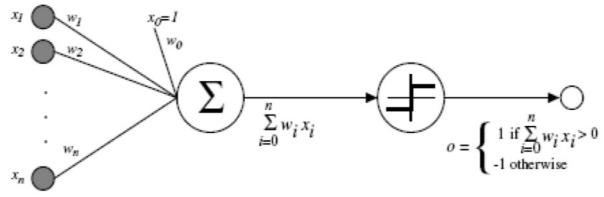
- Synapses change size and strength with experience.
- Hebbian learning: When two connected neurons are firing at the same time, the strength of the synapse between them increases.
- "Neurons that fire together, wire together."

17

Source: Raymond J. Mooney, University of Texas at Austin, CS 391L: Machine Learning Neural Networks

17

Perceptrons



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Sometimes we'll use simpler vector notation:

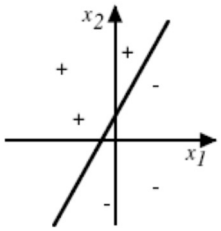
$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

18

18

Decision surface of a perceptron

We can view the perceptron as representing a hyperplane decision surface in the n-dimensional spaces of instances (i.e. points).



(a)

19

19

Decision surface of a perceptron

What weights represent "AND" function?

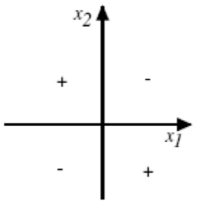
$$g(x_1, x_2) = AND(x_1, x_2)?$$

w0=-0.8, w1=w2=0.5

Perceptrons can represent all of the primitive Boolean functions AND, OR, NAND, NOR

But some functions not representable

- e.g., not linearly separable
- Therefore, we'll want networks of these...



(b)

Not linearly separable!

20

Gradient descent and Delta rule
(least-mean-square (LMS) rule)

To understand, consider simpler *linear unit*, where

$$o = w_0 + w_1 x_1 + \cdots + w_n x_n$$

Let's learn w_i's that minimize the squared error

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

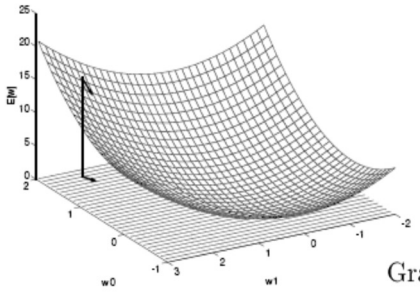
Where D is set of training examples

If the training examples are not linearly separable, the delta rule converges toward a best-fit approximation to the target concept

21

21

Visualizing the gradient descent



Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

22

Derivation of the gradient descent rule

Gradient

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Training rule:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

i.e.,

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\
&= \frac{1}{2} \sum_d 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\
&= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})
\end{aligned}$$

23

23

Perceptron training rule

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta (t - o) x_i$$

Where:

- $t = c(\vec{x})$ is target value
- o is perceptron output
- η is small constant (e.g., .1) called *learning rate*

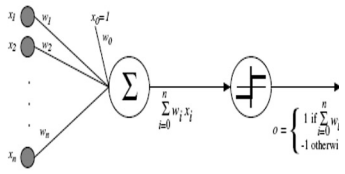
Can prove it will converge

- If training data is linearly separable
- and η sufficiently small

24

24

Gradient Descent algorithm



GRADIENT-DESCENT($training\_examples, \eta$)

Each training example is a pair of the form $\langle \vec{x}, t \rangle$*, where* $\vec{x}$ *is the vector of input values, and* t *is the target output value.* η *is the learning rate (e.g., .05).*

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - Initialize each Δw_i to zero.
  - For each $\langle \vec{x}, t \rangle$ in *training_examples*, Do
    - Input the instance $\vec{x}$ to the unit and compute the output o
    - For each linear unit weight w_i, Do

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$

  - For each linear unit weight w_i, Do

$$w_i \leftarrow w_i + \Delta w_i$$

25

Summary

Perceptron training rule guaranteed to succeed if

- Training examples are linearly separable
- Sufficiently small learning rate η

Linear unit training rule uses gradient descent

- Guaranteed to converge to hypothesis with minimum squared error
- Given sufficiently small learning rate η
- Even when training data contains noise
- Even when training data not separable by H

26

26

Incremental (stochastic gradient) descent

Batch mode Gradient Descent:
Do until satisfied

1. Compute the gradient $\nabla E_D[\vec{w}]$
2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_D[\vec{w}]$

Incremental mode Gradient Descent:
Do until satisfied

- For each training example d in D
  1. Compute the gradient $\nabla E_d[\vec{w}]$
  2. $\vec{w} \leftarrow \vec{w} - \eta \nabla E_d[\vec{w}]$

$$E_D[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

$$E_d[\vec{w}] \equiv \frac{1}{2}(t_d - o_d)^2$$

Incremental Gradient Descent can approximate *Batch Gradient Descent* arbitrarily closely if η made small enough

27

27

Key differences between standard gradient descent and stochastic gradient descent

* In standard gradient descent, the error is summed over all examples before updating weights, whereas in stochastic gradient descent weights are updated upon examining each training example;
* Summing over multiple examples in standard gradient descent requires more computation per weight update step.
* In case where there are multiple local minima, stochastic gradient descent can sometimes avoid falling into these local minima.

Both methods are commonly used in practice

28

28

perceptron training rule and delta rule

perceptron training rule:

$$\Delta w_i = \eta(t - o)x_i$$

$$o(\vec{x}) = \begin{cases} 1 \text{ if } \vec{w} \cdot \vec{x} > 0 \\ -1 \text{ otherwise.} \end{cases}$$

Delta rule:

$$\Delta w_i = \eta(t - o)x_i$$

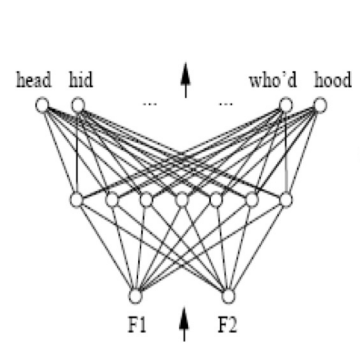
$$o = w_0 + w_1 x_1 + \cdots + w_n x_n$$

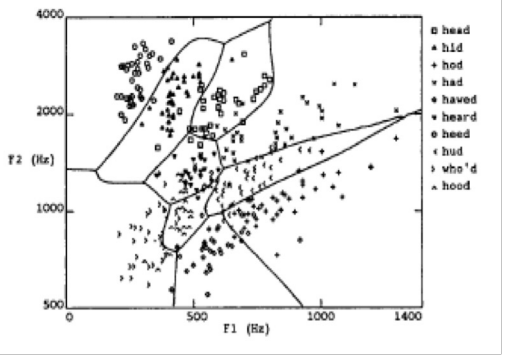
The definition of output ***o*** is different!
Perceptron rule updates weights based on the error in the ***thresholded*** perceptron output, whereas delta rule updates weights based on the error in the ***unthresholded linear combination of inputs***

29

29

Is linear decision surface enough?





Multilayer network to represent highly nonlinear decision surface

30

30