Lecture 2: Neuron models and basic learning rules

Contents of this lecture
- After this lecture, you should know
  - How a neuron works?
  - Some basic neuron models.
  - Basic steps for using a neural network.
  - General learning rule for one neuron.
  - Learning of discrete neuron.
  - Learning of continuous neuron.
  - Learning of single layer NNs with discrete neurons.
  - Learning of single layer NNs with continuous neurons.

How “large” is a human brain?
- A neuron is the basic element in a biological brain.
- There are approximately 100,000,000,000 neurons in a human brain.
- One neuron is connectedly with approximately 10,000 other neurons.
- The human brain is very large and very complex system.
- Although each neuron is slow, un-reliable, and non-intelligent, the whole brain can make decisions very quickly, in a relatively reliable and intelligent way.

What is a bio-neuron?
- A B-neuron contains
  - a cell body for signal processing,
  - many dendrites to receive signals,
  - an axon for outputting the result, and
  - a synapse between the axon and each dendrite.
A neuron works as follows

- Signals (impulses) come into the dendrites through the synapses.
- All signals from all dendrites are summed up in the cell body.
- When the sum is larger than a threshold, the neuron fires, and sends out an impulse signal to other neurons through the axon.

The McCulloch-Pitts neuron model

- Proposed by McCulloch and Pitts in 1943.
- A processor (system) with multiple input and a single output.
- Effective input: weighted sum of all inputs.
- Bias or threshold: if the effective input is larger than the bias, the neuron outputs a one, otherwise, it outputs a zero.

Some terminologies

- The parameters used to scale the inputs are called the weights.
- The effective input is the weighted sum of the inputs.
- The parameter to measure the switching level is the threshold or bias.
- The function for producing the final output is called the activation function, which is the step function in the McCulloch-Pitts model.

Generalization of the neuron model

- In general, there are many different kinds of activation functions.
- The step function used in the McCulloch-Pitts model is simply one of them.
- Because the activation function takes only two values, this model is called discrete neuron.
- To make the neuron learnable, some kind of continuous function is often used as the activation function. This kind of neurons are called continuous neurons.
- Typical functions used in an artificial neuron are sigmoid functions, radial basis function, sinusoidal functions, etc.
**Activation function of continuous neuron: sigmoid function**

\[ f(u) = \frac{2}{1 + \exp(-\lambda u)} - 1 \]

\[ f(u) = \frac{1}{1 + \exp(-\lambda u)} \]

(Fig. 2.5 in the textbook written by Prof. Zurada)

**A neuron model with augmented input**

\[ o = f(\sum_{i=1}^{n} w_i x_i) \]

A dummy input is added so that the effective input is calculated simply using inner product.

**Single layer neural network and multi-layer neural network**

**Basic steps for using a neural network**

- **Learning**: to store the information into the network.
  - Supervised and unsupervised learning.
  - On-line learning and off-line learning.
- **Recall**: to retrieve information stored in the network.
  - Auto-association and hetero-association.
  - Classification and/or recognition.
### Basic Diagram of Learning

**Neural Network**

- **Training Data Set**

**Teacher signal available:** supervised learning
**Teacher signal un-available:** un-supervised learning

---

### Basic diagram of recall

- **Neural Network**

**Auto-association:** The output is the same pattern as the input.
**Hetero-association:** The output is a different representation of the input pattern.

---

### General learning rule for one neuron

\[ W^{k+1} = W^k + crx \]

- \( c \) is a learning constant.
- \( r \) is the learning signal, which is a function of
  - \( W \): the current weight vector
  - \( x \): the input vector

---

### Perceptron learning rule

\[ W^{k+1} = W^k + crx \]

- \( c \): learning constant in [0,1]
- \( r = d - f(u) \)
- \( d \): given teacher signal
- \( u = \langle W^k, x \rangle \)
- \( f(u) = \begin{cases} 1 & \text{if } u > 0 \\ -1 & \text{if } u < 0 \end{cases} \)
- \( W^0 \): Given at random

**This is a supervised learning rule because teacher signals are required**
**Delta learning rule**

\[ W^{k+1} = W^k + crx \]

\( c \): learning constant in [0,1]
\( r = [d - f(u)]f(u) \)
\( d \): given teacher signal
\( u = \langle W^k, x \rangle \)
\( f(u) \): sigmoid function
\( W^0 \): Given at random

This is also a supervised learning rule because teacher signals are required.

**Program for perceptron learning**

- **Initialization**: Initialize the weights at random
- **While** (Error > desired_error)
  - For (Error = 0, p = 0; p < n_sample; p++)
    - \( o = \text{FindOutput}(p) \)
    - Error += 0.5 * pow(d[p] - o, 2.0)
    - For (i = 0; i < I; i++)
      - \( \text{LearningSignal} = \eta \cdot (d[p] - o) \)
      - w[i] += LearningSignal * x[p][i]
  - \( \text{PrintResult}() \)

**Program for delta rule**

- **Initialization**: Initialize the weights at random
- **While** (Error > desired_error)
  - For (Error = 0, p = 0; p < n_sample; p++)
    - FindOutput(p);
    - Error += 0.5 * pow(d[p] - o, 2.0)
    - For (i = 0; i < I; i++)
      - delta = (d[p] - o) * (1 - o * o) / 2;
      - w[i] += w[i] + c * delta * x[p][i];
  - \( \text{PrintResult}() \)

**Example 1**

- There are four training examples shown in the left figure:
  - (1,1), (-1,-1), and (-1,1)
- The teacher signals are -1, 1, 1, and -1
- That is, we want to divide the data into two groups using a line

\[ w_1 x_1 + w_2 x_2 - w_3 = 0 \]
1. The input is augmented with an extra element fixed to -1.
2. If effective input is larger than or equal to zero, the input belongs to group 1.
3. Otherwise, the input is in group 2.

Results of perceptron learning

The initial weights:
(0.811319 0.102490 0.100490)

The error in the 1st learning cycle is 2.000000
The connection weights of the neurons are
(-0.188681 1.102490 -0.899510)

The error in the 2nd learning cycle is 4.000000
The connection weights of the neurons are
(1.811319 1.102490 -0.899510)

The error in the 3rd learning cycle is 0.000000
The connection weights of the neurons are
(1.811319 1.102490 -0.899510)

Results of delta learning

Error in the 161-th learning cycle=0.010610
Error in the 162-th learning cycle=0.010541
Error in the 163-th learning cycle=0.010472
Error in the 164-th learning cycle=0.010405
Error in the 165-th learning cycle=0.010338
Error in the 166-th learning cycle=0.010273
Error in the 167-th learning cycle=0.010208
Error in the 168-th learning cycle=0.010144
Error in the 169-th learning cycle=0.010081
Error in the 170-th learning cycle=0.010018
Error in the 171-th learning cycle=0.009956

The connection weights of the neurons:
3.165432 3.167550 -3.163318

Single layer neural network for solving multi-class problems

• There are J inputs and K outputs.
• The last input is fixed to -1 (dummy input).
• For a given input vector y
  - The effective input of the k-th neuron is net_k
  - The actual output of the k-th neuron is o_k
  - The desired output of the k-th neuron is d_k
  - The error to be minimized is E.
Learning of single layer network

• The learning of a single layer network can be performed by adopting the perceptron learning rule or the delta learning rule separately to each neuron.
• The only thing to do is to add one more LOOP in the program.

Example 2

• Find a single layer neural network with two discrete neurons.
• One is to realize the AND gate, and another is to realize the OR gate.

Numerical results for the example

The initial condition:
\[ W[0]: 0.248875 \quad 0.165883 \quad 0.093191 \]
\[ W[1]: 0.111389 \quad 0.443656 \quad 0.543946 \]

For the 1-th learning cycle:
The error is 6.000000
\[ W[0]: 1.248875 \quad 1.165883 \quad -0.906809 \]
\[ W[1]: 0.111389 \quad 0.443656 \quad 0.543946 \]

For the 2-th learning cycle:
The error is 0.000000
\[ W[0]: 1.248875 \quad 1.165883 \quad -0.906809 \]
\[ W[1]: 0.111389 \quad 0.443656 \quad 0.543946 \]

Team Project I: Part 1

• Write a computer program to realize the perceptron learning rule and the delta learning rule.
• Train a neuron using your program to realize the AND gate. The input pattern and their teacher signals are given as follows:
  - Data: (0,0,-1); (0,1,-1); (1,0,-1); (1,1,-1)
  - Teacher signals: -1, -1, -1, 1
• Program outputs:
  - Weights of the neuron, and
  - Neuron output for each input pattern.
Remarks

- The program given in the web page is for delta learning rule only. You should extend this program for this homework.
- The learning process is iterative. You should provide the data one by one, and start from the first datum again when all data are used once.
- One learning cycle is called an epoch.
- The total errors for all data is used as the terminating condition.
- From this experiment we can see that a neuron can be used to realize an AND gate.

Team Project I: Part 2

- Extend the program written in the first step to learning of single layer neural networks.
- The program should be able to design
  - Case 1: A single layer neural network with discrete neurons.
  - Case 2: A single layer neural network with continuous neurons.
- Test your program using the following data
  - Inputs: (10,2, -1), (2, -5, -1), (-5, 5, -1).
  - Teacher signals: (1, -1, -1), (-1, 1, -1), and (-1, -1, 1)