CS 4990

Practical Artificial Intelligence

Spring 2020
Instructor Dr. Zhiguang Xu

Project 5
The hebbian learning rule

Due by 03/24/2020 (2:00pm)

---

1. Introduction

In neural networks, learning is achieved mostly (but not exclusively) through changes in the strengths of the connections (synapses) between neurons. Mechanisms of learning include:

- changes in neural parameters (threshold, time constants)
- creation of new synapses
- elimination of synapses
- changes in the synaptic weights or connection strengths

One common way to calculate changes in connection strengths in a neural network is the so called "hebbian learning rule", in which a change in the strength of a connection is a function of the pre – and post – synaptic neural activities. It is called the "hebbian learning rule" after D. Hebb - "*When neuron A repeatedly participates in firing neuron B, the strength of the action of A onto B increases*".

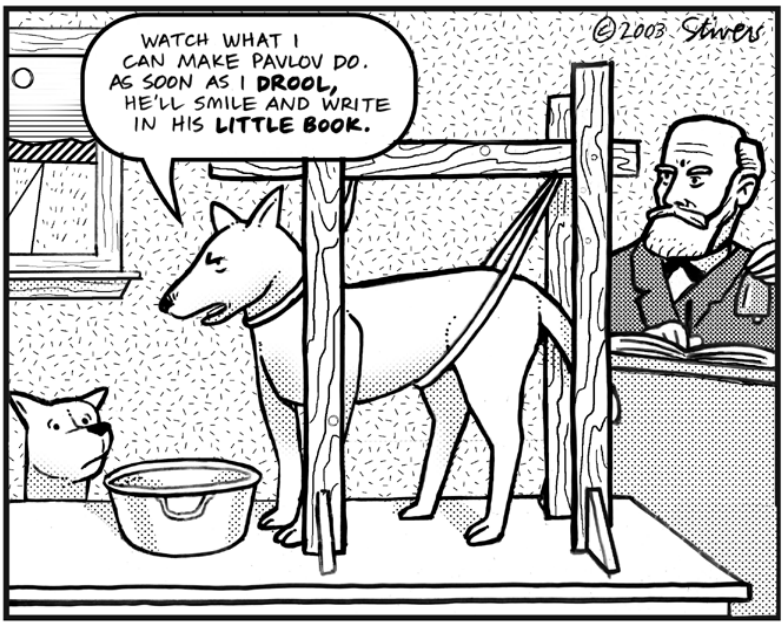
2. Project Subject

In this project, you are asked to implement an Artificial Neural Network with hebbian learning in Java. To inspire your thoughts on what kind of application you use to train and test your ANN (which is all up to you), I am throwing you some ideas in the following sections:

2.1. Pavlov Dog

In 1904 *Ivan Pavlov* was awarded a Nobel Prize in Physiology and Medicine in relation to his researches in the area of digestive processes. He had become interested in the relationship between salivation and the digestive process. Quite apart from finding that saliva was of the first importance as an aid to digestion he also noticed that dogs that had been familiarized with the pre-feeding routines in his research facility began to salivate apparently in association with certain pre-feeding routines being initiated. In order to explicitly validate his observations he began to feed his dogs in association with the ringing of a bell. After a certain time the dogs were shown to salivate profusely in association with the ringing bell where the actual sight or smell of food was not also present. Pavlov regarded this salivation

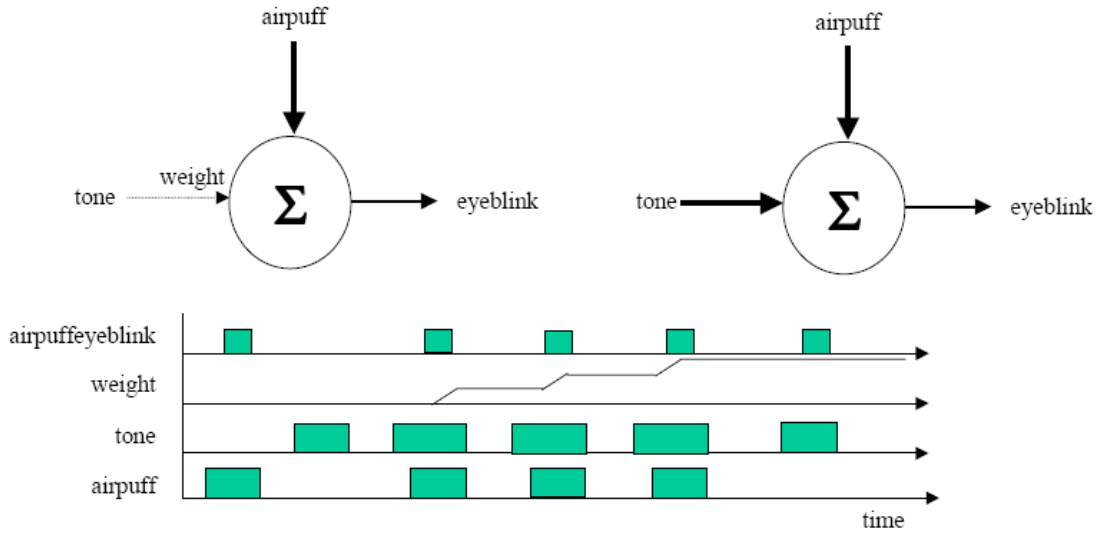
as being a *conditioned reflex* and designated the process by which the dogs had picked up this reflex *classical conditioning*.



2.2. Air puff, tone, and eye blink

Classical conditioning can be modeled with a hebbian synapse. Consider an unconditioned stimulus (an air puff), an unconditioned response (eye blink), a conditioned stimulus (a tone) and a conditioned response (eye blink). Under normal conditions, an animal responds to an air puff with an eye blink. It does not respond to a tone with an eye blink. If the tone is paired with the air puff several times, then the animal acquires an association between the air puff and the tone, and will now respond to the tone alone with an eye blink.

Consider a “black box” neural approach where one neuron receives input from the air puff and from the tone. The neurons output represents the eye blink. The neuron is wired in such a way that at first, the air puff but not the tone activates the neuron and produces an output. If we apply the air puff input and the tone input together several times, then the neuron is active while the tone-input is active, and a hebbian learning rule will reinforce the strength of the connection between the tone and the neuron. This will lead to the fact that after a few trials, the tone alone will be able to activate the neuron.



2.3. Honeybees responding to the odor

Honeybees can be conditioned to respond to an odor stimulus by extending their proboscis (proboscis extension paradigm). In this paradigm the unconditioned stimulus is sucrose, the unconditioned response the proboscis extension and the conditioned stimulus the odor. A specific neuron in the honeybee brain, called VUMmx1 arborizes all over the bee brain. Under normal conditions, it responds to the presentation of sucrose to the proboscis by increased firing rates, but it does not respond to odors. However, after an odor has been presented simultaneously with the sucrose reward for several trials, the neuron responds to the odor alone.

3. Project Execution Environment

This project should be implemented on *Greenfoot* (http://www.greenfoot.org), an award-winning educational programming environment that uses simulations and games to teach object-oriented concepts and techniques in a fun and easily accessible manner. A couple of demos of Greenfoot have already been given in class.

What to submit?

Start this project by newing a Scenario. Then subclass `World` and `Actor` classes, and/or add your own class trees (if needed). Work on them until all the goals pertaining to unsupervised learning are accomplished.

Click Scenario, then "Save As…", name your output folder as `CS4990_P5_YourLastName`, zip it, and submit the zipped file to me. In addition, submit your Final Project Report in the PDF format (as described below).

4. Grading criteria

Your grade will be determined based on the quality of the execution, source code, and a report of your project.

The following grading criteria will be used: (Note that your project may be graded relative to the quality of the other projects in the class)

Functionalities and Correctness	75%	
Coding Style, Documentation and Report	15%	
Progress Report (**Due 3:15pm on 03/12,** through Blazeview Email)		2%
Final Project Report (**Due 03/24**, see below)		13%
Creativity	10%	

You will need to write up a project report documenting your system, which should discuss:

- Motivations for your work
- Objectives you want to achieve and your approaches to achieving them.
- Software components/classes and architecture of your system. This should be a major section of your report. You should explain in detail the functions of each component/class, and how components interact with each other. You should also describe the principle data structures and network messages in your system.
- Instructions for how to run your program. This part of particular importance due to the openness of this project.
- Running results. This is another major section. You are asked to include and explain the running results of your program. Charts, figures, and tables will always help.

- Discussions of your work. Describe its major strengths and weaknesses. Include the features you would like your system to have, but did not do because of resource/time limitations.

This is a project that is designed as an individual project. As always, Plagiarism will automatically result in a grade of 0.