Artificial Intelligence Fundamentals Laboratory

Piotr Łuczak M.Sc.Eng.

Winter 2019

1 Introduction

The aim of the laboratory is to familiarize the students with the basics of artificial intelligence. Two main foci of the laboratory course are artificial neural networks and search algorithms.

2 Laboratory dates and task deadlines

The laboratory will take place in weeks 2 - 15 of the winter semester in alternating weeks per group from 12:15 till 14:00 in room 311, Institute of Applied Computer Science, building A10. Deadlines:

1. 2019-10-09
2. 2019-10-16 (single hour per group) → Data generator and visualizer
3. 2019-10-23 & 2019-10-30 → Perceptron training and decision boundary visualization
4. 2019-11-06 & 2019-11-20 → Different activation functions for a single neuron
5. 2019-11-27 & 2019-12-04 → Dataflow in shallow neural networks and decision boundary visualization
6. 2019-12-11 & 2019-12-18 → Shallow neural network training (backpropagation)
7. 2020-01-08 & 2020-01-15 → Planning using search algorithms
8. 2020-01-22 & 2020-01-29 → Search algorithms ctnd

The final grade will be the average of the grades from each of assignments, a delay in handing in the task solution will decrease the grade for this task by 0.5.

3 Task descriptions

3.1 Data generator and visualizer

Prepare a GUI that enables generation and visualization of data samples from two classes. Samples should be two-dimensional, so that they can be plotted in x-y space. Each class should consist of one or more gaussian modes[1] with their means and variances chosen randomly from some given inteval (e.g. $\mu_x, \mu_y \in [-1..1]$).
The interface should allow for setting a desired number of modes per class and a desired number of samples per mode, as well as visualization of the generated samples on a two-dimensional plot. Class labels, which are either 0 or 1, should be indicated by colors.

[1] `https://en.wikipedia.org/wiki/Normal_distribution`

3.1.1 Grading

Generation of samples with single mode per class + visualization in GUI → 3
Addition of user provided number of samples per mode → 4
Addition of multiple modes → 5

3.2 Single neuron

Implement an artificial neuron. The neuron should take samples generated by a code from the task 1 as its input and predict their class membership at its output. The neuron should be trainable through the formula presented during the lecture:

$$\Delta\vec{w_j} = \eta\varepsilon f'(s)\vec{x_j} = \eta(d-y)f'(\vec{w}^T\vec{x_j})\vec{x_j} \tag{1}$$

where:

- x_j is the j^{th} sample
- $f'(s)$ is the derivative of the activation function evaluated for the j^{th} sample
- $\vec{w}$ are the weights associated with inputs
- η is the learning rate
- d is the expected (true) class label
- y is a class label predicted by the neuron

Implemented neuron should allow for different activation functions for evaluation:

- Heaviside step function (perceptron)

$$H(s) = \begin{cases} 0 & s < 0 \\ 1 & s \geq 0 \end{cases}$$

- sigmoid (logistic function)

$$y = \frac{1}{1+e^{\beta s}}$$

- sin
- tanh
- sign

$$sgn(s) = \begin{cases} -1 & s < 0 \\ 0 & s = 0 \\ 1 & s > 0 \end{cases}$$

- ReLu

$$ReLu(s) = \begin{cases} s & s > 0 \\ 0 & s \leq 0 \end{cases}$$

- leaky ReLu

$$lReLu(s) = \begin{cases} s & s > 0 \\ 0.01s & s \leq 0 \end{cases}$$

At least the Heaviside and logistic functions must be implemented. Training of the neuron should support the heaviside (assume that the derivative is 1) and logistic function. Variable learning rate may be implemented but is not required. The GUI should present the decision boundary by setting two backround colors for two half-planes.

3.2.1 Grading

Evaluation and training of a neuron with Heaviside and logistic activation functions + decision boundary visualization in GUI → 3
Addition of sin and tanh activation functions for evaluation and training OR introducing variable learning rate → 4
Addition of sign, ReLu and leaky ReLu activation functions for evaluation → 5

3.3 Shallow neural network

Implement a shallow (up to 5 layers) fully connected neural network. The network should be based on the neuron implemented in the task #2 and use the data generated by the task #1 as its input and yield the predicted class membership at its output. According to theory, any data classification problem should be solvable using a three layered network, thus at the minimum the implemented solution should support the evaluation and training of such a network. The output should be presented in the form of two values describing the confidence that the network belongs to each class. The network should consist of a two-neuron input layer, two-neuron output layer and at least one hidden layer, the remaining values of width and depth of the network should be configurable. The neurons should use the logistic activation function, though the implementation may be extended by adding more options. The network should be trained using the backpropagation formula:

$$\Delta\vec{w_j^k} = \eta\delta_k f'(s^k)\vec{x}_l = \eta\Sigma_i(\vec{d^i} - \vec{y^i})f'(s^i)W_k^i f'(s^k)\vec{x}_l \tag{2}$$

As was the case in task 2, the GUI should present the decision boundary for the network through colouring the corresponding parts of the plot.

3.3.1 Grading

Evaluation and training (backpropagation) of 3 layered fully connected neural network + decision boundary visualization in GUI → 3
Parametrizing number of layers / neurons per layer → 4
Addition of multiple activation functions OR training the samples in batches → 5

3.4 Search algorithms

Implement a program capable of finding a solution to the the following logistic problem: find the optimal route for the delivery of goods from a supplier to n recipients using a truck of some fixed capacity. The cost is the total distance needed for delivering all requested goods.
Each recipient needs a specific volume of goods. Assume also, that goods cannot be taken from the recipient nor can they be moved back to the supplier.

The demand for goods is specified using the following table:

Recipient name	Volume of goods requested
A	5
B	3
C	7

There exists a direct route between the supplier and all recipients as well as between any two recipient and mutual distances are shown in the table below (S denotes the supplier):

	A	B	C	S
A	-	2	1	11
B	2	-	4	5
C	1	4	-	8
S	11	5	8	-

Assume that the truck can fit four items at a time and starts at the location of the supplier.

3.4.1 Grading

Working solution based on a blind search algorithm → 3
A solution that implements heuristic search → 4
Extension that enables parameter modifications (demand, supply, number of consumers) → 5

4 Implementation requirements

- Task solutions must be implemented in Python[2] version 3
- Use of existing implementations (libraries, code samples available online, etc.) of solutions to given problems is prohibited. Libraries may be used for ancillary code (plotting, matrix calculations, GUI, etc.), for these tasks matplotlib[3] and NumPy[4] are suggested.
- All libraries used along with the exact version numbers must be specified in `requirements.txt`[5]
- All libraries must be installable through pip[6]
- Solutions to tasks 1 -3 should be implemented as a single program with graphical user interface
- Solution to task 4 may be implemented either in GUI or terminal-based form

[2] `https://www.python.org/`
[3] `https://matplotlib.org/`
[4] `https://numpy.org`
[5] `https://pip.pypa.io/en/latest/user_guide/#requirements-files`
[6] `https://packaging.python.org/tutorials/installing-packages/#requirements-files`