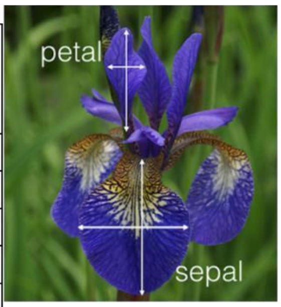
FLOWER CLASSIFICATION USING A SIMPLE NEURAL NETWORK

A flower we see in nature is known to be a lily (or iris) plant. We want to write an algorithm to find out which of the three different species it belongs to. We have data measured from 150 samples of each flower species. For each sample, 4 characteristics (sepal length, sepal width, petal length, petal width) and the class (species) of the flower are provided. Table 3 shows 6 of them. Dataset https://archive.ics.uci.edu/ml/machine-learning_databases/iris/iris/iris.data will be used in the project.

Table 3: Information on 6 plant samples from the flower dataset

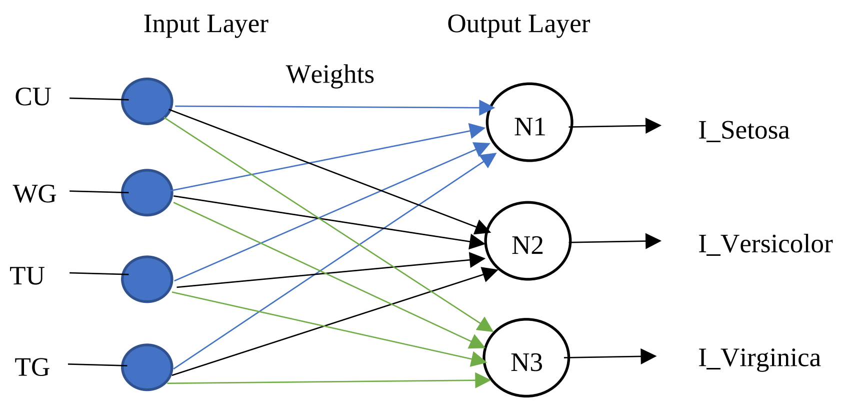
Örnek No	Çanak Yaprak Uzunluğu	Çanak Yaprak Genişliği	Taç Yaprak Uzunluğu	Taç Yaprak Genişliği	Tür
0	5.1	3.5	1.4	0.2	*I. setosa*
1	4.9	3.0	1.4	0.2	*I. setosa*
2	7.0	3.2	4.7	1.4	*I. versicolor*
3	6.4	3.2	4.5	1.5	*I. versicolor*
4	6.3	3.3	6.0	2.5	*I. virginica*
5	5.8	2.7	5.1	1.9	*I. virginica*
…	…	…	…	…	…



Şekil 1: Zambak çiçeğinde çanak (sepal) ve taç (petal) yapraklar.

Create an artificial neural network (class) like the one below and use it to solve the given flower classification problem. You can choose the Java or C# languages. Do not use a ready-made Machine Learning and Data Mining library.

CU : Sepal leaf length, CG: Sepal leaf width,
TU : Petal length, TG : Petal width,
N1 : Output neuron 1 N2 : Output neuron 2 N3 : Output neuron 3,



The most basic structures in Artificial Neural Networks (ANNs), which are Machine Learning methods and also form the basis of the deep learning field, are Artificial Neural Cells (Artificial Neurons). ANNs are used to solve many problems such as classification, clustering and prediction.

The structure of an artificial neural cell and an example computation process is shown in Figure 1. The neuron in the figure has 4 inputs (x) and 1 output (y).

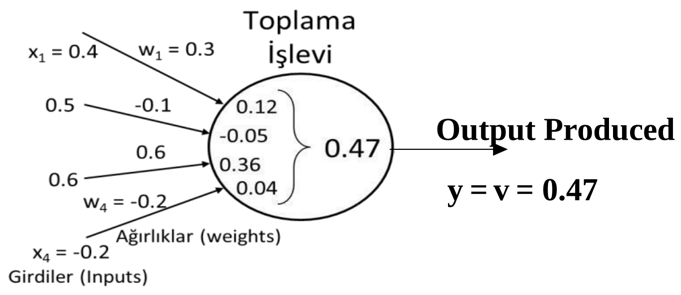


Figure 1: Nerve Cell (Neuron) Model and Functioning

The Summation Function is performed by taking the sum of the product of the inputs and the weights:

$$v_k = \sum_{i=1}^{n} w_i\, x_i \;= w_1\, x_1 \;+ w_2 x_2 \;+\; \dots + w_n x_n \;= [\, w_1\, w_2 \dots\, w_n\,] \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

In Supervised Learning, along with the inputs, the output values that should be (target) is given / provided by the system. When training the network, the iris.data dataset linked above should be used. In the training set each

```
5.1,3.5,1.4,0.2,Iris-setosa
```

There are 150 pieces of data (plant samples) in format. The first 4 values in the row are the attributes of the corresponding flower sample and will be given as input to the network, while the last value is the type of lily plant and will be used as the target, i.e. the expected value.

a) Create **a Neuron (Neural Cell) class**. Choose **appropriate data structures** to hold inputs and weights. Initially set all weights to random positive values between [0, 1]. **Write the method** or methods **that perform the calculations and necessary operations** (calculate the output of the neuron).

b) **Create the object that will contain the Neural Network class and 3 neurons. Write the training method** based on expected values and neuron outputs**: Before training, divide all input data by 10 and scale it to the interval [0, 1] (**if you want**,** you can do actual normalization instead of division**).** Training will be done according to a simplified learning rule: For data with output I Setosa, the expected value of N1, for data with output I Versicolor, the expected value of N2, for data with output I Virginica, the expected value of N3 is 1; the value of the other outputs is 0. The output values of the network (N1, N2 and N3) will be calculated for the relevant data, if the expected output value and the neuron with the largest value among the outputs produced by the network are the same, no action will be taken; if different;

- **The values of the weights (w) associated with the largest of the outputs produced by the network will be decreased by the formula w = w - (λ * x), where λ is the learning coefficient and x is the value of the input associated with the relevant weight.**

- **The values of the weights (w) related to the expected output will be increased by the formula w = w + (λ * x).**

c) **Train the network for 50 epochs with λ (learning coefficient) = 0.01**. An epoch is the process of giving all the training data (here 150 pieces) to the system once in a sequence and changing the weights. After the process is finished, feed the input data to the network in such a way that only the result is obtained (without changing the weights) and calculate the output values. At this stage, if the largest of the outputs produced by the network and the neuron with an expected value of 1 are the same, increase the number of known correct ones by one. Calculate and print the **accuracy value (number of correctly classified samples (data) / total number of samples)**. If 120 of the 150 data you have are classified correctly, the accuracy = 120/150 = 80 %.

d) Repeat this process again for **20 and 100 epochs**. Repeat the experiments for λ = 0.005 and 0.025 (20, 50 and 100 epochs). Record the accuracy values in a 3x3 matrix with the number of epochs (20,50,100) in the rows and λ values (0.005, 0.01 and 0.025) in the columns and present them in the report. Repeat the experiments in items c and d 2 more times and create 2 new matrices with the results. Observe whether the success values have changed. Think about why.