Evolution Strategies (1+1)-ES to train Multilayer Perceptron (MLP)

The aim is to implement the (1+1)-ES applied for changing weights in a given MLP architecture.

1 Training dataset

Use IRIS dataset for training `https://www.kaggle.com/uciml/iris`.

Important informations from dataset that influence MLP architecture:

1. number of attributes in dataset except species - number of inputs
2. number of species (3) - number of outputs

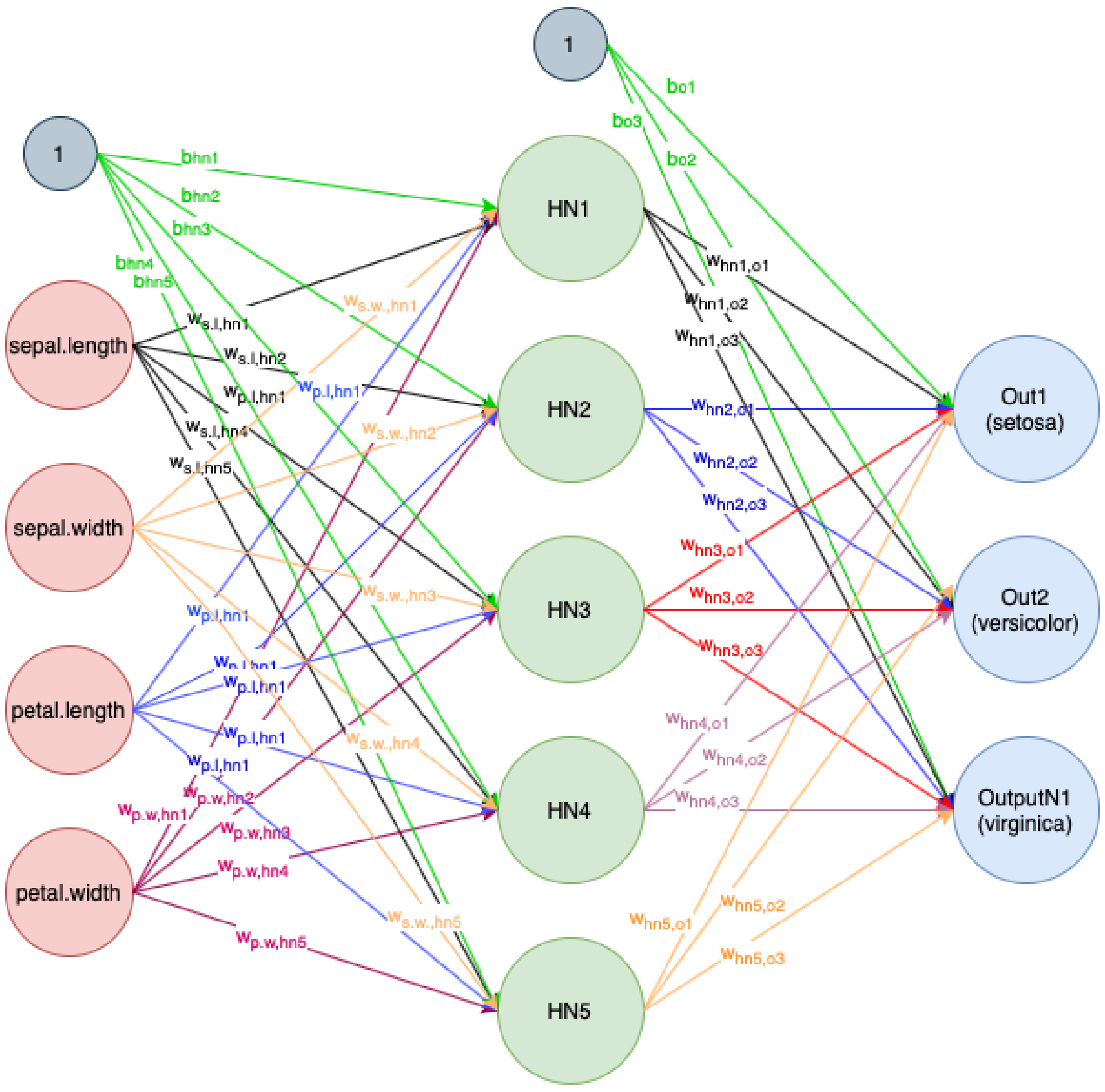
Number of samples (observations) is 150. Each class is represented by 50 observations. The class is categorical and need to be transformed into a binary vector as follows:

class	binary coding
setosa	1 0 0
versicolor	0 1 0
virginica	0 0 1

Each binary value is the output from neural network (3 output neurons).

2 MLP architecture

A picture presents a network architected that is to be trained. The hidden layer contains 5 neurons.



Training is a process which tunes weights (internal MLP parameters) to minimize the MLP's error. All weight compose an individual in ES, and error is the objective (fitness) function. The number of weights in proposed model is 43:

$$|w_{\{s.l,s.w,p.l,p.w\},hn_i}| + |b_{hn_i}| + |w_{hn_i,o_j}| + |b_{o_j}| = 4 * 5 + 5 + 5 * 3 + 3 = 43$$

1. $w_{\{s.l,s.w,p.l,p.w\},hn_i}$ - connections between inputs and first hidden layer ;
2. b_{hn_i} biases of every neuron in the hidden layer;
3. w_{hn_i,o_j} - connections between first hidden layer and the output neurons;
4. b_{o_j} for every neuron in the output layer;
5. $i \in \{1, 2, 3, 4, 5\}$ - index of hidden neurons;
6. $j \in \{1, 2, 3\}$ - index of output neurons.

3 (1+1)-ES for training MLP

```
$parent \leftarrow initialize(individual)$
$g \leftarrow 0$
**while** $g < max\_num\_gen$ **do**
  $s \leftarrow s\_mutation\,(s)$
  $offspring \leftarrow y\_mutation\,(parent, s)$
  $fitness\_function\,(offspring)$
  $parent \leftarrow selection(offspring, parent)$
  $g \leftarrow g + 1$
**end while**
**return** $parent$
```

4 Algorithm description

- *initialize*(*individual*) returns an individual (parent) which could be structure/object:
  - W — vector with 43 numbers which represents weights in MLP.

$$\begin{aligned}(w_{s.l,hn_1}, w_{s.w,hn_1}, w_{p.l,hn_1}, w_{p.w,hn_1}, b_{hn1},\\ w_{s.l,hn_2}, \ldots, b_{hn2},\\ ,\ldots,\\ w_{s.l,hn_5}, \ldots, b_{hn5},\\ w_{hn_1,o_1}, w_{hn_2,o_1}, \ldots, w_{hn_5,o_1}, b_{o_1},\\ w_{hn_1,o_2}, \ldots, b_{o_2},\\ w_{hn_1,o_3}, \ldots, b_{o_3})\end{aligned}$$

    Initialized randomly from uniform distribution, range $= [-1, 1]$ (real numbers).
  - s — vector with 43 numbers which represents standard deviation to control the mutation strength in every dimension. Initialized randomly from uniform distribution and the range $= [0.1 : 10]$.
  - function — fitness function $f(W)$- MLP error calculated as cross-entropy loss function (CE).

    Fitness function is mean cross-entropy loss function for all training samples.

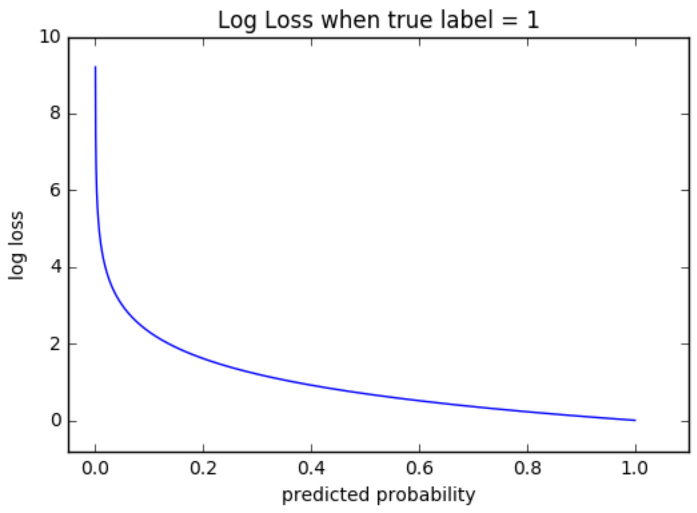
$$f(W) = -\sum_{i}^{n} \frac{CE_i}{n},$$

    where n is the number of samples/observations (in Iris dataset $n = 150$).
    Cross-entropy or log loss measures the performance of a classification model whose output is a probability value between 0 and 1. For a single training sample i CE:

$$CE_i = -\sum_{c=1}^{M} y_{i,c} \ln\left(Out_{i,c}\right),$$

where:

* $M = 3$ is the number of classes (in Iris dataset 3(setosa, versicolor, virginica)),
* ln - the natural logarithm
* y - binary indicator (0 or 1) if class label c is the correct classification for observation i. There are n observations or samples in the dataset .
* Out - predicted probability for observation i if class c (output from the MLP).



The predicted probability is calculated as follows:

1. Calculate the output of the first hidden layer using ReLU transfer function. The output from the $j-th$ hiden neuron is calculated as:

$$HN_{i,j} = ReLU(w_{s.l_i,hn_j}s.l_i + w_{s.w_i,hn_j}s.w_i + w_{p.l_i,hn_j}p.l_i + w_{p.w_i,hn_j}p.w_i + b_{hn_j}) =$$
$$= \max(0, w_{s.l_i,hn_j}s.l_i + w_{s.w_i,hn_j}s.w_i + w_{p.l_i,hn_j}p.l_i + w_{p.w_i,hn_j}p.w_i + b_{hn_j}),$$

   where:
   - HN is the output from the hidden neuron;
   - i is the sample/observation index $i \in \{1, \dots, 150\}$;
   - j is the hidden neuron index $j \in \{1, \dots, 5\}$;
   - $s.l_i$ sepal length in the i-th sample;
   - $s.w_i$ sepal width in the i-th sample;
   - $p.l_i$ petal length in the i-th sample;
   - $p.w_i$ petal width in the i-th sample;
   - $w_{s.l_i,hn_j}$ - weight connecting $s.l_i$ input with the hidden neuron hn_j;
   - b_{hn_j} hidden neuron Out_c bias.

2. Calculate the output from the network for class c and the ith sample using softmax function. The outputs from the hidden layer are multiplied by corresponding weights and sum up. The biases are added afterwords.

$$Out_{i,c} = \frac{e^{tmp_{i,c}}}{\sum_c^M e^{tmp_{i,c}}},$$

where:

$$tmp_{i,c} = w_{hn_{i,1},o_c} \cdot HN_{i,1} + w_{hn_{i,2},o_c} \cdot HN_{i,2} + w_{hn_{i,3},o_c} \cdot HN_{i,3} +$$

$$+ w_{hn_{i,4},o_c} \cdot HN_{i,4} + w_{hn_{i,5},o_c} \cdot HN_{i,5} + b_{o_c},$$

where:

- $M = 3$ is the number of classes (in Iris dataset 3(setosa, versicolor, virginica)),
- i is the sample/observation index $i \in \{1, \dots, 150\}$;
- c is the output index;
- HN is the output from the hidden neuron;
- $w_{hn_{i,1},o_c}$ weight connecting HN_1 hidden neuron with the output neuron o_c;
- b_{o_c} output neuron Out_c bias.

- $s\_mutation\,(s)$

  1. perform the (1+1)-ES for a number G generations (e.g $G = 5$)
     (a) keep s constant during that period
     (b) count the number G_s of successful mutation (when $fitness\_function(child) < fitness\_function(parent)$ during that period
  2. calculate $P_s = \frac{G_s}{G}$
  3. change s (every s_i):

  $$s_i := \begin{cases} s_i/a, & \text{if } P_s > 1/5 \\ s_i \cdot a, & \text{if } P_s < 1/5 \\ s_i, & \text{if } P_s = 1/5 \end{cases}$$

  4. goto 1

  Parameter $0 \leq a \leq 1$, a recommended value is $a = 0.85$.

- $y\_mutation(parent, s)$

```
for $i = 1$ to $length\,(parent)$ do
    $mutant_i \leftarrow y_i + s_i \cdot \aleph\,(0, 1)$
end for
return $mutant$
```

  where $\aleph\,(0, 1)$ is the randomly generated value from normal distribution.

- max_num_gen - maxim number of generations e.g. 10000