Artificial Intelligence – Lab Assignment 4

UNIZG FER, academic year 2013/14

Handed out: June 4. Due: June 12 at 23:59

The topic of this lab assignment are neural networks and the genetic algorithm. The task is to write a program for training a neural network (i.e., determining its weights) using a genetic algorithm. A neural network of the type used in this assignment is typically trained with the backpropagation algorithm. However, training a network using a genetic algorithm is applicable to various types of networks and has additional advantages. The main idea is to build a population of neural network instances (each individual corresponds to a single neural network instance and compactly encodes all its weights). We then use the genetic algorithm (which applies the selection, crossover, and mutation operators) to optimize the weights with respect to the error of the network on the training set. In this assignment the network architecture is fixed; in a more general case the procedure may also be used to optimize the network structure along with its weights. After the network has been trained, we must check how well it performs. To this end we must use a test set, which is different from the training set. You can implement this task in a programming language of your choice.

Neural network

You will use a fully connected $1 \times n \times 1$ neural network (single input neuron, n neurons in the hidden layer, and one output neuron). The neurons in the hidden layer use a sigmoidal activation function, while the neurons in the input layer use the identity function ($f(x) = x$). Throughout the network training, the program should print out the iteration number and the network error (the square error of the output neuron averaged over all training instances). The number of individuals and the termination criterion (the maximum allowed network error and the number of iterations) are specified by the user. After the training is completed, the program must allow the user to provide new input examples and for each example print out the network output. Moreover, the program must allow the user to specify a file with test instances (a test set) and then print out the network error on the whole test set (mean square error).

Genetic algorithm

You will use a genetic algorithm to train the network. More specifically, you will use a generation-based genetic algorithm with elitist strategy, ensuring that the best-fitted individual is copied into the next generation. Each individual encodes the weights of the network as an array of real numbers – the size of the array equals the number of weights. To crossover two individuals, compute the arithmetic mean between their corresponding values. To mutate an individual, add to its value a random number drawn from the normal distribution $N(0, K)$, where K is a predefined value. Take care than only p percent of the networks weights may be modified in a single mutation operation (let $p = n/N$, where n is the number of neurons in the hidden layer N is the number of weights). Test the behavior of the algorithm using various values of K (e.g., $K = 0.1$, $K = 1$, and $K = 10$). Parameter p should be set in the program. For each individual network, you should compute the network error as mean square error of its outputs. You are to use the fitness-proportional selection strategy.

Training set

The network should be trained to do the mapping $y = f(x)$, defined as follows (follow this link to get the data set):

x	$f(x)$
0.026	1.050
0.220	1.341
0.693	1.461
0.701	1.458
0.722	1.448
0.896	1.342
1.166	1.149
1.198	1.128
1.223	1.112
1.254	1.095
1.382	1.035
1.667	1.009
1.964	1.141
1.988	1.157
2.512	1.484
2.733	1.479
2.929	1.333
3.009	1.229
3.036	1.189
3.229	0.811
3.240	0.785
3.515	0.004
3.711	−0.659
3.769	−0.863
3.857	−1.173
3.958	−1.521
3.992	−1.632
4.092	−1.951
4.165	−2.166
4.809	−2.972
4.819	−2.966
4.854	−2.940

Test set

Machine learning models are typically trained on one data set (the training set) and evaluated on another data set (the test set). In this way one can get a good estimate of how well the network generalizes over unseen examples. Test the performance of your network using the following data set (follow this link to get the data set) and report the network error (mean square error over all instances in the test set):

x	$f(x)$
0.049	1.093
0.197	1.315
0.436	1.488
0.837	1.382
0.865	1.364
1.021	1.251
1.047	1.233
1.056	1.226
1.087	1.203
1.117	1.182
1.399	1.029
1.536	1.001
1.776	1.041
1.935	1.123
2.087	1.227
2.350	1.410
2.366	1.420
2.407	1.442
2.415	1.446
2.773	1.461
2.818	1.434
2.984	1.265
2.993	1.252
3.005	1.235
3.147	0.989
3.368	0.451
3.466	0.159
3.52	−0.013
4.126	−2.054
4.483	−2.844
4.531	−2.902
4.854	−2.940