CS 470/670 – Intro to Artificial Intelligence – Spring 2016

Instructor: Marc Pomplun

Assignment #5 Sample Solutions

Question 1: Classifying Iris Flowers

You can find a sample solution file iris_nn.c in the "Software" section of the course homepage. It gets an average of 93 to 94% correct classifications.

For CS670 students: What happens if you train with 20 flowers and test performance on the remaining 80? Do not submit the modified code but just report the average accuracy over 10 runs that you obtain.

It turns out that even with such few training exemplars, average performance drops only very little. If you decrease the number of training exemplars below 10, then you will see strong decrease in performance.

Question 2: The Soccer Network

First you have to consider how to format the input to the network:

- The skill level represents a scale, and therefore we only need one neuron to represent each value. Just divide the level by 10 to get input values ranging from 0 to 1. If you only consider the mean level or the mean and its standard deviation, that is also OK. But if you consider each individual player's level, you need a total of 40 neurons.

- The number of matches that each team has played during the last two weeks can be represented by two neurons, one for each team. As there are never more than seven matches in that period of time, just take each number and divide it by seven.

- The statistics of former matches between the same two teams within the past 10 years can be represented by three neurons representing the proportions of (1) wins for Team A, (2) ties, and (3) wins for Team B.

- The continent that each team comes from does not represent a scale, and therefore we have to use six neurons to represent it. Then we could use the following input patterns: North America: 100000, South America: 010000, Europe: 001000, Africa: 000100, Asia: 000010, and Australia: 000001).

- To represent where the match takes place (Team A's stadium, Team B's stadium, or neutral place), you could either use three neurons in the same way as for the continent, or you could use one neuron and consider the following scale (from best to worst for Team A): Team A's stadium, neutral place, Team B's stadium.

- The phase of the soccer season (early season vs. late season) requires one neuron with value 0 or 1, respectively.

Then the input vectors consist of 40 + 2 + 3 + 6 + 3 + 1 = 55 elements.

Regarding the output (predicted number of goals scored by each team), these values form scales and should be represented by one neuron each. Let us say that for high-level matches there will never be more than seven goals per team (remember Germany vs. Brazil at the 2014 World Cup!). Then we can define that we multiply the output by seven and round to the closest integer in order to get the goal prediction for each team.

So your network needs 55 input neurons, maybe 20 hidden-layer neurons (just a guess), and 2 output neurons. As usual, each hidden neuron and each output neuron also receive an extra "offset" input.

In the next step you have to access soccer databases to obtain exemplars for training and testing. For each match in the database, you collect all required data to form the 55-element input vector and the 2-element desired output vector (the final score). These vector pairs serve as your exemplars. Make sure that you cover the input space as best as possible. For example, you should cover all continents, all phases of the season, and all types of competitions, leagues, and skill levels. Since our network has $56 \cdot 20 + 21 \cdot 2 = 1162$ weights, we should ideally have at least ten times as many training exemplars, and just as many test exemplars, so it would be best to collect data from at least 23,000 matches.

We now initialize the network with random weights and then pick the training exemplars in random order. The input part of each exemplar is fed into the input neurons, and the network's output is compared to the output part of the exemplar. Then the weights are adjusted according to the backpropagation rule. We repeat this procedure until the network error, i.e., the mean squared error between actual and desired outputs, falls below a threshold that we determine empirically. Then we freeze the weights and measure how well the network predicts the outcomes of the test exemplars.

If the performance is high, we have a way to make lots of money!

(In reality, this would not work well. There are too many factors influencing soccer results, including coincidences that cannot be predicted.)

Question 3: From Gradient Descent to Backpropagation

Explain in your own words how the concepts of gradient descent and backpropagation are related to each other.

Gradient descent is a technique for efficiently finding the input x_0 to a function f that (ideally) produces the absolute minimum of f. It starts with an estimate of x_0 and iteratively refines it by moving against the function's gradient. This is exactly what backpropagation does. In backpropagation, the function to be minimized is the mean squared error of the network, and the inputs to this function (i.e., the variables that the error depends on) are the network weights. In other words, we have to find those weights for which the network error is minimal, and we do this using gradient descent in a typically very high-dimensional function.