UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

Artificial Intelligence
COS314

Project 2: Neural Networks

Due: Monday 9 May, 14:00

For this project you will implement a feedforward neural network, trained using gradient descent, to predict which uppercase letter is represented by an image. You will therefore implement the backpropagation learning algorithm. A summary of the learning algorithm is given in Section 2. The letter recognition problem is described in Section 1. Section 3 outlines the procedure to follow to complete this project.

1 The Letter Recognition Problem

The classification problem to solve using a neural network is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique patterns. Each pattern corresponds to one character in some arbitrary font. Each of these patterns was converted into 16 primitive numerical attributes (statistical moments and edge counts) which characterizes the corresponding letter. These numerical values were then scaled to fit into a range of integer values from 0 through 15.

Download the files `letter-recognition.data` and `letter-recognition.names`. The first file provides a description of the attributes used to describe a letter, while the second file contains the actual data. Go through these files so that you are familiar with their contents and the characteristics of each of the attributes.

2 Backpropagation Learning Algorithm

Find below a pseudocode algorithm to train a feedforward neural network that consists of an input layer, one hidden layer, and an output layer, using gradient descent. Please refer to your notes for the mathematical details. Note that this algorithm implements stochastic learning, and not batch learning.

1. Decide on your NN architecture.
2. Perform pre-processing on the data. That is, do the necessary scaling on input and target values.
3. Shuffle the pre-processed data set, and then divide it into a training set, D_T (60% of the data), validation set, D_V (20%) of the data, and a generalization set, D_G (20%) of the data.

4. Initialize all the weights (including the threshold values) to random values in the range $[-\frac{1}{\sqrt{fanin}}, \frac{1}{\sqrt{fanin}}]$, where $fanin$ is the number of weights leading to the neuron.

5. Initialize values for η (the learning rate), α (the momentum), E_T (the training error), $\xi = 0$ (the epoch counter), ξ_{max} (the maximum number of epochs), and ϵ (the desired training accuracy[1]).

6. Repeat until a convergence criterion has been satisfied

   (a) $E_T = 0$, $\xi++$

   (b) for each pattern in the training set D_T,

      i. Compute the net input, net_{y_j}, to each hidden unit.

      ii. Compute the activation, y_j, of each hidden unit (using the sigmoid activation function).

      iii. Compute the net input, net_{o_k}, to each output unit.

      iv. Compute the activation, o_k, of each output unit (using the sigmoid activation funciton).

      v. Determine if the actual output, a_k, should be 0 or 1, as follows: If $o_k \geq 0.7$, then let $a_k = 1$, meaning that the letter recognized is that represented by the k-th output unit. If $o_k \leq 0.3$, then $a_k = 0$, meaning the letter is not that represented by the k-th output unit. For outputs between 0.3 and 0.7, the network is uncertain about the classification, and you record a classification error for that pattern.

      vi. Determine if the target output has been correctly predicted. Let $e = 1$ if the target is correctly predicted (if $t_k = a_k$ for all output units, then the target is correctly predicted); otherwise, $e = 0$.

      vii. $E_T += e$

      viii. Calculate the error signal for each output

$$\delta_{o_k} = -(t_k - o_k)(1 - o_k)o_k$$

      ix. Calculate the new weight values for the hidden-to-output weights

$$\Delta w_{kj}(\xi) = -\eta \delta_{o_k} y_j$$

$$w_{kj}(\xi) += \Delta w_{kj}(\xi) + \alpha \Delta w_{kj}(\xi - 1)$$

      x. Calculate the error signal for each hidden unit:

$$\delta_{y_j} = \sum_{k=1}^{K} \delta_{o_k} w_{kj}(1 - y_j)y_j$$

      where w_{kj} is the weight between output unit k and hidden unit j.

      xi. Calculate the new weight values for the weights between hidden neuron j and input neuron i

$$\Delta v_{ji}(\xi) = -\eta \delta_{y_j} z_i$$

$$v_{ji}(\xi) += \Delta v_{ji}(\xi) + \alpha \Delta v_{ji}(\xi - 1)$$

   (c) Calculate the percentage correctly classified patterns as $E_T = E_T/P_T * 100$, where P_T is the total number of patterns in the training set. Do the same for the generalization set, D_G.

   (d) Save the epoch number, E_T and E_G to a file.

3 Procedure to Follow

For this project you will submit your code (which can be in C++ or Java, but running on Linux) and a pdf document. You have to provide a Makefile and a Readme file (to explain how to compile and run your code). The content of this document will be specified below. Please provide the items to be included in the document in the same order as specified, and under appropriate section headings. Please provide your name and student number as the first item on your document. Please note that this document MUST BE in pdf.

You need to follow the following steps to complete this project:

1. Study the letter recognition problem and the given data files.

[1] Do not use the mean squared error, but the percentage of correctly classified patterns as the training accuracy.

2. Perform the necessary pre-processing on the data to be in a format acceptable for neural network training. Your pre-processing should be such that the training process is optimized. For this you will have to remember the discussions in class. Describe in your document exactly how you have pre-processed the data, and provide motivations. [Marks: 20]

3. Decide on which stopping conditions you will use, and provide motivations. [Marks: 5]

4. Experiment 1: For this experiment, you will simply distinguish between one letter and the rest. In other words, if the letter to recognize is 'A', then the classification will be if a pattern corresponds to an 'A' or not. You will not distinguish between all 26 letters. Therefore, your neural network will have only one output unit. If the actual output, $a_k = 1$, then letter 'A' is recognized, otherwise the pattern represents any of the other 25 letters. Your program should be generic to allow the user to specify any letter to be recognized. For this experiment you need to determine the best number of hidden units, and the best value for the learning rate and the momentum. In your document, list these values, and give the training and generalization errors. You also need to provide motivations for why you say these values are best. The way to do this is to provide a table with different values for the number of hidden units, learning rate and momentum that you have tried, and the associated errors. You may even include graphs to illustrate this table. You have to include a graph to illustrate the learning profile, i.e. a line graph where the x-axis represents epoch number, and the y-axis represents accuracy. [Marks: 50]

5. Experiment 2: Here the task will be to distinguish between vowels (A,E,I,O,U), and non-vowels. Again, use only one output unit. Report the same information as above. [Marks: 20]

6. Experiment 3: Now, you need to identify the appropriate letter from the 26 letters. In other words, your network will have 26 output units. Report the same information as above. [Marks: 30]

7. Experiment 4: Repeat experiment 3, but use batch learning. In your report say which of stochastic learning or batch learning did the best. [Marks: 30]

8. Submission procedure: Submit separate programs for each of the experiments, and name each main program (and the executable) as `experiment?`, where the '?' is replaced with the experiment number. For each experiment, the program must use the original data file provided. All required processing must occur on this file. Submit a compressed archive with all the necessary files to successfully compile and execute your programs, as well as your report. Name your report `s????????.pdf`, where the question marks are replaced with your student number. If your archive expand to subfolders, then your report must be in the root folder, and not in any subfolder.

For the above, please note that training starts with random weights. This means that you should not make any decisions based only on one training session. To make valid decisions, one should train a neural network on the same problem at least 30 times. The average performance for these 30 networks is then used to make decisions. This process is done as follows: Repeat the training process 30 times starting from step 3 of the training algorithm, and for each of the 30 training sessions record the final network accuracy. Compute the average over these, and use this average to guide your decisions. Be aware that the training will take long, so, allocate enough time for this. State in your report, for each of the experiments, over how many training sessions have you based your decisions (if time becomes a problem, you may use less training sessions, but at least 5; note that the more training sessions you use, the more accurate your conclusions will be).