CPSC 420-500: Program 3, Perceptron and Backpropagation

Yoonsuck Choe
Department of Computer Science
Texas A&M University

October 31, 2008

1 Overview

You will implement perceptron learning from scratch (see section 3 for details), and train it on AND, OR, and XOR functions. Then, you will take an existing backpropagation code (see section 4), train it and test it under different conditions and report your findings. The same three boolean functions will be used for the backpropagation learning. For further details on perceptrons and backpropagation, see the lecture slides and also Hertz et al. (1991). Specific submission instruction will be given in section 5 and section 6.

2 Language and OS

You may use either C/C++, Java, Matlab (or Octave), or Lisp. The resulting code should be able to compile and run on the departmental unix host (unix.cs.tamu.edu). You may use a different language with a permission from the instructor, in which case you will be asked to do a demo in front of the TA.

To compile your programs other than Lisp (which you already know), see the following instructions.

- C program file: **perceptron.c**
  to compile: **cc -o perceptron perceptron.c -lm**
  to execute: **./perceptron**
- C++ program file: **perceptron.C**
  to compile: **c++ -o perceptron perceptron.C -lm**
  to execute: **./perceptron**

- Java program file: **perceptron.java**
  to compile: **javac perceptron.java**
  to execute: **java perceptron**

The full paths for the compilers are (on compute.cs.tamu.edu):

- /usr/bin/cc
- /opt/csw/gcc3/bin/g++
- /usr/jdk/latest/bin/java
- /usr/jdk/latest/bin/javac

3 Perceptron

Perceptron activation is defined as:

$$Output = step\left(\sum_{i=0}^{2} W[i] * INP[i]\right), \tag{1}$$

where $step(X) = 1$ if $X \geq 0$ and $step(X) = 0$ if $X < 0$ (see figure 1).

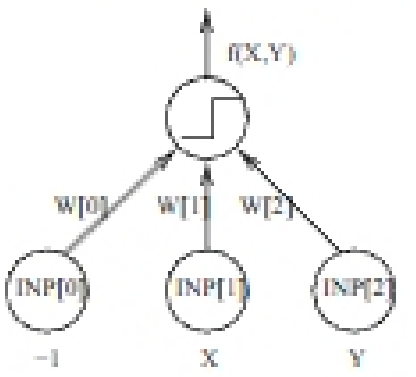

Figure 1: **Perceptron.** The input $INP[0]$ is the *bias unit*, fixed to -1, and has an associated weight $W[0]$, which is the threshold. The two inputs X and Y are given and the output $f(X, Y)$ will be calculating (or attempting to calculate) a boolean function, one of OR, AND, or XOR.

Implement a perceptron with two input units (three, including the bias unit that has a fixed input value -1) and one output unit. Your program should take three inputs from the command line.

1. maximum number of epochs to run (integer),
2. learning rate parameter α value (double),
3. function selection string ("and", "or", and "xor").

For example, a typical run would go like this ($ is the unix prompt):

```
$ ./perceptron 10000 0.0001 and
```

A pseudo code for perceptron learning is as follows:

1. Initialize weights to random numbers between 0.0 and 1.0. Note that you have just one set of 3 weights ($W[0], W[1], W[2]$). You will train this same set using the four input-target pairs.
2. Initialize epoch count to 0.
3. while sum of $(target - output)^2$ for all input patterns is not 0 do:
   - for each input-target pattern
     (a) present input and calculate output
     (b) calculate the $error = target - output$
     (c) update the weights $W[i]$ using the perceptron learning rule.
   - endfor
   - increment epoch count
   - if (epoch count > max epochs), break from while loop
4. Print out the output for the inputs (0,0), (0,1), (1,0), and (1,1).

4 Backpropagation

For backpropagation, you will download the following file (make it one line):

```
http://faculty.cs.tamu.edu/choe/
src/backprop-1.6.tar.gz
```

and run it under different conditions. First, you need to unzip and untar it by running the following:

```
$ tar xzvf backprop-1.6.tar.gz
$ cd backprop
```

then read the README file to learn how to compile and run it.

Running the bp program (which is generated by compiling) will give you a huge dump on the screen. To selectively view the data you're more interested in, use the `grep` command. For example, to view the progression of error: