ENGR110.
Machine Learning.
Assignment

September 2, 2019

Abstract

This Assignment is preparation for the ML Project. We start with simple example and gradually introduce more code until we have functional neural network code.
All coding is done in C++.

Submission is due on 16 September, Monday, midnight.

5% percent of course marks.

Not a group assignment.

1 Tools

We will be using visualisation software called **gnuplot** in this assignment.
You can find it at `http://www.gnuplot.info/`.
You can start it using command line interface by typing **gnuplot**.
If there is a text file name "gs1.txt" with 2 whitespace separated columns,

```
0  453
1  427
2  403
3  379
4  357
```

you can plot second column value versus first column using **plot "gs1.txt" using 1:2 with lines**.
If file contains three columns you can plot value in 3rd column as z against
splot 'ex2plot.txt'

2 Simple neuron output

As a start we will implement NAND gate as a neuron.
Truth Table of NAND gate is:

$x[0]$	$x[1]$	Target $t[]$
0	0	1
0	1	1
1	0	1
1	1	0

The Truth Table is our training set. We can draw Truth Table as shown in Fig.1 - x[0] and x[1] along (x,y) axes and t values along z.

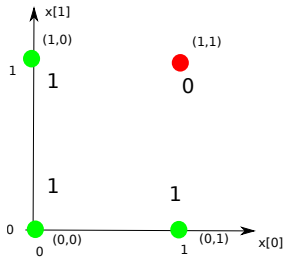

Figure 1: Target(answer) as a function of x_0 and x_1

Exercise 1, 10 points

Using Fig.1 as a template draw two regions: one region contains all output 0s and another all 1s. Boundary between regions should be the straight line. Include the picture into your report.

Now we will train NN to do same thing as you just did.

Neuron output is:

$$y = \sigma(bias + w_0 \cdot x_0 + w_1 \cdot x_1) \tag{1}$$

, where

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

First, we visualise what an error $y - t$ looks like if w_0 and w_1 are changing.

Download, compile, link and run program Ex2.cpp. Read the code. Understand what each function does.
Program produces file with $error(w_0, w_1)$ surface for fixed value of bias b. Weights w_0 and w_1 are changing resulting in changing error.

Exercise 2, 15 points

- Explain what an **error** variable represents and how it is calculated in Ex2.cpp.
- Include picture produced gnuplot taking output file of Ex2.cpp. Try different value of b to obtain another picture and include that too.
- Explain what values of weights w_0 and w_1 for given value of bias b are best to provide minimum of the error.

Exercise 3, 10 points

Modify Ex2.cpp:

- It asks for values of b,w_0 and w_1 (you can investigate C++ **cin** operator)
- It calculates output y using these values of b,w_0 and w_1 for 4 combinations of the inputs x_0 and x_1 : {0,0},{1,0},{0,1},{1,1} using equations (1) and (2). You can use some parts of Ex2.cpp if you like.
- It calculates error for each input combination
- It calculates and prints combined error

Include code (as a text, not picture) in your submission.
Run your program at several times for different values of b,w_0 and w_1. Try combination from Exercise 2 as well.
Explain which combination is better.

```
import numpy as np
import matplotlib.pyplot as plt

b=15.2
w0=-10
w1=-10

x0 = np.arange(0.0,1.1,0.1)
x1 = (-b/w1)  + x0*(-w0/w1)
plt.plot(x0,x1)
plt.ylim((0,1))
plt.xlim((0,1))
plt.show()
```

Exercise 4, 5 points

Simple Python program above can plot neuron decision boundary , solving the equation $b + w_0 \cdot x_0 + w_1 \cdot x_1 = 0$ for given Run the Python program: type **idle3** in CLI, click "File-New file", type in the program, change values of b, w_0, w_1 to your liking, push F5 to run. Compare how different is your answer from Exercise 1 and plot produced by this Python program.
Try different values of b, w_0, w_1 which are far from optimum.
Include pictures and brief explanation in your submission.

3 Search for best values of b,w_0 and w_1

Searching all possible combinations (global search):

Exercise 5, 15 points

Save Ex2.cpp (or your program from Exercise 4) and modify it as follows:

- Introduce new function **GlobalSearch()**
- In this function run three nested cycles for b,w_0 and w_1 calculating error every time. You may want to change the resolution.
- Remember best (providing minimum error) values of b,w_0 and w_1
- Print these values together with value of **error**

Submit you code of **GlobalSearch()** function.

Try to time execution time of the code. You may want to recall timing C++ functions from ENGR101 lecture on PID.

Accelerating search - using gradient
Background:

- To calculate in simplest possible way the gradient $\frac{de}{db}$:
  - Calculate value of e, save it as **e0**
  - Increment b by small amount d: $b = b + d$
  - Calculate value of e again, save it as **e1**
  - Calculate $\frac{de}{db}$ = (e1-e0)/d
  - Restore value of b: $b = b - d$

  Gradient estimations are

- To make one step of search:
  For each of b,w_0 and w_1 make one step in the direction of smaller error: $b = b - learning\_rate * \frac{de}{db}$ Repeat it for w_0 and w_1

- To conduct search following gradient direction at each step:
  1. Declare variable nstep, make it 0.
  2. Calculate error
  3. Calculate gradient
  4. Make one step
  5. Calculate error again
  6. If new value of error is too big, repeat form 1

  Best implemented as while() loop

Exercise 6, 20 points

Modify your code from last exercise:
Introduce variables to store the gradient components $\frac{de}{db}$, $\frac{de}{dw_0}$ and $\frac{de}{dw_1}$ (last two as an array size 2).
Introduce three more functions:
void CalculateGradient();
void GradientStep();
void GradientSearch();

Program and test these functions. Submit your code for these functions.

Hint for next Exercise: You can plot several graphs on top if you enter in gnuplot : plot "ex6plot_1.txt" using 1:2 with lines, "ex6plot_2.txt" using 1:2 with lines
You can chain as many file names as you wan

Exercise 7, 25 points

Run several searches with different values of *learning_rate*, at least try 0.1; 1.0; 5.0; 10.0; 30.0
Plot convergence of the **error** versus number of search steps for different values of *learning_rate* on same graph.

Explain what is happening with search convergence as *learning_rate* increases.

4 Submission

1. Please submit *.pdf file. Answer all boxed questions.
2. For Exercises 3,5 and 6 you should submit the code. Submit all of your C++ code in one zip file. Each code file name un-zipping the archive should be in form ex3.cpp (as an example). Submit whole program. If submitted code does not compile and run on ECS computers - mark for this Exercise is 0.