EEE 498 ML with application to FPGAs

Homework 2

All homework is handed in online. Don't hand in pictures of the code files, hand in the actual code files so I can see if they work. Hand in images of outputs of your code (displays, plots …) converted to pdf. If there are questions to be answered answer them in PowerPoint, Word, or on paper and convert to pdf (It doesn't actually matter how you get to the pdf).

Organize your work so that the answer to each problem can be easily identified. You will not get credit if your submission is poorly organized, or if the grader cannot find the answers.

You can use any language python, c, C++, MATLAB,…, to do coding though Python is likely the easiest and easiest to translate from lectures.

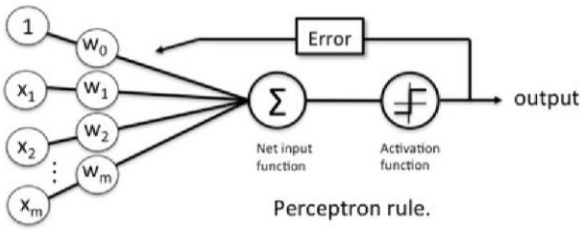
Be careful copying code from office products to Spyder or other editors, some characters are changed like single quotes. Sometimes invisible characters occur causing you to get an error on a line until you completely retype it.

1) What is the Perceptron Cost function? Does it have a finite derivative? If so differentiate it and provide that solution, if now why not?
   What If we said that the Cost function it is piecewise continuous? That would suggest you could take the derivative in some regions? Redo your answer on that basis.

2) Write your own Perceptron code. Train using the dataset using the data given below. Inputs are columns x0,x1,x2 which have only 5 observations and output target known is y. Use the Perceptron cost function and mathematics from Lecture 5. **Use compact notation.**

	x0	x1	x2	y
1	-2	4	-1	-1
2	4	1	-1	-1
3	1	6	-1	1
4	2	4	-1	1
5	6	2	-1	1

This is a general perceptron flow chart but don't take it too literally. Note that the perceptron weight is updated based on each observation, individually. Note that it uses the ReLU activation function.



Perceptron rule.

Run it for the dataset provided and turn screen grabs of your output. An iteration is a complete pass through all the data. This is an iterative solution so it will have multiple passes, but you want it to have as few as possible. A quantizer is used on the output, all positives become 1, all negatives become -1.

Your output should contain

1) A display of the value of the prediction, weights, and error in each iteration as the algorithm converges to a solution
2) At the end a plot of error between your prediction of y and the actual y (which we call a target) as a function of iteration
3) Percent error
   (I used

   ```
   from sklearn.metrics import accuracy_score
   print('Accuracy: %.2f' % accuracy_score(y, ypreds))
   ```

)
4) Number of misclassified cases in the final prediction

Make sure to turn in your code files and screen captures of all output.

3) What is the Perceptron Softmax Cost Function? Find the gradient of this cost function? Show your work.

chain rule

$$\frac{df(u)}{dx} = \frac{df(u)}{du}\frac{du}{dx}$$

for this problem

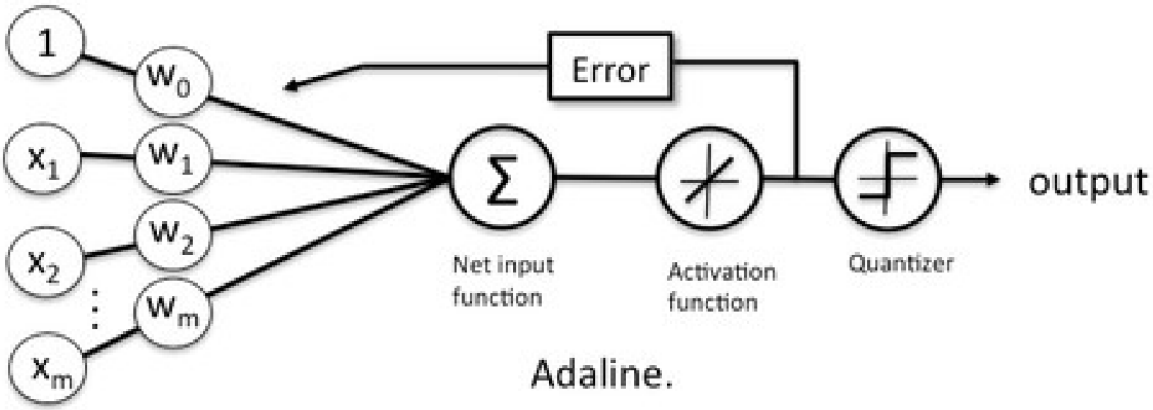
note

$$\frac{d\log(u)}{dx} = \frac{1}{u}\frac{du}{dx}$$

$$\frac{de^z}{dx} = e^z\frac{dz}{dx}$$

4) Adaline was a competing machine learning algorithm shortly after the Perceptron was published. It more directly uses gradient descents and has a linear activation function rather than the ReLU activation function used in the Perceptron. Note that the weights are updated for all cases not just the error cases. Like the Perceptron it has a quantizer to translate the linear output into 1 and -1, which is explicitly shown in the flow here. Use a least-squares cost function for your algorithm. Study the Adaline flow shown before and try to understand the differences between it and the Perceptron flow.

   1) Normalize/standardize the dataset as described in the lecture
   2) Use a random 2/3 of the dataset for training and 1/3 for testing as described in the lecture
   3) If you do update the weight for all observations, rather than the ones that have error your solution may find the best solution and the diverge away from that solution so stop iterating when the error drops below a tolerance, I used while (t<epochs and total_error/Nobs>tol): where Nobs is the number of observations, and tol=0.001.
      In your assessment of whether it has error, you should quantize your prediction.



Also, we need a larger dataset to do this because we average weights over a statistical sampling. Two are provided. **Dataset_1.csv has no noise, and Dataset_2.csv has noise.**

Develop you code with Dataset_1.csv and then run it again on Dataset_2.csv. Comment on your results. Results in each case should contain %error, number of misclassification, and number of iterations.

A pseudo code for Adaline

```
t = 0; epochs = 100; tol = 1e-6;
while (t<epochs and abs(total_error-last_total_error)>tol):
        last_total_error = total_error
        Ypred = model(X,w)
        Error = Ypred-Y
        w -= eta*dCostF(Error,X)
        total_error = np.dot(Error,Error)
        errors.append(total_error/len(X))
        t += 1
        Yq = quantizer(Ypred)
        print( t, total_error,Yq)
plot total_error vs t
```

dCostF is the derivative of the cost function. There are of course many ways to implement it but this is perhaps the simplest. This is just a guide.

5) Use something like sklearn.linear_model.Perceptron to compare your results for this and problem 4. This is a canned library version of the Perceptron algorithm which makes a nice reference. Screen capture your results. Compare number of miss-classifications, and accuracy for each dataset. As always, hand in your code file.

Hand in all output as a screen grab for each problem converted to pdf. It is a good practice to paste well organized and labeled images into PowerPoint, answer all questions in the PowerPoint file, and then convert the PowerPoint file to pdf. Hand in all your code files. If code files are missing you will not get credit. It is typically easier to grade individual files rather than a zipped file, but either is accepted.

[1] B. Widrow, "An Adaptive "Adaline" Neuron Using Chemical "Memistors."," in "Solid State Electronics Laboratory Technical Report ", October 1960, vol. No 1553-2.

[2] S. Raschka. "Machine Learning FAQ What is the difference between a Perceptron, Adaline, and neural network model?" https://sebastianraschka.com/faq/docs/diff-perceptron-adaline-neuralnet.html (accessed.