24-787 Artificial Intelligence and Machine Learning for Engineering Design
Homework 3
Neural Networks

Boolean Algebra

1. *(20 points)*

   Create a neural network with a single hidden layer to represent the Boolean expressions listed below. Use as few hidden neurons as possible. Manually assign weights such that the network perfectly classifies all possible combinations of the input variables. You may assume that the activation function for all hidden and output units is a linear threshold at zero (step function). Draw your network and show all weights.

   (a) $(A \wedge B) \oplus (A \wedge C)$

   (b) $(A \oplus B) \wedge (C \oplus D)$

Visualizing Decision Boundaries

2. *(25 points)*

   In this problem, you will be feeding test data through a given network and visualizing the results. Consider a 2-2-1 neural network with bias. Let the activation function at the hidden and output units be the hyperbolic tangent function given by $f(z) = a \tanh(bz)$, where $a = 1.716$ and $b = 2/3$.

   Suppose the matrices defining the input-to-hidden weights ($W^{(1)}$) and the hidden-to-output weights ($W^{(2)}$) are given as, respectively,

$$\begin{bmatrix} 0.5 & -0.5 \\ 0.3 & -0.4 \\ -0.1 & 1.0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1.0 \\ -2.0 \\ 0.5 \end{bmatrix}$$

   where $w_{ij}^{(1)}$ represents the weight connecting the ith input unit to the jth hidden unit, and $w_{jk}^{(2)}$ represents the weight connecting the jth hidden unit to the kth output unit. The first row in all weight matrices corresponds to the bias.

   (a) Sketch the network described above and label the weights accordingly.

   (b) Write a MATLAB script to evaluate the network at all points in the 2-dimensional input space defined by $\{(x_1, x_2) \in \mathbb{R}^2 | (x_1, x_2) \geq -5, (x_1, x_2) \leq 5\}$. Use a resolution of 0.2; in other words, each dimension should vary from $-5{:}\ 0.2{:}\ 5$. This yields a total of 2601 (x_1, x_2) input points.

   (c) Plot the results, using a green 'x' for positive outputs and a red 'o' for negative outputs.

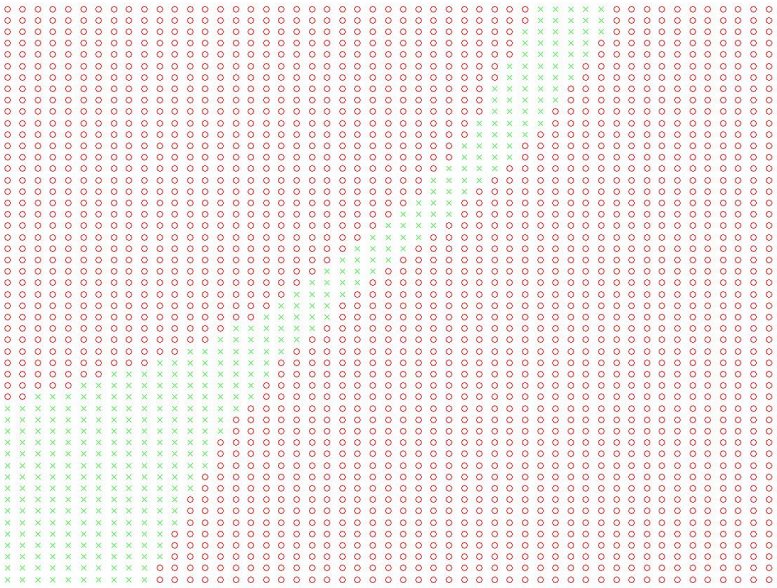
   (d) What is the value of y (the unthresholded output of the network) at $(x_1, x_2) = (2.2, -3.2)$ and $(x_1, x_2) = (-3.2, 2.2)$?

(e) Repeat (a)-(d) for the following weight matrices:

$$\begin{bmatrix} -1.0 & 1.0 \\ -0.5 & 1.5 \\ 1.5 & -0.5 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0.5 \\ -1.0 \\ 1.0 \end{bmatrix}$$

In your report, be sure to include the output plots as well as relevant equations or pseudocode to describe how you solved this problem.

Extra credit: *(10 points max)* Considering your output plots for the two cases above, describe in a few sentences how you might design a neural network to encode the decision boundary given in the figure below.

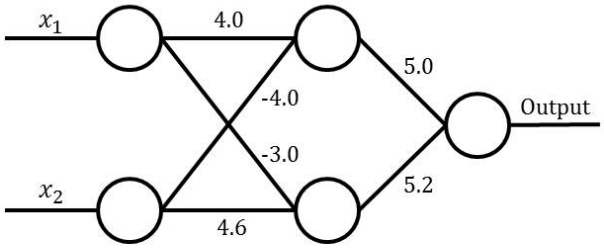


Sketch the proposed network and label the weights accordingly. Also, specify the activations functions used.

Backpropagation

3. *(20 points)*

Consider the following 2-2-1 neural network with no bias terms:



The activation function for all hidden and output units in the neural network is the logistic function, $f(x) = 1/(1 + e^{-x})$.

(a) Show that $f'(x) = f(x)\big(1 - f(x)\big)$.

(b) Suppose you have one training example given by $(x_1, x_2) = (-0.5, 1)$. What is the output of the network for this input? Pay attention to the network structure and activation function.

(c) If the desired output for the training example is 0.9 and the learning rate η is 0.10, what is the update to the weight (Δw) between the second hidden node and the output node (currently valued at 5.2)? What is the update to the weight between the first input node and the first hidden node (currently valued at 4.0)? Show your calculations.

Extra credit: *(5 points max)* Compute the updates for the remaining weights. Using the new weights, calculate the output of the network for the provided training example. Did the error decrease?

Handwritten Digit Recognition

4. *(35 points)*

In this programming exercise, you will use the Netlab toolkit (provided with the assignment) in MATLAB to create a neural network for recognizing handwritten digits from the popular MNIST dataset.

To get started,

- Download the dataset (four files total) from http://yann.lecun.com/exdb/mnist/.
- Extract the files to the directory of your choice.
- Run the following line of code in MATLAB:

```
[training,testing] = setupMNIST();
```

You should now see two structures in the MATLAB workspace called `training` and `testing`, containing 60,000 and 10,000 sample images of handwritten digits, respectively. The 28x28 grayscale images have been vectorized such that each column vector in the data arrays corresponds to the pixel intensities for a single example. Some example images are shown below.

To build the neural network, you will need the following three functions from Netlab:

- `mlp`: creates the neural network structure
- `netopt`: trains the network
- `mlpfwd`: tests the network

Use the help command to learn more about these function calls. Use 'logistic' for the activation function in `mlp` and 'scg' (scaled conjugate gradient) for the algorithm input in `netopt`. Before calling `netopt`, you should insert the following lines of code in your script and then use the *options* vector as an input in `netopt`:

```
options = zeros(1,18);
options(1) = 1; %display iteration values
options(14) = 500; %maximum number of training cycles (epochs)
```

You will create a 784-*n*-10 neural network, where *n* is the number of hidden units, which you should vary (e.g. 100, 500, 1000 may be good choices). You should also vary the number of training samples; in this case, 100, 1000, 10000, and 60000 may be good options, but feel free to explore others. For every set of parameters, implement some form of cross-validation (e.g. hold-out or *k*-fold). After training a network, test the network on the data and compute the training error, validation error, and test error. In all cases, the error is simply the average number of misclassifications; while the output of the network is a 10x1 vector, you need to convert this to a predicted digit (0-9).

(a) In your report, give a brief summary of your approach; remember, the idea is to make it clear to the graders what you did so that they can give you partial credit if something is wrong with your code.

(b) Save your best-performing network as a mat-file, with the network called *net*. Include this file in the 'code' directory of your submission. The purpose of this is so that the graders do not need to re-train your network, which can take a long time. In your report, list the parameters used in the best network as well as the error statistics.

(c) Discuss your results briefly (a paragraph should suffice). You should consider the effect, if any, of varying parameters (e.g. training samples, hidden nodes) on computational efficiency and accuracy. Also, explain the importance of each type of error (training, validation, test). Questions to consider include: How would you determine overfitting? How did you pick the best network?

(d) Show a few example images from the test set of correctly classified digits as well as misclassifications. In all cases, clearly specify the predicted digit based on the neural network output.

To recap, a correct implementation will include:

- use of Netlab functions (no need for MATLAB's Neural Network Toolbox)
- cross-validation
- varied number of hidden units
- varied number of training samples
- varied number of epochs (optional, but may be useful if your code is too slow or the error is too high)

- selection of the best network based on performance characteristics

Extra credit: *(10 points max)* Create an interface that allows a user to "sketch" a new test example, which is then classified by your trained neural network. The GUI needs to be intuitive to use and error-free in order to receive credit.