1 Submission instructions:

1. Practical Part

   (a) You are not allowed to use any machine learning packages or tools (e.g. scikit-learn, PyBrain, PyML, etc.).
   (b) You are allowed to use numpy package.
   (c) Use Python 2.7
   (d) Submit your solution via the [Submit] web interface.

Your files should include:

   i. A text file called details.txt with your full name (in the first line) and ID (in the second line).
   ii. logic.py: the file provided.
   iii. mlp.py: the file provided, after you modified it.

   Good Luck!

2 Questions:

2.1 Practical Part: Neural Networks

We will first describe the network you will use for this assignment. Without including biases, the network has an input layer of size $L$ (i.e., the size of the input data); a single hidden layer of size $M$; and an output layer of size $N$. That way, if we include the bias, there are $(L + 1) \times M$ weights between the input and the hidden layer, denoted by $v$, where the weight $v_{ij}$ connects
neuron $i$ to neuron $j$ of the following layer. There are $(M + 1) \times N$ weights between the hidden layer and the output layer, denoted by $w$, where the weight $w_{jk}$ connects neuron $j$ to neuron $k$ of the following layer.

Given a single example $x$ and its label $y$, the feedforward algorithm is as follows:

1. $h^{\text{hidden}}_j = \sum_{i=0}^{L} x_i v_{ij}$
2. $a^{\text{hidden}}_j = \sigma(h^{\text{hidden}}_j)$
3. $h^{\text{output}}_k = \sum_{j=0}^{M} a^{\text{hidden}}_j w_{jk}$
4. $\hat{y}_k = \sigma(h^{\text{output}}_k)$

where the superscript "hidden"/"output" indicates whether $h$ refers to a hidden or output neuron.

We will use the sum-of-squares error function, which calculates the difference between $y$ and $\hat{y}$ for each node $k$, squares them, and adds them all together:

$$E = \frac{1}{2} \sum_{k=1}^{N} (\hat{y}_k - y_k)^2$$

We will now derive the backpropagation update rules for the weights $w$ and $v$.

The update rule of the elements of $w$ is of the following form:

$$w_{jk} = w_{jk} - \eta \frac{\partial E}{\partial w_{jk}}$$

Using the chain rule, we get that:

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial h^{\text{output}}_k} \frac{\partial h^{\text{output}}_k}{\partial w_{jk}}$$

Starting with the last term first, we get that:

$$\frac{\partial h^{\text{output}}_k}{\partial w_{jk}} = \sum_{j=0}^{M} \frac{\partial a^{\text{hidden}}_j}{\partial w_{jk}}$$

$$= \sum_{j=0}^{M} \frac{\partial a^{\text{hidden}}_j}{\partial w_{jk}}$$

$$= a^{\text{hidden}}_j$$

We will now deal with the first part of $\frac{\partial E}{\partial w_{jk}}$ that we previously ignored. Denote:

$$\delta_o(k) = \frac{\partial E}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial h^{\text{output}}_k}$$
The first part:

$$\frac{\partial E}{\partial \hat{y}_k} = \frac{\partial}{\partial \hat{y}_k} \left[ \frac{1}{2} \sum_{k=1}^{N} (\hat{y}_k - y_k)^2 \right]$$

(5)

$$= \hat{y}_k - y_k$$

(6)

In the second part we use the derivative of the sigmoid function:

$$\frac{\partial \hat{y}_k}{\partial h^\text{output}_k} = \frac{\partial \sigma(h^\text{output}_k)}{\partial h^\text{output}_k}$$

(7)

$$= \sigma(h^\text{output}_k)(1 - \sigma(h^\text{output}_k))$$

(8)

$$= \hat{y}_k(1 - \hat{y}_k)$$

(9)

Thus, we get that:

$$\delta_o(k) = (\hat{y}_k - y_k)\hat{y}_k(1 - \hat{y}_k)$$

(10)

And our update rule is

$$w_{jk} = w_{jk} - \eta \delta_o(k) a^\text{hidden}_j$$

We will now continue and find the update rule for the first layer weights, \(v\). As before, the update rule is as follows:

$$v_{ij} = v_{ij} - \eta \frac{\partial E}{\partial v_{ij}}$$

When running backwards through the network, the value of \(\frac{\partial E}{\partial v_{ij}}\) is influenced by all \(N\) output nodes. That is, each hidden node contributes to the value \(h^\text{output}_k\) of all of the output nodes (i.e., for all values of \(k\)), so we need to consider all of the output nodes when updating the weight of a hidden node.

Therefore, if we start by only calculating \(\frac{\partial E}{\partial h^\text{hidden}_j}\), which we will denote by \(\delta_h(j)\), we get

$$\delta_h(j) = \frac{\partial E}{\partial h^\text{hidden}_j}$$

(11)

$$= \sum_{k=1}^{N} \frac{\partial E}{\partial \hat{y}_k} \frac{\partial \hat{y}_k}{\partial h^\text{output}_k} \frac{\partial h^\text{output}_k}{\partial h^\text{hidden}_j}$$

(12)

$$= \sum_{k=1}^{N} \delta_o(k) \frac{\partial h^\text{output}_k}{\partial h^\text{hidden}_j}$$

(13)

Calculating the last term, we get
\[
\frac{\partial h_k^{\text{output}}}{\partial h_j^{\text{hidden}}} = \frac{\partial h_k^{\text{output}}}{\partial a_j^{\text{hidden}}} \frac{\partial a_j^{\text{hidden}}}{\partial h_j^{\text{hidden}}} = \frac{\partial}{\partial h_j^{\text{hidden}}} \left[ \sum_{j=0}^{M} a_j^{\text{hidden}} w_{jk} \right] \frac{\partial}{\partial h_j^{\text{hidden}}} \left[ \sigma(h_j^{\text{hidden}}) \right] = w_{jk} \sigma(h_j^{\text{hidden}})(1 - \sigma(h_j^{\text{hidden}}))
\]  

(14)

(15)

(16)

Putting the values together, we get that

\[
\delta_h(j) = \sum_{k=1}^{N} \delta_o(k) w_{jk} \sigma(h_j^{\text{hidden}})(1 - \sigma(h_j^{\text{hidden}}))
\]

(17)

\[
= \sigma(h_j^{\text{hidden}})(1 - \sigma(h_j^{\text{hidden}})) \sum_{k=1}^{N} \delta_o(k) w_{jk}
\]

(18)

\[
= a_j^{\text{hidden}}(1 - a_j^{\text{hidden}}) \sum_{k=1}^{N} \delta_o(k) w_{jk}
\]

(19)

Thus, we calculated the value of \( \delta_h(j) = \frac{\partial E}{\partial h_j^{\text{hidden}}} \). The last thing left to finding \( \frac{\partial E}{\partial v_{ij}} \) is

\[
\frac{\partial h_j^{\text{hidden}}}{\partial v_{ij}} = \frac{\partial}{\partial v_{ij}} \left[ \sum_{i=0}^{L} x_i v_{ij} \right] = x_i
\]

(20)

(21)

Finally,

\[
\frac{\partial E}{\partial v_{ij}} = a_j^{\text{hidden}}(1 - a_j^{\text{hidden}}) \left( \sum_{k=1}^{N} \delta_o(k) w_{jk} \right) x_i
\]

(22)

which could be rewritten as

Thus, the update rule for \( v_{ij} \) is

\[
v_{ij} = v_{ij} - \eta \frac{\partial E}{\partial v_{ij}}
\]

(23)

\[
= v_{ij} - \eta a_j^{\text{hidden}}(1 - a_j^{\text{hidden}}) \left( \sum_{k=1}^{N} \delta_o(k) w_{jk} \right) x_i
\]

(24)

\[
= v_{ij} - \eta \delta_h(j) x_i
\]

(25)

To summarize the backpropagation algorithm:
1. \( \delta_o(k) = (\hat{y}_k - y_k) \hat{y}_k (1 - \hat{y}_k) \)

2. \( \delta_h(j) = a_j^{\text{hidden}} (1 - a_j^{\text{hidden}}) \sum_{k=1}^{N} \delta_o(k) w_{jk} \)

3. \( w_{jk} = w_{jk} - \eta \delta_o(k) a_j^{\text{hidden}} \)

4. \( v_{ij} = v_{ij} - \eta \delta_h(j) x_i \)

The dataset you will use for this assignment will be the AND and XOR datasets (both consisting of four examples). These datasets are very basic, and are used here for simplicity.

You are provided with a script called \texttt{mlp.py} which contains a general skeleton of a code for neural networks. Your task is to fill in the code for the feedforward and backpropagation algorithms, in the marked places.

The script \texttt{logic.py} contains the two datasets you will use, and trains a network using \texttt{mlp.py}. Use this script in order to run your neural network on the datasets and evaluate its performance.