CS 672 – Neural Networks – Fall 2010
Instructor: Marc Pomplun

Practice Exam – Sample Solutions

Duration: 2:30 hours

No calculators, no books, and no notes are allowed (in the actual exam).

Question 1: ____ out of ____ points
Question 2: ____ out of ____ points
Question 3: ____ out of ____ points
Question 4: ____ out of ____ points
Question 5: ____ out of ____ points
Question 6: ____ out of ____ points
Question 7: ____ out of ____ points (bonus question)

Total Score:

Grade:
**Question 1: Is it true?**

Tell whether each of the following statements is true or false by checking the appropriate box. Do not check any box if you do not know the right answer, because you will lose points for incorrect answers.

<table>
<thead>
<tr>
<th>Statement</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) In production mode, a CPN with eight hidden-layer neurons can create at most eight different output vectors.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>b) After training a Self-Organizing Map, output neurons that win for similar inputs are usually far apart from each other in the map.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>c) A single perceptron can compute the XOR function.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>d) The Hopfield Network has only a single layer of neurons.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>e) During the first phase of CPN training, it is a good idea to increase the step size parameter $\alpha$ with every epoch.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>f) CPNs can only compute linear functions.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>g) In production mode, an RBF network with five hidden-layer units can create at most five different output vectors.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
<tr>
<td>h) A single Threshold-Logic Unit can realize the AND function.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>i) The Rprop algorithm should only be used for “per-epoch” (batch mode) learning.</td>
<td>[X]</td>
<td>[ ]</td>
</tr>
<tr>
<td>j) If in a CPN we set $\alpha = 0$, the network will usually still learn the desired function.</td>
<td>[ ]</td>
<td>[X]</td>
</tr>
</tbody>
</table>
**Question 2: Do it Yourself**

Assume that you want to build a Counterpropagation network that computes the following function:

For input $\mathbf{x} = (1, 0)$ it outputs $\mathbf{y} = (0.5, 0.5)$,  
for input $\mathbf{x} = (0.6, 0.8)$ it outputs $\mathbf{y} = (1, 0)$, and  
for input $\mathbf{x} = (0, 1)$ it outputs $\mathbf{y} = (0, 0)$.

You decide to use two input neurons, three hidden-layer neurons, and two output neurons. Unfortunately, you forgot how to train a CPN, and so you have to set the weights yourself. In the diagram below, enter values for all the 12 weights in the CPN in the empty boxes so that the network computes the desired function.
Question 3: SOMs

a) What problem could occur in SOM learning if you use a very small neighborhood? Why?

Answer: It is possible that neurons in separate areas of the output layer become selective for the same type of input. Because there are no global interactions between neurons in the output layer, separate “islands” of similar selectivity can develop without ever “meeting” each other and merging.

b) SOMs can reduce the dimensionality of a given data space. Explain what that means. Please give an example of how this capability can be used for practical applications.

Answer: The map layer of an SOM is typically one- or two-dimensional, whereas the input space for the SOM usually has many more dimensions (784 dimensions in your Assignment #4). Nevertheless, the SOM tries as much as possible to establish a topology-conserving mapping such that inputs from neighboring regions in the input space will make neighboring neurons (or even the same neuron) in the map layer win the competition. In this way, the high-dimensional input space is mapped onto a lower-dimensional output space. One example is the visual representation of large data sets, for example, image or document databases. Using an SOM, such high-dimensional spaces can be mapped, for example, onto a two-dimensional space in which similar images or documents are usually close to each other. The user can browse through this space to find a desired item much more easily than through the original, high-dimensional space.

c) Why are SOMs interesting for researchers who study biological nervous systems?

The topology-conserving property of the SOM between its input and output spaces is in fact a very common feature of biological neural networks. For instance, neighboring areas in our sensory cortex respond when our ring finger or our middle finger on our right hand is being touched. However, the area responding to touch in our left foot is far away from the first two areas. The development of this mapping and its adaptation to changes (my favorite example: losing one of your fingers) is assumed to be accomplished by competitive mechanisms as in the SOM. SOMs have been used to predict the development of neural connections in our brain with good accuracy.

Question 4: Hopfield Networks

a) Hopfield networks are most often used for auto-association. What does that mean, and why can this be useful at all?

Auto-association is a network’s ability for a certain number of stored patterns, whenever they are input to the network, to produce the same pattern as its output. This is useful if the network is still able to output precisely the stored pattern even if the input pattern is
noisy, i.e., slightly deviates from the stored one. In this case, the network can be used to remove noise from known input patterns.

b) Why do we demand for auto-association that all $w_{ii} = 0$?

If we did not demand this, the trivial solution $w_{ii} = 1$ and $w_{ij} = 0$ for $i \neq j$ could be chosen, which formally leads to auto-association for all input patterns but does not store any pattern information and cannot be used for error correction. With $w_{ii} = 0$ we ensure that the pattern information (the correlation between pairs of pixels) is stored in the connections between neurons.

c) Please explain how it can be shown that for a constant input, Hopfield networks are guaranteed to reach a stable state after a finite number of iterations. You do not have to write down any equations, but just describe the concepts that are introduced and why they prove this property of Hopfield networks.

First, we introduce an energy function in such a way that it decreases whenever the state of the network gets closer to one of the stored patterns. Every possible state has a particular energy value. Then we show that whenever the Hopfield rule demands that the state of a neuron switches, the network energy decreases. Since in the discrete Hopfield network there is only a finite number of possible states, it means that after a finite number of switches a state will be reached in which no further switches can occur because none of them would lower the energy.

**Question 5: Interpolative Associative Memory**

Let us say that for some reason you need a network with the following characteristics:

For input $x = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$ it outputs $y = \begin{bmatrix} 2 & 3 \end{bmatrix}^T$, for input $x = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^T$ it outputs $y = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$, and for input $x = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ it outputs $y = \begin{bmatrix} 0 & 5 \end{bmatrix}^T$.

Since you want the network to interpolate, and since by a weird coincidence your exemplars already have orthonormal input vectors, you decide to use an Interpolative Associative Memory for this task.

(a) Build this network, i.e., draw its neurons and connections, and indicate the values of all weights in the network.

$$ W = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 5 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} $$
(b) Test the network’s interpolative ability by giving it the inputs \([0.5 \ 0.5 \ 0]^T\) and \([0 \ -1 \ -1]^T\) and determining its output in each case. If you did not build the network, just write down what you believe its output would have been, and explain why.

Input \([0.5 \ 0.5 \ 0]^T\) results in output \([0.5 \ 1.5]^T\).

Input \([0 \ -1 \ -1]^T\) results in output \([1 \ -5]^T\).

Question 6: Cascade Correlation

a) Describe the Cascade Correlation learning algorithm in your own words. You do not have to use any equations or pseudocode, but a diagram showing the progression of the algorithm would be useful.

Well, please just refer to the slides.

b) What advantages and disadvantages of this algorithm as compared to backpropagation can you think of?

There are two main advantages: First, the training is faster, because only one hidden-layer neuron or the output-layer neurons are trained at any given time. Second, no experimentation with the number of hidden-layer units is necessary; it determines the number of necessary hidden-layer units itself based on a specified error threshold.
A disadvantage is that the weights of hidden-layer units are frozen after their training, whereas BPNs allow all weights to adapt during the entire learning process. Therefore, Cascade Correlation is less likely to find optimal weights for a given task.

**Question 7 (Bonus Question):**

a) Describe in precise English words or in mathematical equations the condition under which an ART-1 network “grows” a new neuron.

- A new (not previously presented) input is given to the network.
- For none of the network’s current output neurons, the similarity between that neuron’s top-down weights and the input reaches the value of the vigilance parameter.

b) What is the effect of increasing or decreasing the vigilance parameter in an ART-1 network?

A greater vigilance parameter makes the network less “tolerant,” i.e., data points in only a small area of the input space can belong to the same category (prototype). In the extreme case ($\rho = 1$), each distinct input forms a category by itself. Smaller values, on the other hand, will lead to fewer, broader categories.

c) What is the stability-plasticity dilemma and how does the ART-1 network attempt to solve it?

The stability-plasticity dilemma refers to the problem that NNs need to be plastic (flexible) in order to learn new data. At the same time, we do not want them to forget data that they learned before.

ART networks can always adapt to unknown inputs (by creating a new cluster with a new weight vector) if the given input cannot be classified by existing clusters. Existing clusters are not deleted by the introduction of new inputs, because new clusters will just be created in addition to the old ones.