NAME________________________
login:________________________
Signature: ________________________

Computer Science and Engineering 150
Programming Languages for Artificial Intelligence
March 16, 2000
SECOND MIDTERM EXAM
DO NOT TURN THIS PAGE UNTIL YOU ARE TOLD TO START!!!!

- Please DO NOT put your name at the top of each page:
  - This should prevent residual sexism, racism, favoritism lurking in the unconscious of your professor & TA from biasing grading!!
- THE EXAM IS CLOSED BOOK. IN FACT, PLEASE LEAVE YOUR BOOKS AT THE FRONT OF THE CLASS!
- Once the exam has started, SORRY, NO TALKING!!!
- No, you can't even say "see ya at Porter's!"
- There are 6 problems: Make sure you have all of them - AFTER I TELL YOU TO START!
- Read each question carefully.
- Remain calm at all times!

Problem	Type	Points	Score
1	True/False	15	
2	Fill in the blanks	15	
3	Multiple Choice	18	
4	Representation: Perceptrons	15	
5	Learning: Perceptrons	10	
6	Learning: Back propagation	12	
E	Extra credit	10	
	Total Total with extra credit	85 95	

True/False

1. (15 pts: +1 for correct, -1 for incorrect, 0 for no answer)
   If you would like to justify an answer, feel free.

______ One of the "tricks of the trade" described in class is to use a very large learning rate in order to find the answer quickly.

______ One of the "tricks of the trade" described in class was to use a particular sigmoid that had the property that it would output a negative number for negative input.

______ You can decide what a unit in a neural network "likes" by looking at which units it sends large activations to through large weights.

______ The reason why the linear threshold function is not used with back prop is that it is not differentiable everywhere.

______ The expression recognition network described in class did not require a nonlinear hidden layer in order to classify expressions.

______ To improve generalization, it is sometimes useful to minimize not only the error, but a measure of network complexity as well.

______ The perceptron learning algorithm always converges.

______ The error in an autoencoder network can be used to determine the *novelty* of an input to the network, and this idea was used in the system that drove a Chevy Van.

______ A cognitive model should be used to explore new areas in psychology, where there is not much data.

______ One way to construct a network for any boolean function is to have a hidden unit for every input pattern, that responds only to that pattern.

______ In the text, a system was described that learned to understand speech, using Time Delay Neural Networks (TDNN's).

______ Back propagation, because it is a powerful learning technique, will always find the *global* minimum in the error function.

______ The Russell and Norvig text describes (briefly) Hopfield networks and Boltzmann machines, both recurrent networks, before launching into their description of feed forward networks.

______ Backpropagation is a technique that uses the derivative of the output with respect to the inputs in order to change the weights in the network.

______ Neural networks were invented to torture computer science students who thought that they didn't *really* need to know any of that calculus stuff they learned.

Fill in the blanks/short answer (15pts)

2(a). (5 pts total) Sketch an autoencoder network (sometimes used as an image compression network) (3 pts):

Describe how this was used in the Chevy Van driving system (2pts).

(1 pt each blank):

2(b). Back propagation networks learn by performing ____________ ____________ in weight space.

2(c). The Perceptron Learning Rule described intuitively in class was later shown to be described equally simply mathematically as Δw = ________________________. (Assume a learning rate of 1.0, and use the letters i for input, o for output, and t for teacher).

2(d). The reason most people use online back propagation instead of batch training for neural networks is that many real-world data sets are highly ____________ and so an online algorithm may actually learn the whole problem before the end of a single epoch.

2(e). The ____________ in an artificial neural network correspond to the synaptic strengths in a "real" neural network.

2(f). A perceptron is only able to learn to classify patterns that are ____________ separable, since it is only those kinds of problems that it can represent.

2(g). Back propagation networks can do two kinds of tasks, classification and regression. Expression recognition is an example of a ____________ task.

2(h). The essence of the proof of convergence of the perceptron algorithm was that the projection of the weight vector onto a given vector that already was assumed to ____________ the problem would ____________ without bounds if the weights continued to be updated. This is clearly impossible, and hence the number of updates to the weights will be ____________.

Multiple Choice (18 points) (circle the correct answer)

3(a). Backpropagation is derived by computing the partial derivative of

a) the weights with respect to the activations.
b) the sigmoid, which turns out to be out*(1-out).
c) the weights with respect to the error
d) gabor filter outputs
e) the error with respect to the weights.

3(b). The back propagation algorithm works by

a) propagating the deltas backwards
b) computing the error at the output units
c) changing the weights
d) propagating the activations forward.
e) (b), then (c), then (d)
f) (d), then (b), then (a), then (c)
g) (d), then (b), then (c), then (a)
h) one of the above, I'm not sure which.

3(c). The appearance of the slope term in the computation of the δ's means that

a) the rise over the run is coming back to haunt me.
b) learning gets very slow when net inputs to unit are large
c) learning is fastest near the extremes of the sigmoid
d) learning xor is best when you have linear units, since the slope is 1.0
e) the error will always get bigger the closer the layer is to the input in a network with many hidden layers.

3(d). perceptron learning can be thought of as

(a) adding and subtracting incorrect patterns to/from the weights
(b) learning on every pattern
(c) when the output is ON and it should be OFF, raising the weights to the active input lines.
(d) (a) and (c)

3(e). The form of one rule for learning (I'm not saying which one) is:

(a) $net_i \;=\; \sum_j w_{ij} * output_j$
(b) $\Delta w_{ij} \;=\; \eta * activation_i * output_j$
(c) $\Delta output_i \;=\; (teacher - output_i) * output_j$
(d) $\Delta w_{ij} \;=\; (teacher - w_i) * input_j$
(e) $\Delta w_{ij} \;=\; (output_i) * (1 - output_i) sum_k w_{ki} * \delta_k * output_j$

3(f). It is not surprising that the expression network's weights to the "Happy" node model Happy faces since

a) The network has no hidden layer, so in this case, we can think of the network as essentially adding Happy patterns to the weights and subtracting the other patterns.
b) Sneezy, Doc and Dopey were not in the training set.
c) the patterns the Happy node "likes" will be represented in the activations of the unit.
d) Surprised faces are so similar to Happy faces in the Multidimensional scaling.

3(g). Two ways of getting good generalization in neural networks are

a) using a regularizing term in the objective function and using weight decay.

b) Using a hold out set to change the weights and using a lot of training data.

c) Using as many hidden units as possible, and making the network have cross-validated activation patterns.

d) using a hold out set to stop training and using a regularizing term in the objective function.

3(h). The teaching signal in autoencoder networks is usually:

(a) The class of the input, e.g., happy, sad, surprised...

(b) The input

(c) The principal components of the input

(d) The function of the input that is to be learned, e.g., x squared.

3(i). Principal Components Analysis is:

(a) a rotation and translation of the coordinate system.

(b) a way of reducing the dimensionality of the data, by throwing away eigenvectors with little variance.

(c) a linear algebra technique that finds the eigenectors of the covariance matrix of the data.

(d) all of the above.

(e) (b) and (c) only.

Representation: Perceptrons (15 points)

4a. Sketch below a set of points in 2 dimensions that are linearly separable into 2 classes, using "+"'s for one class and "-"'s for the other.

4b. Using the same scheme, sketch a set of points that are *not* linearly separable.

4c. Using the linear threshold activation rule:

```
if (net_input >= threshold)  then 1
                             else 0
```

draw a perceptron that takes two inputs, with the weights on the links and the threshold in the unit, that will ONLY turn on for the pattern "0 1", and NOT turn on for the patterns "0 0", "1 0" or "1 1".

Learning: Perceptrons (10 points)

5. The following is a table describing the learning in a perceptron like we did in class. It differs from the one we did in class in that it uses a bias (as you did instead in your programming assignment) instead of a threshold.

Fill in the next 4 lines of the table, assuming the activation rule is:

```
if net_input (including bias) >= 0.0  then 1
                                       else 0
```

(you have to remember the learning rule, assume a learning rate of 1.0, as we did in the programming assignment. Also, as we did in class, the weights on a row correspond to the *current weights* , with the *new weights* on the next row.) Recall that the bias is a weight from an input that is always 1.0.

Input1	Input2	Input3	Weight1	Weight2	Weight3	Bias	Output	Teacher
1	1	0	0	0	0	0	1	0
0	1	0	-1	-1	0	-1	0	0
0	1	1	-1	-1	0	-1	0	1
0	0	1	-1	0	1	0	1	0
1	1	1						0
0	1	1						1
0	1	0						0
-	-	-						-

Learning: Back propagation (12 points)

6(a) (9 pts). Compute the δ's for the output unit and the hidden units after the following pattern presentation. The network structure is at the left (recall δ for an output is (t-o) times the slope, and assume standard sigmoids). The second-from-left column shows the network weights, with the biases shown at the nodes. The middle column shows the net input to a node. The next column shows the activations of the inputs, hiddens and outputs. Your job: Fill in the last column.

```
                                                                    teacher = 1.0
             |   weights/biases   |    net_input    |   activation   |     delta      |
             |                    |                 |                |                |
             |                    |                 |                |                |
    OUT      |        -.18        |      -.13       |      .46       |  OUT:___       |
    / \      |        /   \       |                 |                |                |
   /   \     |      .57   -.10    |                 |                |                |
  /     \    |      /        \    |                 |                |                |
H1       H2  |   -.81      -.02   |  -1.64    -.27  |  .16      .43  |H1:___  H2:___  |
| \     / |  |    | \       / |   |                 |                |                |
|  \   /  |  | .58 -.82 -.07 .24  |                 |                |                |
|    X    |  |    |     X     |   |                 |                |                |
|   / \   |  |    |    / \    |   |                 |                |                |
IN1    IN2   |                    |                 |   0      1.0   |                |
```

6(b) (3pts). Finally, given the δ's you computed, what will the new weight from hidden unit 1 (H1) to the Output (OUT) (currently 0.57) be, if no momentum is being used and the learning rate is 1.0?

Extra Credit: 10 points

These questions might be considered “harder” than the previous ones.

E1: Why is it a good idea to normalize the inputs to a network so that they have zero mean and unit variance?

E2: Draw the theoretical graph of error over training showing: 1) The error on the training set and 2) the hypothesized error on a test set. Show where it would be ideal to stop training.

E3: Why should weights in a multi layer network be initialized to small random values? What would happen if all the weights started out at the same value?

E4: Prove that **XOR** cannot be computed by a single-layer perceptron.