

Fall 1998 exam was a 90 minute exam.

(24)
I. LISP Programming. You cannot use PROGS or any LOOP/DO constructs.
An $N \times N$ complex matrix R is represented by N lists of N pairs, where the first element of each pair is the real part of the matrix component and the second element of each pair is the imaginary part of the corresponding matrix component. Give a recursive LISP function or use a mapping function to compute the Hermitian, i.e., the conjugate transpose matrix R^t

$$\begin{vmatrix} (1+j2) & (3+j4) \\ (5+j6) & (7+j8) \end{vmatrix} \rightarrow \begin{vmatrix} (1-j2) & (5-j6) \\ (3-j4) & (7-j8) \end{vmatrix}^t$$

Lisp→ (CONJ-TRANSPOSE '(((1 2) (3 4)) ((5 6) (7 8))))
(((1 -2) (5 -6)) ((3 -4) (7 -8)))

Lisp→ (CONJ-TRANSPOSE '(((1 2) (3 4) (5 6)) ((7 8) (9 10) (11 12)) ((13 14) (15 16) (17 18))))
(((1 -2) (7 -8) (13 -14)) ((3 -4) (9 -10) (15 -16)) ((5 -6) (11 -12) (17 -18)))

Hint: You may wish to write a recursive LISP function or use mapping functions to transpose an $N \times N$ matrix represented as N lists of N integers each (and receive partial credit for your effort), e.g.,

Lisp→ (TRANSPOSE '((1 2 3) (4 5 6) (7 8 9)))
((1 4 7) (2 5 8) (3 6 9))

Lisp→ (TRANSPOSE '((1 2) (3 4) (5 6)))
((1 3 5) (2 4 6))

Fall 1998 exam was a 90 minute exam.

(24)

II. LISP Programming. Choose between A or B below.

A. Write a function that inverts and consolidates an association list (a list of pairs or a list of dotted pairs, your choice!). For example,

```
Lisp→(INVERT-ALIST '( (McGriff . WR) (Jackson . RB) (Johnson . QB) (Gillespie . RB) (Taylor . WR) (Karim . WR)))  
((RB GILLESPIE JACKSON) (WR KARIM TAYLOR MCGRIFF) (QB JOHNSON))
```

```
Lisp→(INVERT-ALIST '( (McGriff WR) (Jackson RB) (Johnson QB) (Gillespie RB) (Taylor WR) (Karim WR)))  
( (RB Gillespie Jackson)  
  (WR Karim Taylor McGriff)  
  (QB Johnson))
```

Suppose:	(SETF X '(D E F))
Hint:	(RPLACD X '(A B C)) → (D A B C)
or	(SETF (cdr X) '(A B C)) changes X to (D A B C)

B. Write a recursive function that finds the minimum number that occurs at any depth in a list. For example,

```
Lisp→ (MIN-NO '((A 3) (B (C 1.5)) 3.2))  
1.5
```

Fall 1998 exam was a 90 minute exam.

(24)

III. LISP Programming. Do *both* (a) and (b) below.

(12) a. MAPC is like MAPCAR in that it hands to its functional argument (funarg) successive cars of the input list. MAPC differs from MAPCAR in that MAPC prematurely stops applying the funarg to the successive cars of the input list if the value returned by the argument function is NIL. Further, MAPC returns NIL if it was stopped or T if it was allowed to exhaust the input list. This is useful because many of the functions we have defined in LISP should be stopped when the result is obvious and there is no need to traverse the input list until it is exhausted. Give a definition of MAPC.

```
(defun mapcar (f lis) (cond
  ( (null lis) nil )
  ( t (cons (funcall f (car lis)) (mapcar f (cdr lis))) )))
```

(12) b. Define EQSET, a predicate that determines whether two sets are the same set using the MAPC mapping function.

```
Lisp> (eqset '(a b c) '(b a) ) ==> Nil
Lisp> (eqset '(a b) '(a b c) ) ==> Nil
Lisp> (eqset '(a b c) '(b a c) ) ==> T
Lisp> (eqset '(coke is it) '(is it coke) ) ==> T
Lisp> (eqset '(inhuman acts are human mistakes)
            '(human acts are inhuman mistakes) ) ==> T
```

Fall 1997 exam was a 60 minute exam...

(25)

LISP Programming

IV. Write a recursive function NTHCAR that returns the first N elements of a list. For example,

```
Lisp> (nthcar 3 '(a b c e d f))
```

```
(A B C)
```

```
Lisp> (nthcar 2 '(a))
```

```
(A)
```

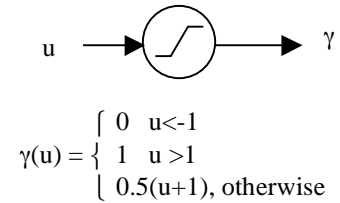
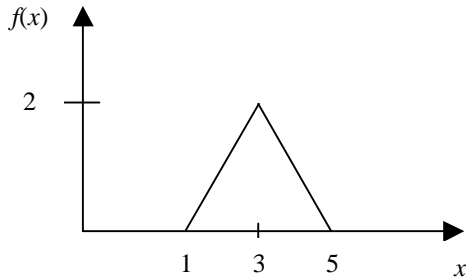
```
Lisp> (nthcar 0 '(b))
```

```
NIL
```

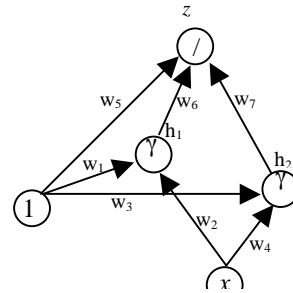
Fall 1999 was a 60-min exam

IV. Artificial Neural Networks

A single input, single output, fully-connected 1-hidden layer, 2 hidden unit ANN is used to approximate the function f below. Find the required weight vector if the ANN uses the "ramp" non-linear activation function below for its hidden units.



Requirements (for partial credit purposes): (a) Give an analytic expression for $f(x)$; (b) Draw the network assuming a linear unit on the output z ; (3) Find the weight vector $\{w_i\}$; (4) verify that your design works. Failure to follow these guidelines may result in a lower than expected score on this problem.



Fall 1995 exam was a 90 minute exam

LISP Programming

(12)

1a. Write a recursive or non-recursive function that transposes an nxm matrix represented as n lists of m characters each. You cannot use PROGS or any LOOP/DO construct. For example,

Lisp→ (transpose '((a b c) (d e f) (g h i)))

((a d g) (b e h) (c f i))

Lisp→ (transpose '((1 2) (3 4) (5 6)))

((1 3 5) (2 4 6))

(12)

1b. Describe the evident purpose of the procedure by tracing the output. Show Your Work! Note No Work = No Credit.

```
(defun enigma (n)
  (if (null n)
      #'(lambda (lis) lis)
      #'(lambda (lis) (funcall (enigma (cdr n)) (cdr lis))
    ))
  (defun exotic (block lis) (funcall (enigma block) lis) )
```

Lisp→(exotic '(x x x) '(a b c d e))

Answer: _____

Recursive LISP Programming

(12)

2a. Write the function FILTER of two argument strings S1 and S2 represented as lists where S2 is a pattern against which the expression S1 is to be matched; they are compared character by character from left to right, but S2 may contain the special characters +, * which have the following effects:

- if + is encountered in S2 any character in the corresponding position in S1 is accepted
- if * is encountered in S2 any number of characters in S1 are accepted

Lisp→(FILTER '(A+C) '(A B C))

T

Lisp→(FILTER '(* C) '(A B C))

T

Lisp→(FILTER '(* B K + M * Z) '(B A O B A B K L M Y Z))

T

(12)

2b. **MAP2** is similar to **MAPCAR**, except that it accepts functions of two arguments and applies them to list components in pairs. Give a recursive definition for **MAP2**.

Lisp→ (map2 #'cons '(a (b c) (x y) (z) m ((n) o)))

((a b c) ((x y) z) (m (n) o))

(24)

I. LISP Programming. You cannot use PROGS or any LOOP/DO constructs.

- a. Write a LISP function to take the dot product of two vectors each stored as LISP lists. The function returns the LISP atom ERROR! if the vectors do not have the same number of elements.

```
Lisp→ (DOT_PRODUCT '(1 2 3 4) '(1 2 3 4))
```

```
30
```

```
Lisp→ (DOT_PRODUCT '(3 4 5) '(5 4 3))
```

```
46
```

```
Lisp→ (DOT_PRODUCT '(1 1) '(1))
```

```
ERROR!
```

- b. Write a LISP function which uses the DOT_PRODUCT function from part a above to multiply the $n \times m$ matrix A by the $m \times k$ matrix B, where an $n \times m$ matrix is represented internally as n lists of m characters each. Assume you have available the function TRANSPOSE which transposes a matrix represented this way. (Hint: You may wish to consider using MAPping function(s)). For example,

```
Lisp→ (TRANSPOSE '( (1 2) (3 4) (5 6) ))
```

```
((1 3 5) (2 4 6))
```

```
Lisp→ (MATRIX_PRODUCT '((1 2 3) (4 5 6)) '((1 1) (1 -1) (-1 1)))
```

```
((0 2) (3 5))
```

```
Lisp→ (MATRIX_PRODUCT '((1 1) (1 -1) (-1 1)) '((1 2 3) (4 5 6)))
```

```
((5 7 9) (-3 -3 -3) (3 3 3))
```

- c. For extra credit, write the function TRANSPOSE from part b above.

(24)

I. LISP Programming. You cannot use PROG's or any LOOP/DO constructs.

- (10) a. Write the function MAPPENDCAR, which is like MAPCAR, except that it uses APPEND instead of CONS to construct the list of results.

```
Lisp->> (DEFUN INCR(X) (LIST (+ X 1)))  
Lisp->> (MAPPENDCAR #'INCR '(1 2 3))  
Lisp-> (2 3 4)
```

- (4) b. Give the resulting symbolic expression(s) for the following LISP function evaluations. Show work for partial credit.

```
Lisp-> (MAPCAR #'INCR '(4 5 6))
```

 RESULT: _____

```
Lisp-> (MAPPENDCAR #'INCR '(4 5 6))
```

 RESULT: _____

- (10) c. Using the function(s) MAPCAR or MAPPENDCAR give a definition for SETDIFF which when given two sets as input, returns all elements in the first set which are not in the second set.

```
Lisp-> (SETDIFF '(A B C D) '(B A))  
(C D)  
Lisp-> (SETDIFF '(A B C D) '(D E F))  
(A B C)  
Lisp-> (SETDIFF '(A B C) '(D E F))  
(A B C)
```

(35)

IV. Consider a one-hidden layer feedforward neural network, fully connected between layers, with 2 inputs, 2 hidden units in the first layer, and 1 output. Assume sigmoidal activation functions .

a. How many total independent weights are contained in this neural network? Show Work! No Work, No Credit.

Answer: ____ independent weights.

b. *Without any local variables*, how many additions, multiplications, and function evaluations (γ, γ') are required to compute the output of the neural network in terms of the inputs and the weights of the neural network? Show Work! No Work, No Credit.

Additions: _____ Multiplications: _____ Function Evaluations: _____

c. Using the backpropagation algorithm, how much total computation is required [additions, multiplications and function evaluations (γ, γ')] to compute the error derivatives with respect to all the weights in the neural network, given a single training pattern $\langle x_1, x_2, y \rangle$ and the error measure $E = \frac{1}{2}(y-z)^2$? Show Work! No Work, No Credit.

Forward Pass:

Additions: _____ Multiplications: _____ Function Evaluations: _____

Backward Pass:

Additions: _____ Multiplications: _____ Function Evaluations: _____

Total Computation:

Additions: _____ Multiplications: _____ Function Evaluations: _____

(35)

I. LISP Programming. You cannot use PROG or any LOOP/DO constructs.

- (10) a. Given the following association lists of adjective and noun opposites write a function FLIP to return the opposites of a simple two atom list.

```
Lisp→ ( SETF *ADJ-OPP* '((BIG LITTLE) (LITTLE BIG) (GOD BAD) (BAD GOOD) (HOT COOL) (COOL HOT)) )  
((BIG LITTLE) (LITTLE BIG) (GOD BAD) (BAD GOOD) (HOT COOL) (COOL HOT))  
Lisp→ ( SETF *NOUN-OPP* '((BROTHER SISTER) (SISTER BROTHER) (DOG CAT) (CAT DOG) (APPLE ORANGE) (ORANGE APPLE)) )  
((BROTHER SISTER) (SISTER BROTHER) (DOG CAT) (CAT DOG) (APPLE ORANGE) (ORANGE APPLE))  
Lisp→ ( FLIP '(BIG SISTER) )  
(LITTLE BROTHER)  
Lisp→ ( FLIP '(BAD APPLE) )  
(GOOD ORANGE)  
Lisp→ ( FLIP '(HOT DOG) )  
(COOL CAT)
```

- (15) b. Write a function that evaluates expressions written in infix notation, containing only Lisp functions that can take two arguments. Assume the expressions contain only numbers and defined XLISP functions. In this version dispense with type error checking completely. Example expressions and the values they should yield are:

```
Lisp→ (INFIX-EVAL '(3 + 4))  
7  
Lisp→ (INFIX-EVAL '(3 * 4))  
12  
Lisp→ (INFIX-EVAL '(3 + (4 * 5)))  
23  
Lisp→ (INFIX-EVAL '((4 * 5) - (3 + (2 * 4))))  
9
```

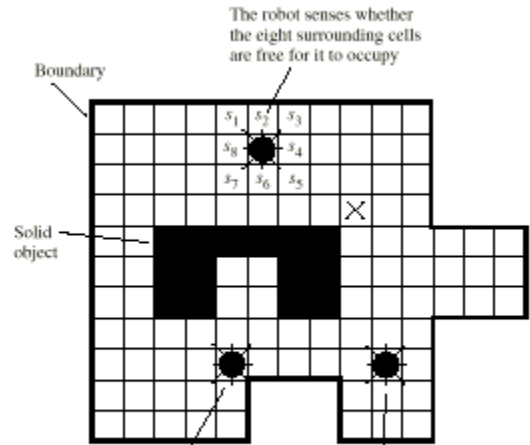
- (10) c. Using the built-in function MAPCAN, which is like MAPCAR except it uses APPEND instead of CONS, give a definition for a function to filter one or more lists, picking out only entries that satisfy some test (e.g., numbers). For extra credit (5) pass the test function as an argument to the filter function.

```
Lisp→ (FILTER '(A 3 B 2 4 C 7))  
(3 2 4 7)
```

(20)

III. SR Agents

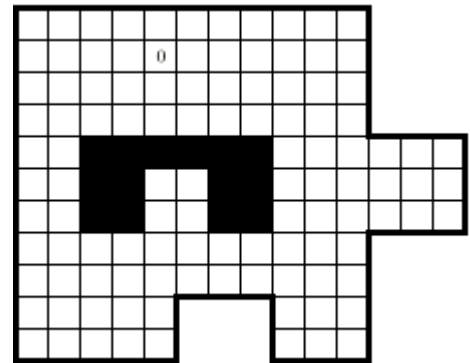
In a boundary following robot the sensory input vector s is given by the eight sensory inputs $\{s_1, s_2, s_3, \dots, s_8\}$. Define a feature vector x and give the production system rules for *left* wall-following. Simulate your system in the figure below starting at the position marked with 0. Give the details for the first four moves and at least one corner move. **NO DETAILS NO CREDIT. JUST DRAWING THE PATH WILL RESULT IN ZERO CREDIT!**



(5) Define the feature vector x .

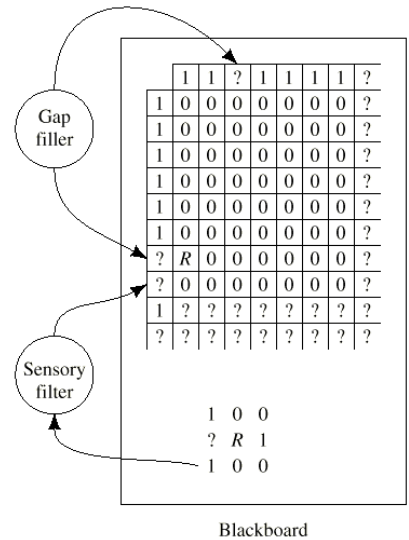
(5) Give the production system rules.

(10) Give details for the first four moves and one corner move and draw the path starting at point 0.



IV. Blackboard Systems

Our robot in the grid world can sense all eight cells immediately surrounding it. However, this robot's sensors (like all real sensors) sometimes give erroneous information. The robot also keeps a partial map of its world as in the accompanying figure at time τ . Because of previous sensor errors, the map can be incomplete and incorrect. The data structure representing the map and a data structure containing sensory data compose the blackboard at time τ . To help the robot correct blackboard errors, it has two knowledge sources (KSs), namely, a *gap filler* and a *sensory filter*. The gap filler looks for tight spaces in the map and either fills them in with 1's or expands them with additional adjacent 0's. Show Work/Explanation! No Work/Explanation, No Credit!



- (4) a. Assuming the gap filler works for 1 time step at time τ give the resulting blackboard at time step $\tau+1$.

- (5) b. Assuming the sensory filter works for 1 time step at time $\tau+1$ give the resulting blackboard at time step $\tau+2$.

- (4) c. Assuming the gap filler works for 1 time step at time $\tau+2$ give the resulting blackboard at time step $\tau+3$.

(39)

I. LISP Programming. You cannot use PROGs or any LOOP/DO constructs.

- (15) a. Write the function SUBST-SPLICE, which is like SUBST, except that it “splices in” its first argument for the second.

Lisp→ (SUBST-SPLICE '(1 2) 'b '(a b c))

(a 1 2 c)

Lisp→ (SUBST-SPLICE '(1 2) 'b '(a (b c) d))

(a (1 2 c) d)

- (10) b. Give the resulting symbolic expression(s) for the following LISP function evaluations. Show work for partial credit.

Lisp→ (FUNCALL #'(LAMBDA (N) (** 2 N)) '(4 6 8))

RESULT: _____

Lisp→ (MAPCAR #'(LAMBDA (SX) (LIST (QUOTE →) SX)) '(GATORS ARE NUMBER 1))

RESULT: _____

- (14) c. Using the built-in function MAPCAN, which is like MAPCAR except it uses APPEND instead of CONS, give a definition for INTERSECT which when given two sets as input, returns all elements in the first set which are also in the second set, in order.

Lisp→ (INTERSECT '(A B C D) '(B A))

(A B)

Lisp→ (INTERSECT '(A B C D) '(D E F))

(D)

Lisp→ (INTERSECT '(A B C) '(D E F))

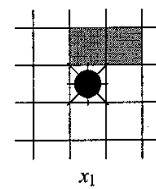
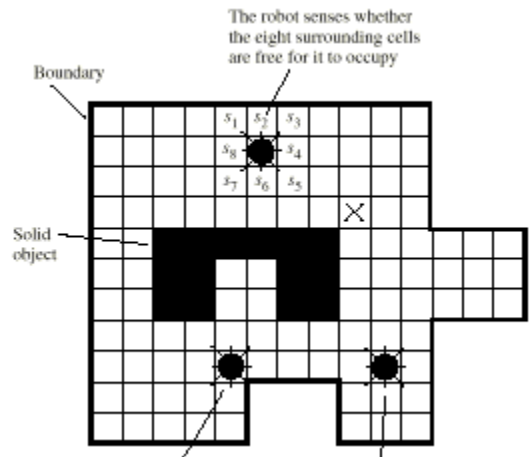
NIL

II. SR Agents

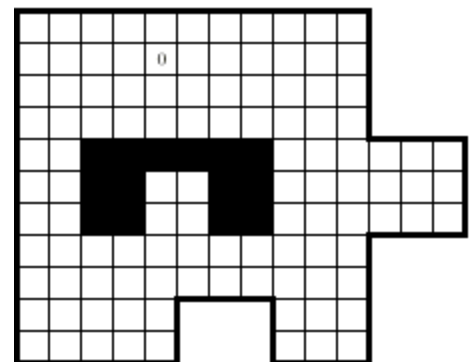
In chapter 2 of the Nilsson textbook, for the boundary-following robot, the sensory input vector s was given by the eight sensory inputs $\{s_1, s_2, s_3, \dots, s_8\}$. Suppose the robot has a "new" sensor that can only sense the cells $\{s_2, s_4, s_6, s_8\}$. Recall $s_i=1$ if the cell is not free.

(5)a. Is it possible to still do wall-following with this new sensor? Why or why not? Explain fully.

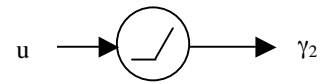
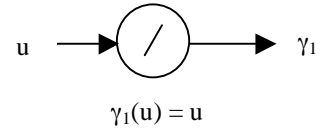
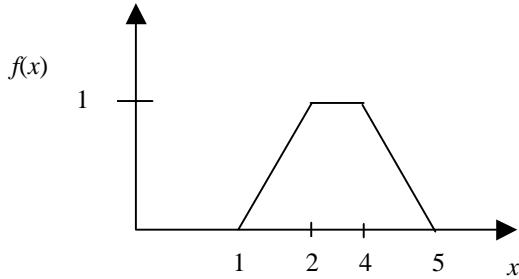
(10)b. Define the feature vector $\{x_i\}$ corresponding to $i=\{2,4,6,8\}$ and give a possible production system representation for a wall-following behavior. For example, the feature vector x_1 from chapter 2 is shown on the right.



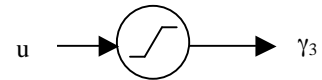
(10)c. Simulate your system in the figure on the right starting at the position marked with 0. Make sure you give me enough details to grade, i.e., just drawing a path is not sufficient. You must give the details of how you arrived at each position. Otherwise you will lose 50% or more of the points available!



III. A single input x , single output z , fully-connected 1-hidden layer, 2 hidden unit ANN is used to approximate the function $f(x)$ below. Requirements: (1) Draw the complete ANN network assuming a single linear output unit for z ; (2) Give the analytic expression for z in terms of x , γ and the weight vector $\{w_i\}$; (3) Which of the activation functions $\gamma_i(u)$ below {if any} should be used to estimate $f(x)$ and how well do you think the network can be expected to perform; (4) Find the weight vector $\{w_i\}$ for your choice of $\gamma_i(u)$ above, and (5) Verify that your design works.



$$\gamma_2(u) = \begin{cases} 0 & u < 0 \\ u, & \text{otherwise} \end{cases}$$



$$\gamma_3(u) = \begin{cases} 0 & u < 0 \\ 1 & u > 1 \\ u, & \text{otherwise} \end{cases}$$

I. Recursive LISP Programming. You cannot use PROG's or any LOOP/DO constructs.

- (10) Matrices and vectors are conveniently represented using arrays. When they get large, this can become unwieldy. In the case of sparse matrices/vectors storage is wasted on the large number of 0 entries, which also lead to useless arithmetic operations. Adding 0 and multiplying by 0 can be avoided if just the non-zero elements are stored. A sparse vector can be represented conveniently as a list of two-element sub-lists. Each sub-list contains an index and the corresponding component. The vector $[1.2, 0, 3.4, 0, 0, -6.7, 0]^T$ can be represented by the list ((1 1.2) (3 3.4) (6 -6.7)).

a. Develop a procedure to multiply a sparse vector by a scalar.

Lisp→ (SCALAR-BY-SPARSE-V 2 '((1 1.2) (3 3.4) (6 -6.7)))
((1 2.4) (3 6.8) (6 -13.4))

- (15) b. Develop a procedure to calculate the dot product of two sparse vectors.

Lisp→ (SPARSE-DOT-PRODUCT '((1 2) (3 3) (6 4)) '((1 1) (6 3)))
14

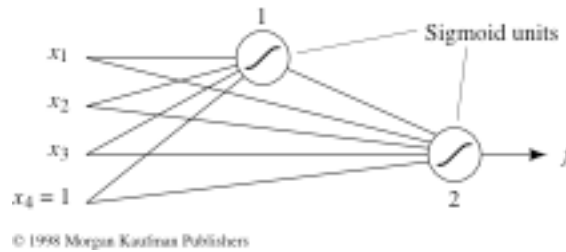
- (15) c. Develop a procedure to add two sparse vectors.

Lisp→ (SPARSE-V-PLUS '((1 2) (3 3) (6 4)) '((1 1) (6 3)))
((1 3) (3 3) (6 7))

(30)

II. Artificial Neural Networks

- (5) a. For the two-layer ANN network shown below derive an expression for the output function f in terms of the input vector \mathbf{x} , the weight vector \mathbf{w} , and the sigmoid function \mathbf{u} .



- (15) b. If an input pattern is given by (x,y) and the error function is given by $E=0.5(y-f)^2$ give an expression for the Gradient of the Error Function ($\partial E/\partial \mathbf{w}$).

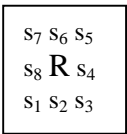
- (10) c. Suppose you have 5 training patterns $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5\}$, which yield 5 corresponding outputs $y=\{y_1, y_2, y_3, y_4, y_5\}$. Explain (give a procedure) how would you use this data to train the ANN (i.e., How would you use this data to obtain estimates of the unknown weight vector \mathbf{w})? Your answer should give an appropriate explanation and equation(s) to obtain the weight vector \mathbf{w} , and discuss any issues that might influence the process.

Fall 2003

(30)

III. SR Agents

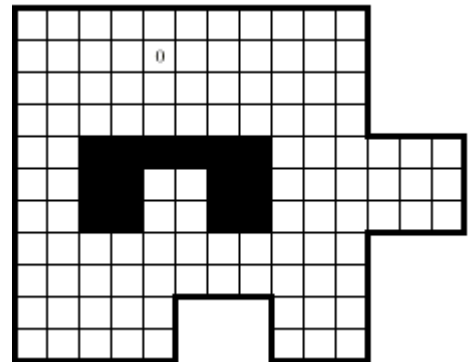
In a boundary following robot the sensory input vector s is given by the eight sensory inputs $\{s_1, s_2, s_3, \dots, s_8\}$. Define a feature vector x and give the production system rules to go *south* until it meets a boundary and then the robot moves to its *left* to do boundary-following. Simulate your system in the figure below starting at the position marked with 0. Give the details for the first seven moves (which includes at least one corner move.) NO DETAILS NO CREDIT. JUST DRAWING THE PATH WILL RESULT IN ZERO CREDIT!



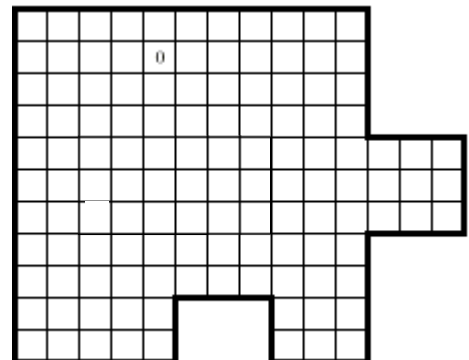
(8) a. Define the feature vector x . {Draw the feature and give the corresponding equation}

(5) b. Give the production system rules (with respect to the robot front or up is south).

(11) c. Give details for the first ten moves (including one corner move) and draw the path starting at point 0. As $t \rightarrow \infty$, what does the robot do? Explain.



(6) d. Suppose you want to use this robot to avoid the same obstruction shown in the figure in part c but continue south to the outer boundary (the fence) starting from the same (indicated by o) original position. What would have to be changed (if anything) in the figure or robot in order to accomplish the task? Draw the obstruction in the figure and explain in detail.



(40)

I. Recursive LISP Programming. You cannot use PROGS or any LOOP/DO constructs.

- (20) a. The covariance of two lists,
- X
- and
- Y
- , of the same length, is defined as

$$(n-1)^{-1} \times \{ \sum X_i Y_i / n - (\sum X_i / n)(\sum Y_i / n) \}$$

where

 X_i is the i th element of list X , Y_i is the i th element of list Y , n is the number of items

Write a function to compute the covariance of X and Y . Hint: You can use *mapcar* to get the crossproduct of X and Y , which is a list of $X_i Y_i$.

```
Lisp→ (COVARIANCE '(1 2 3) '(4 5 6))
0.33333333333333304
```

- (10) b. Write a function called
- list-props*
- , with one parameter that holds a list of names such as (mary bill ted patty). Each name in the list has two properties associated with it —
- age*
- and
- sex*
- . For each name in the list, this function creates a list consisting of the name and the value of the age and sex property for the name — for example, (mary 20 f) and (bill 19 m). the function returns a list of these lists (in the same order that the names appeared in the parameter list).

```
Lisp→ (list-props '(mary bill))
((MARY 20 F) (BILL 19 M))
> (list-props '(bill mary))
((BILL 19 M) (MARY 20 F))
```

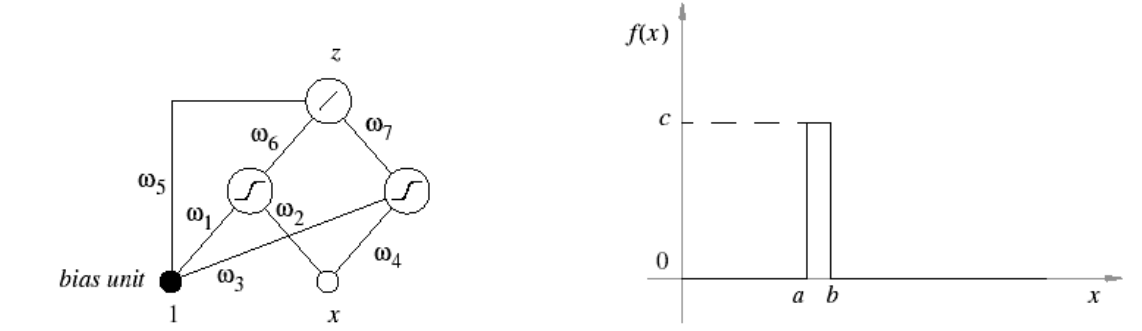
- (9) c. Develop a function called
- add-to-lis*
- that has two parameters. The first parameter holds a number and the second parameter holds a list of numbers. This function returns a list of sums that is created by adding the value of the first parameter to each number in the second parameter.

```
Lisp→ (ADD-TO-LIS 45 '(2 98 -38))
(47 143 7)
```

(30)

II. Artificial Neural Networks

(5) a. Consider the simple, single-input, single-output neural network shown below. Assuming a step function hidden-unit activation function and linear output-unit activation function derive an expression for the output function z in terms of the input vector \mathbf{x} , the weight vector \mathbf{w} , and the step function \mathbf{u} .



(10) b. What values of the weight vector w will approximate the function $f(x)$, where $a=1$, $b=2$ and $c=1$

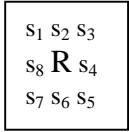
(5) c. If an input pattern is given by (x,y) give an expression for the error function E using the mean-square error criteria.

(10) d. Give an expression for the Gradient of the Error Function $(\partial E/\partial \mathbf{w})$.

(30)

III. SR Agents

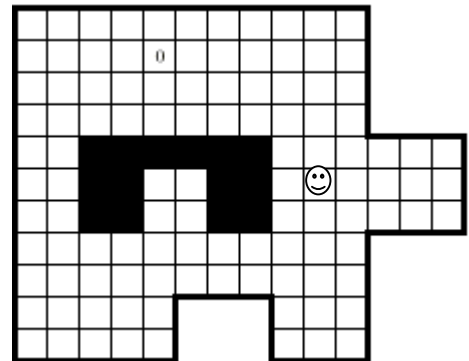
In a boundary following robot the sensory input vector s is defined by the eight sensory inputs $\{s_1, s_2, s_3, \dots, s_8\}$. Suppose the robot sensors are impaired and it can only see the tile to its front left {e.g., if the robot is going North, it can only see tile s_1 } Define a feature vector w and give the production system rules to go *west* until it meets a boundary and then it is desired that the robot move *counter-clockwise* to do boundary-following. NO DETAILS NO CREDIT. JUST DRAWING THE PATH WILL RESULT IN ZERO CREDIT!



(8) a. Define the feature vector w . {Draw the feature and give the corresponding equation}

(9) b. Give the production system rules {up is north}

(12) c. Give details for the first twelve moves and draw the path starting at point ☺ if the robot is pointing north. As $t \rightarrow \infty$, what does the robot do? Explain.



(40)
I. Recursive LISP Programming. You cannot use PROG's or any LOOP/DO constructs unless specified.

- (15) a. Define a function **subset** that takes two arguments, a function and a list. **subset** should apply the function to each element of the list. It returns a list of all the elements of this list for which the function application returns non-**nil**. For example,
{Hint: You may wish to consider the use of mapping & helper function(s)}

```
Lisp→ > (SUBSET 'NUMBERP '(A B 2 C D 3 E F))  
(2 3)
```

- (12) b. Write a function called **sub-splice**, like **subset**, but which “splices in” its first argument for the second. For example,
{Hint: You do not need mapping & helper function(s)}

```
Lisp→ > (sub-splice '(1 2) 'b '(a b c))  
(A 1 2 C)  
Lisp→ > (sub-splice '(1 2) 'b '(a (b c) d))  
(A (1 2 C) D)
```

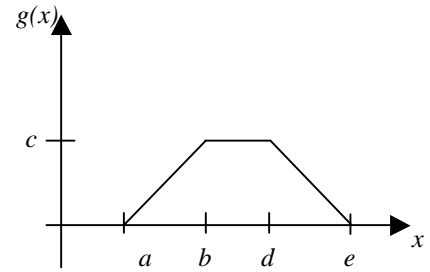
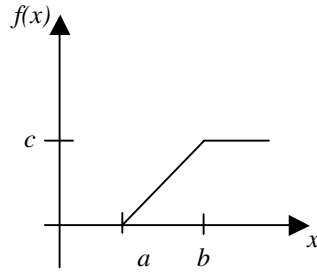
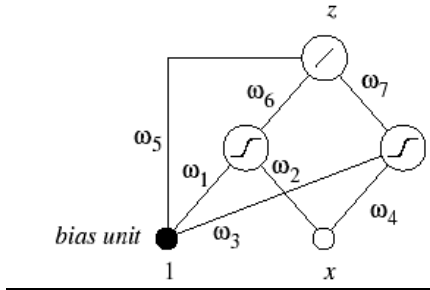
- (12) c. Use property lists to represent information about the cost and model numbers of a set of different makes of automobiles. For example, you might include that a Mercedes Benz 380SL costs \$50,000, a BMW 320i costs \$35,000, a Honda Accord costs \$18,000 and a Honda Civic costs \$12,000. Now write a function that, given a list of cars as input, returns the model of the least expensive car. { You can use any LISP construct you want, such as Prog, DO, Lambda, mapping fcn's, etc. }

```
(PUTPROP 'CAR1 'MB 'MAKE) (PUTPROP 'CAR1 '380SL 'MODEL) (PUTPROP 'CAR1 50000 'COST)  
(PUTPROP 'CAR2 'BMW 'MAKE) (PUTPROP 'CAR2 '320i 'MODEL) (PUTPROP 'CAR2 35000 'COST)  
(PUTPROP 'CAR3 'HONDA 'MAKE) (PUTPROP 'CAR3 'ACCORD 'MODEL) (PUTPROP 'CAR3 18000 'COST)  
(PUTPROP 'CAR4 'HONDA 'MAKE) (PUTPROP 'CAR4 'CIVIC 'MODEL) (PUTPROP 'CAR4 12000 'COST)  
(SETQ CAR-LIST '(CAR1 CAR2 CAR3 CAR4))  
LISP→ > (LEAST-EXPENSIVE CAR-LIST)  
(HONDA CIVIC 12000)
```

(30)

II. Artificial Neural Networks

(5) a. Consider the simple, single-input, single-output neural network shown below. Assuming a unit ramp function $\{r(t)=t \text{ for } t>0; 0 \text{ otherwise}\}$ hidden-unit activation function and linear output-unit activation function derive an expression for the output function z in terms of the input vector x , the weight vector w , and the step function u .



$z =$

(10) b. What values of the weight vector w will approximate the function $f(x)$, where $a=1$, $b=3$ and $c=2$

(10) c. Draw the network and obtain the weights to approximate function $g(x)$, where $a=1$, $b=3$, $c=2$, $d=5$, and $e=7$.

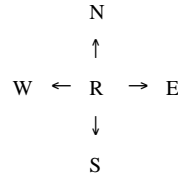
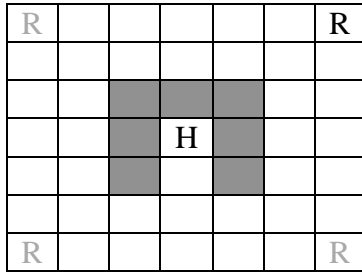
(5) d. Give an expression for the component of the Gradient of the Error Function $(\partial E/\partial w_2)$ for the network in part (a).

(30)

III. SR Agents

In our SR robot the sensory input vector s is defined by the eight sensory inputs $\{s_1, s_2, s_3, \dots, s_8\}$. Suppose the robot is placed on the grid below in any of the four corners (NE, NW, SE, SW, e.g., the NE corner shown in bold black below, all others are in gray) and it is desired that the robot move toward the center to locate the charging station, boundary follow the charging station and come to a complete rest at location H.

s_1	s_2	s_3
s_8	R	s_4
s_7	s_6	s_5



(10) a. Define a feature vector x to get the robot from any corner to its home (base) station. {Draw the feature(s) and give the corresponding equation}

(12) b. Give the production system rules for the robot to move toward the center to locate the charging station, boundary follow the charging station and come to a complete rest at location H

(7) c. Give details for the first six moves and draw the path starting at the SE corner if the robot is pointing north. As $t \rightarrow \infty$, what does your robot do? Explain.

IV. Heuristic Search

The following figure shows a search tree with the state indicated by the tuple inside parentheses. A letter indicates the state name and the integer indicates the estimated cost for finding a solution from that state (a cost of 0 indicates a goal state). Using the Graph-Search algorithm discussed in class, give the solution tree or steps using depth-first search. How many nodes did depth-first expand? Repeat using breadth-first search. How many nodes did breadth-first expand? Repeat using heuristic search. How many nodes did heuristic search expand? Repeat using A* search. How many nodes did A* expand? You must clearly justify your answer(s). "Feelings" or "intuition" are not good/sound reasons. NO JUSTIFICATION <==> NO CREDIT. You must give me the details of the algorithm in order to receive any credit for each case. Can any of these algorithms ever find N as a solution? Explain

[Done in class]

(33)

I. Recursive LISP Programming. You cannot use PROGs or any LOOP/DO constructs unless specified.

- (12) a. Write the function EQUAL-SYMBOLS of two arguments that says any two numbers are equal, a symbol is only equal to itself, and two lists are equal if all their elements are recursively EQUAL-SYMBOLS. For example:

> (EQUAL-SYMBOLS 2.2 3) T	> (EQUAL-SYMBOLS '(B 4 (C D (2.2)) E) '(B 2 (C D (0)) E)) T
> (EQUAL-SYMBOLS 2.2 'A) NIL	> (EQUAL-SYMBOLS '(B 4 (C D (2.2)) E) '(B 4 (3 D (2.2)) E)) NIL
> (EQUAL-SYMBOLS '(A) 'A) NIL	> (EQUAL-SYMBOLS NIL 'A) NIL
> (EQUAL-SYMBOLS 'A '(A)) NIL	> (EQUAL-SYMBOLS 'A NIL) NIL
> (EQUAL-SYMBOLS NIL 1) NIL	> (EQUAL-SYMBOLS 'A 'Z) NIL
> (EQUAL-SYMBOLS 1 NIL) NIL	> (EQUAL-SYMBOLS 'A (CAR '(A B))) T

- (10) b. Give a LISP definition, using a mapping function to calculate the union between two sets S1 and S2. The union of two sets is defined as the set in which every element of the union is a member of either the 1st or the 2nd set.

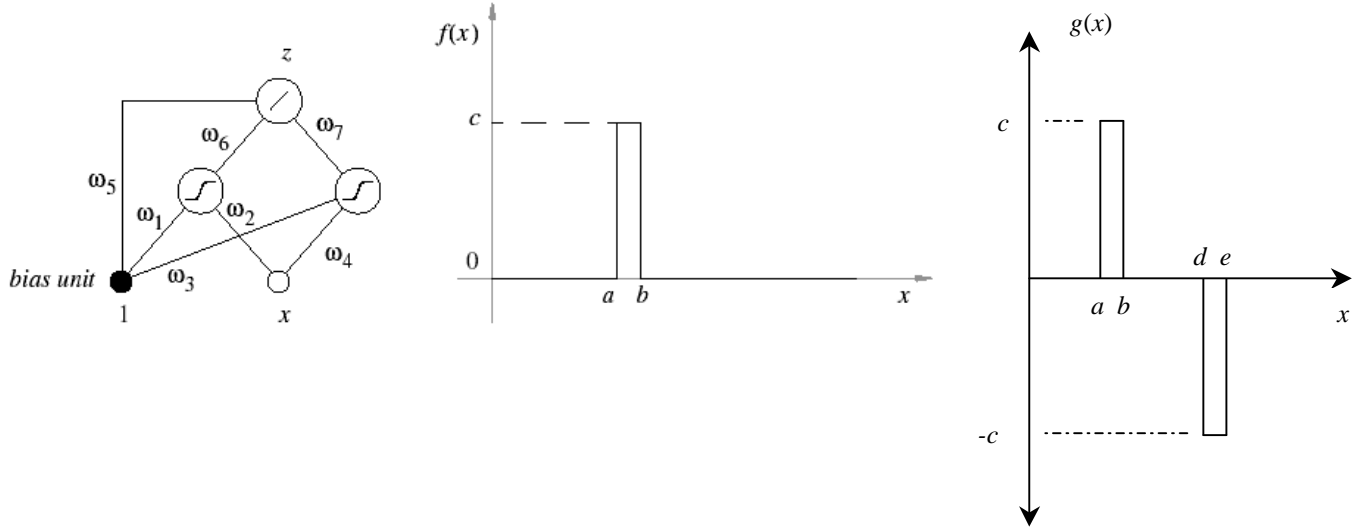
```
(union '(a b) '(c d))  
(A B C D)  
> (union '(a b) '(a d))  
(B A D)  
> (union '(a d) '(a d))  
(A D)  
> (union nil '(a))  
(A)
```

- (10) c. Write a function, MAKESET, which takes a simple list as input and makes a set using an accumulator. Make sure your answers are in the correct order.

```
> (makeset '(a b a d e b f))  
(a d e b f)  
> (makeset '(1 1 1 2 2 2 3 3))  
(1 2 3)
```

II. Artificial Neural Networks

(5) a. Consider the simple, single-input, single-output neural network shown below. Assuming a step function hidden-unit activation function and linear output-unit activation function derive an expression for the output function z in terms of the input vector \mathbf{x} , the weight vector \mathbf{w} , and the step function \mathbf{u} .



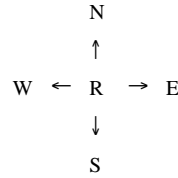
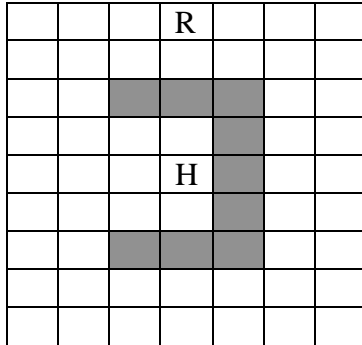
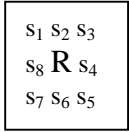
(10) b. What values of the weight vector w will approximate the function $f(x)$, where $a=1$, $b=3$ and $c=2$

(10) c. Draw the network and obtain the weights to approximate function $g(x)$, where $a=1$, $b=3$, $c=2$, $d=5$, and $e=7$.

(5) d. Give an expression for the component of the Gradient of the Error Function ($\partial E/\partial w_6$) for the network in part (a).

III. SR Agents

In our SR robot the sensory input vector s is defined by the eight sensory inputs $\{s_1, s_2, s_3, \dots, s_8\}$. Suppose the robot is placed on the grid below in the tile marked R, and it is desired that the robot move toward the center to locate the charging station, boundary follow the charging station and come to a complete rest at location H. The default direction for the robot is *south*. You want to get the robot to its home base in an *optimum* way.



(12) a. Define a feature vector x to get the robot from R to H (its home base station). {Draw the feature(s) and give the corresponding equation}

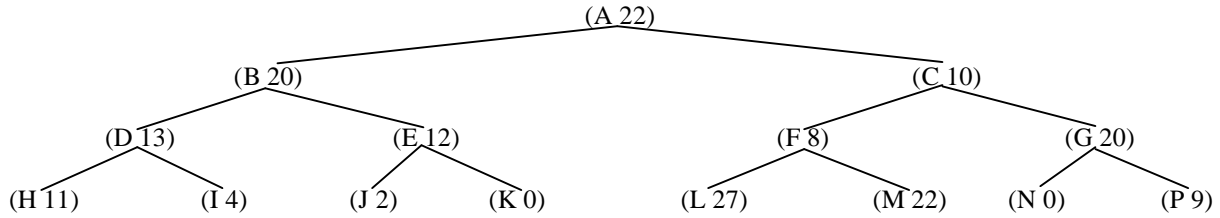
(14) b. Give the production system rules for the robot to move toward the center to locate the charging station, boundary follow the charging station and come to a complete rest at location H if the default direction for the robot is *south*.

(10) c. Give details for the first eight moves and draw the path starting at R if the robot is pointing north. As $t \rightarrow \infty$, what will your robot do? Explain.

(30)

IV. Heuristic Search

The following figure shows a search tree with the state indicated by the tuple inside parentheses. A letter indicates the state name and the integer indicates the estimated cost for finding a solution from that state (a cost of 0 indicates a goal state). Using the Graph-Search algorithm discussed in class, **give the algorithm steps** using (1) **breadth-first search**. How many nodes did breadth-first expand? Repeat using (2) **depth-first search**. How many nodes did depth-first expand? Repeat using (3) **heuristic search** (you **MUST** specify a rule to break ties). How many nodes did heuristic search expand? Repeat using (4) **A*** search. How many nodes did A* expand? You must clearly justify your answer(s). "Feelings" or "intuition" are not good/sound reasons. **NO JUSTIFICATION <==> NO CREDIT**. You must give me the details of each step of the algorithm in order to receive any credit for each case. Can any of these algorithms ever find N as a solution? Explain



BREADTH FIRST:

IV. Heuristic Search. (continued)

DEPTH-FIRST:

HEURISTIC-SEARCH:

A* SEARCH:

(33)

I. Recursive LISP Programming. You cannot use PROGs or any LOOP/DO constructs unless specified.

- (12) a. mapcar2 is like mapcar except that it applies a function of TWO arguments to TWO sets of input lists. Give the XLISP recursive program to implement mapcar2

```
> (mapcar2 '+ '(1 2) '(3 4))      > (mapcar2 '- '(1 2) '(3 4))
(4 6)                             (-2 -2)
> (mapcar2 '* '(1 2) '(3 4))      > (mapcar2 '+ () '(3 4))
(3 8)                             Nil
> (mapcar2 '* '(1 2) ())          > (mapcar2 '- () ())
nil                                 Nil
```

- (10) b. An $N \times N$ complex matrix R is represented by N lists of N pairs, where the first element of each pair is the real part of the matrix component and the second element of each pair is the imaginary part of the corresponding matrix component. Using the function mapcar2 from part (a) above, give a definition for cadd, an XLISP function to add complex matrices.

```
> (cadd '((1 2) (3 4)) '((5 6) (7 8)))  > (cadd '((1 2) (3 4)) ((5 6) (7 8))) '((10 10) (10 10)) ((1 1) (1 1)))
((6 8) (10 12))                         ((11 12) (13 14)) ((6 7) (8 9))
> (cadd '((1 2) (3 4)) ())              > (cadd () '(1 2) (3 4))
()                                         nil
> (cadd () ())                            > (cadd '(1 2)) '(3 4))
nil                                       ((4 6))
```

- (10) c. Give a recursive LISP function or use a mapping function to transpose a matrix of complex numbers, i.e., the transpose of R complex matrix R' is given by

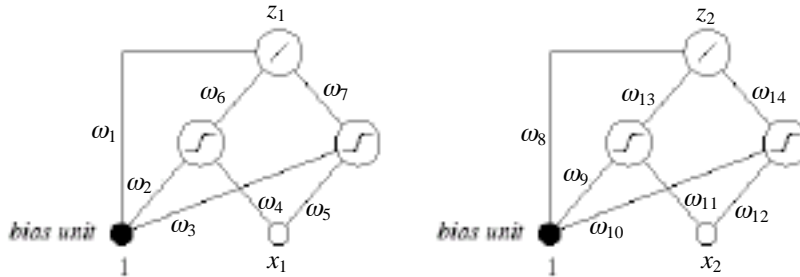
$$\begin{vmatrix} (1+j2) & (3-j4) \\ (5+j6) & (7-j8) \end{vmatrix} \rightarrow \begin{vmatrix} (1+j2) & (5+j6) \\ (3-j4) & (7-j8) \end{vmatrix}$$

```
> (ctransp '((1 2) (3 -4)) ((5 6) (7 -8)))      > (ctransp () )
(((1 2) (5 6)) ((3 -4)) ((7 -8)))              nil
> (ctransp '((1 2) (3 4)) ((5 6) (7 8)))        > (ctransp '((10 10) (10 10)) ((1 1) (1 1)))
(((1 2) (5 6)) ((3 4)) ((7 8)))                (((10 10) (1 1)) ((10 10)) ((1 1)))
```

(24)

II. Artificial Neural Networks

(12) a. Consider the simple, two-input, two-output neural network shown below. Assuming a sigmoid function hidden-unit activation function and linear output-unit activation function for each, derive an expression for the output function vector \mathbf{z} in terms of the input vector \mathbf{x} , the weight vector \mathbf{w} , and the sigmoid function \mathbf{u} .



In particular, give the values for the h 's and net 's for the 4 sigmoid units, and the z 's in terms of w 's, h 's, and z 's in terms of w 's, x 's, and $\mathbf{u}(_)$. { 12 equations / 12 points }

(2) b. How many total independent weights are contained in this neural network? Show Work! No Work, No Credit.

Answer: ____ independent weights.

(6) c. Without any local variables, how many additions, multiplications, and function evaluations (u, \underline{u}') are required to compute the output of the neural network in terms of the inputs and the weights of the neural network? Show Work! No Work, No Credit.

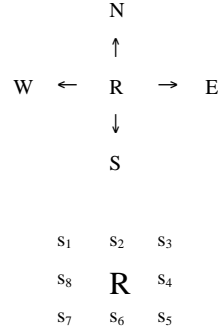
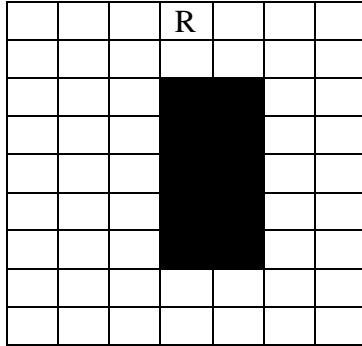
Additions: _____ Multiplications: _____ Function Evaluations: _____

(4) d. Give an expression for the component of the Gradient of the Error Function ($\partial E / \partial w_7$) for the network in part (a). Also give an expression for the component of the Gradient of the Error Function ($\partial E / \partial w_{14}$) for the network in part (a).

(19)

III. SR Agents

In our SR robot the sensory input vector s is defined by the eight sensory inputs $\{s_1, s_2, s_3, \dots, s_8\}$. Suppose the robot is placed on the grid below in the tile marked R, and it is desired that the robot move toward the center to locate obstacle, boundary follow the obstacle *clockwise*.



(8) a. Define the four feature vectors x_i to get the robot from R to boundary follow the obstacle *clockwise*. {Draw the four feature(s) and give the four corresponding equations}

(5) b. Give the five production system rules for the robot to move toward the center and boundary follow the obstacle *clockwise*.

(5) c. Give details for the first four moves and draw the path starting at R if the robot is pointing north. As $t \rightarrow \infty$, what will your robot do? Explain.

IV. Heuristic Search

In the following 8-Puzzle we use {D,R,U,L} and LIFO to resolve ties, we have a choice of Breadth-First Search (BFS), Heuristic Search (HS) and A* Search. Give the order of preference of the procedures (i.e., which procedure yields the least, the second best, and worst total of nodes expanded) if you assume that moves that take you back to a previous state are not allowed?

You must clearly justify your answer. Justification may take the form of drawing the search tree and counting the nodes evaluated or any other "sound reason" by which you can reach the conclusion. "Feelings" or "intuition" are not good/sound reasons. NO JUSTIFICATION <==> NO CREDIT

INITIAL

1 _ 4

8 3 2 ==>

7 6 5

GOAL

1 2 3

8 _ 4

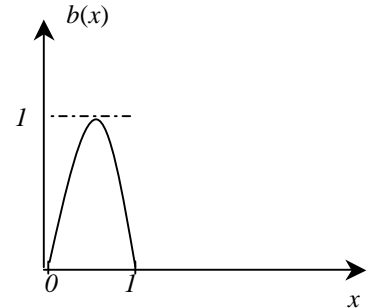
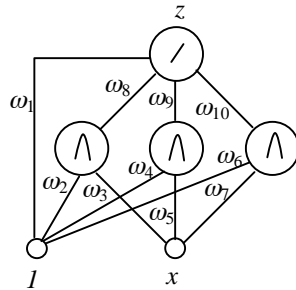
7 6 5

Where $f(n) = g(n) + h(n)$ $g(n) = d(n)$ = depth of each node $h(n) = P(n)$ = Manhattan distance

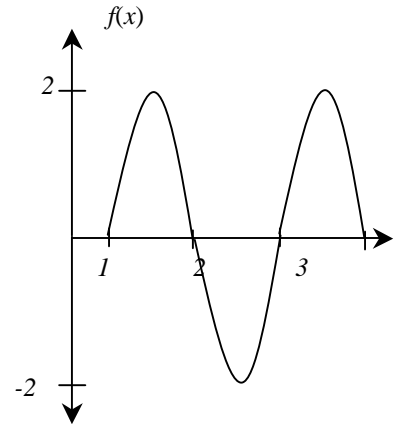
(24)

II. Artificial Neural Networks

(6) a. Consider the simple, single-input, single-output neural network shown below. Assuming a unit bump hidden-unit activation function $b(x)$ {see \wedge below} and a linear output-unit activation function derive an expression for the output function z in terms of the input vector x , the weight vector w , and the bump function b .



(6) b. What values of the weight vector w will approximate the function $f(x)$ shown?



(6) c. If an input pattern is given by (x,y) give an expression for the error function E using the mean-square error criteria.

(6) d. Give an expression for the component of the Gradient of the Error Function ($\partial E/\partial w_5$) for the network in part (a).

(19)

III. SR Agents

In an SR robot the sensory input vector s is defined by the three sensory inputs $\{s_1, s_2, s_3\}$. Suppose the robot is placed on the grid below in the tile marked R and it points north, and it is desired that the robot avoid obstacles.

s_1	s_2	s_3
	↑	
	R	

1	2	3
8	↑	4
7	6	5

		N		
		↑		
W	←	R	→	E
		↓		
		S		

Actions	Result
1) Fwd	R goes to 2 and points North
2) Rev	R goes to 6 and points South
3) HL	R goes to 8 and points West
4) SL	R goes to 1 and points N
5) HR	R goes to 4 and points East
6) SR	R goes to 3 and points N

(8) a. Define feature vectors x_i to get the robot R to avoid obstacles. The actions the robot can take are to move 1 tile (the robot can move 1 tile diagonally) specified in the table above, where the robot is assumed to be at the tile designated “R” and points north. These actions are: forward (Fwd), reverse (Rev), hard left (HL), soft left (SL), hard right (HR) and soft right (SR). {Draw the feature(s) and give the corresponding equations} Also specify tight space conditions/restrictions if you assume that the robot can be off not less than 10% left or right on any one tile position.

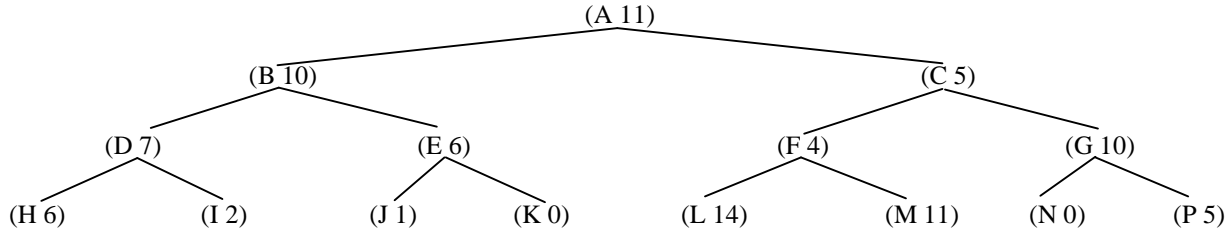
(5) b. Give the production system rules for the robot to avoid obstacles.

(5) c. Give details for the first five moves and draw the path starting at R if the robot is pointing up. Use (row,col) indexing to refer to the robot position. The robot starts at position (1,4). Show your work. Will the robot explore the entire space?

Start:

IV. Heuristic Search

The following figure shows a search tree with the state indicated by the tuple inside parentheses. A letter indicates the state name and the integer indicates the estimated cost for finding a solution from that state (a cost of 0 indicates a goal state). **Give the algorithm steps** using (1) A* search and (2) IDA* search. In particular, how many nodes did A* and IDA* expand? Which algorithm performs the best? You must clearly justify your answer(s). "Feelings" or "intuition" are not good/sound reasons. NO JUSTIFICATION <==> NO CREDIT. You must give me the details of each step of the algorithm in order to receive any credit for each case. Can any of these algorithms ever find N as a solution? Explain



A* SEARCH:

IDA* SEARCH:

Which algorithm performs the best?

Explanation for whether or not the goal node (N 0) is/is not found: