6.034 Final Examination

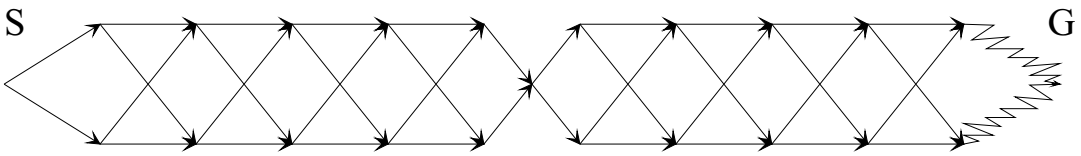
Fall 2001

Name	
E-mail	

Problem Number	Max	Score	Grader
Problem 1	16		phw rcb psz jb ac jl an dr js
Problem 2	16		phw rcb psz jb ac jl an dr js
Problem 3	20		phw rcb psz jb ac jl an dr js
Problem 4	20		phw rcb psz jb ac jl an dr js
Problem 5	18		phw rcb psz jb ac jl an dr js
Problem 6	10		phw rcb psz jb ac jl an dr js
Total	100		

Problem 1: Search in a weird city (16 points)

You have decided to look for a job driving a cab. The recruiter, an MIT graduate, asks you to solve some search problems for streets laid out as shown by the lines in the figure.



The map is drawn to scale, but **note that the final, jagged streets converging on G can be assumed to be 100 times longer than any other street**. All streets are one-way, left to right. In the absence of any other criterion, or a tie, you are to follow the option closer to the top of the page.

Your job is to find a path from the start node, S, to the goal node, G, according to various conditions. Note that the total number of nodes is 23.

You may express your answers either as numbers (e.g. 8) or variable-free expressions (e.g. $2^0 + 2^1 + 2^2 + 2^3$) or both. You get full credit if either is right. Also, you will increase your partial credit if you correctly note when two answers should be the same, even if both are wrong.

Part A: Depth-First Search, with backtracking (2 points)

You are ***not*** to use a visited (enqueued) list or expanded list. Search is to terminate when a path that reaches the goal appears at the front of the search queue. Indicate the number of nodes expanded.

Part B: Breadth-First Search (2 points)

You ***are*** to use a visited (enqueued) list. Search is to terminate when a path that reaches the goal appears at the front of the search queue. Indicate the number of nodes expanded.
queue.

Part C: Branch and Bound (2 points)

You are to find the optimal path, using ***neither*** an expanded list, nor an admissible heuristic. Indicate the number of nodes expanded.

Part D: A* (4 Points)

You are to find the optimal path, ***using*** an expanded list, and the distance to the goal as an admissible heuristic. Indicate the number of nodes expanded.

Part E: Bidirectional Breadth-First Search (4 Points)

You are ***not*** to use a visited (enqueued) list or expanded list. Search from the left is interdigitated with search from the right. That is, one search starts from the left and produces all paths of depth 1; then, another search starts from the right, goes against the arrows, and produces all paths of depth 1. The two searches then alternate, adding additional layers to their search trees, until an expansion produces a path from the left that shares a node with a path from the right. Indicate the number of nodes expanded.

Part F: Comparison (2 Points)

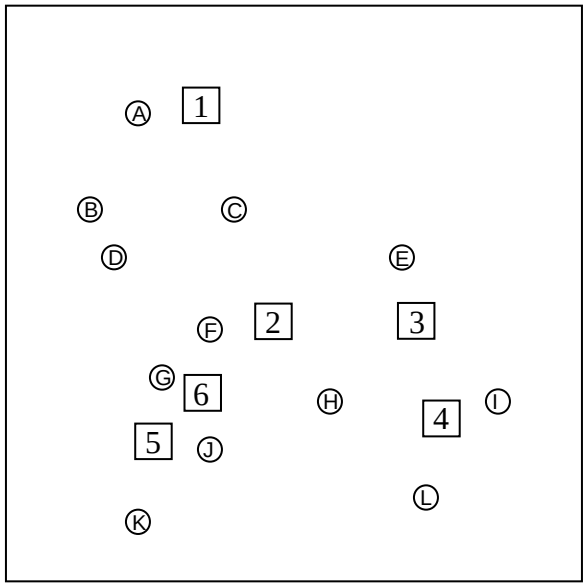
Evidently, bidirectional breadth-first search does less work than unidirectional breadth-first search would have done in Part E. In fact, the ratio of unidirection nodes expanded to bidirectional nodes expanded is on the order of (circle best answer).

- 2
- $b^{d/2}$
- b^{d}
- $2 \times b \times d$

Where b is the branching factor and d is the total depth as seen by unidirectional search.

Problem 2: Nearest neighbors and ID trees (16 Points)

Part A: Nearest neighbors, backwards (6 Points)



You are using the circled data points to classify the square test points.
Each circle is either + or -, but the labels are not shown here.
The square test points are not used to classify other points.

You want to recover the labels for the circles, given some results of classifying the squares:

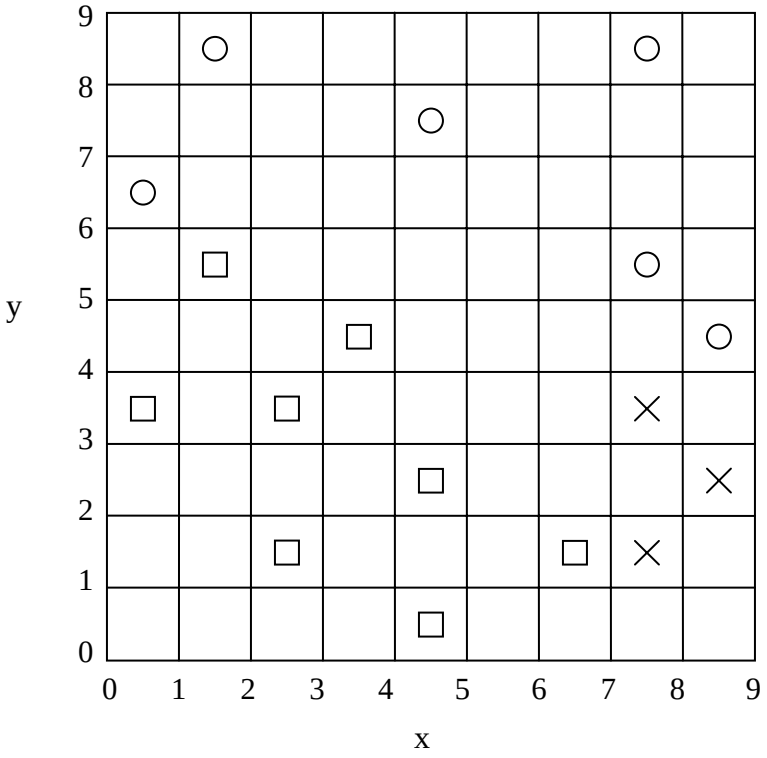
Square Point	Using 1 nearest neighbors	Using 3 nearest neighbors
1	-	+
2	-	
3		+
4	+	-
5		-

For each of the following, mark + , -, or U for cannot be determined.

_____ Circle L.
_____ Circle I.
_____ Circle H:
_____ Circle E.
_____ Circle K.
_____ Circle C.
_____ Square 6 using 1 nearest neighbor.
_____ Square 6 using 3 nearest neighbors.
_____ Square 2 using 3 nearest neighbors
_____ Square 3 using 1 nearest neighbor.
_____ Square 5 using 1 nearest neighbor

Part B: Building a tree (6 Points)

For the following graph, show the decision boundaries that are produced by constructing an ID tree using the standard minimum-entropy greedy algorithm to ***completely classify*** the data. Make cuts using only X or Y (that is, ***vertical and horizontal lines only***). For each cut, write the coordinates of the end points of the line segment (such as (3,0) to (3, 9)). For each cut, calculate the average entropy (that is, disorder) for the points divided by that cut. We provide more than enough places to write.



Cut 1: (_____, _____) to (_____, _____) Entropy: _____

Cut 2: (_____, _____) to (_____, _____) Entropy: _____

Cut 3: (_____, _____) to (_____, _____) Entropy: _____

Cut 4: (_____, _____) to (_____, _____) Entropy: _____

Cut 5: (_____, _____) to (_____, _____) Entropy: _____

Cut 6: (_____, _____) to (_____, _____) Entropy: _____

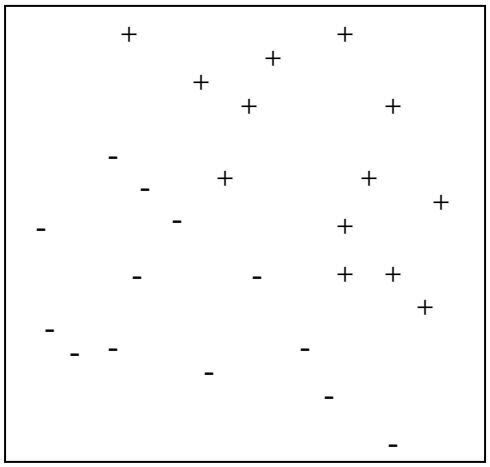
Part C: Derived features (4 Points)

Looking at several graphs of data, you decide to experiment with derived features, thinking you may do better than just using X or Y as plotted. For each figure, pick a feature that an ID tree can use to **classify the data correctly with only a single cut**. Note that the origin is in the lower left, and the square is 1 by 1.

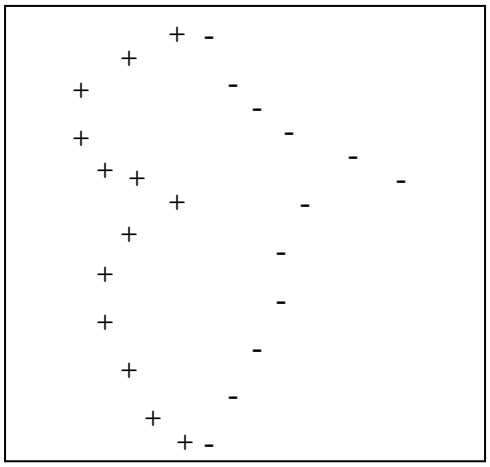
Choose from these features:

1. X 2. Y 3. X+Y 4. X-Y 5. XY 6. X^2+Y

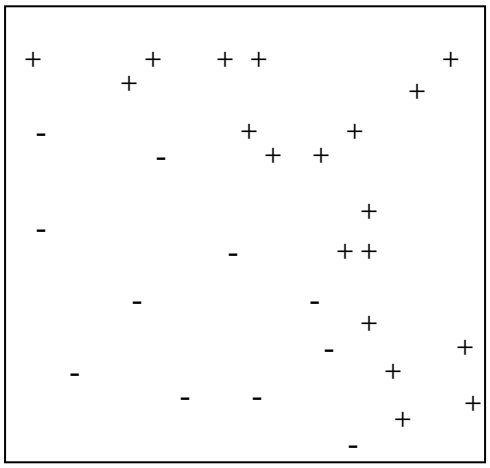
7. $X+Y^2$ 8. X^2+Y^2 9. atan(Y/X) 10. $\sin(4\pi X)+Y$ 11. $X+\sin(4\pi Y)$



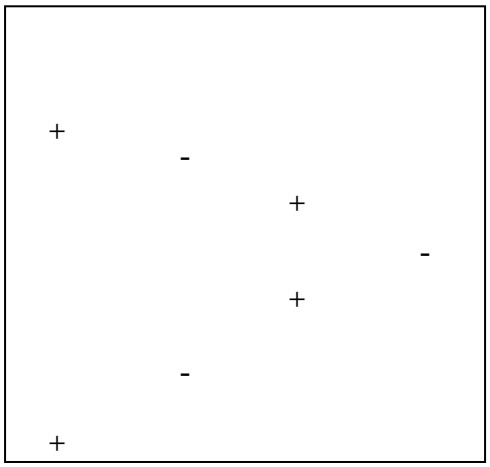
Derived Feature #: _____



Derived Feature #: _____



Derived Feature #: _____



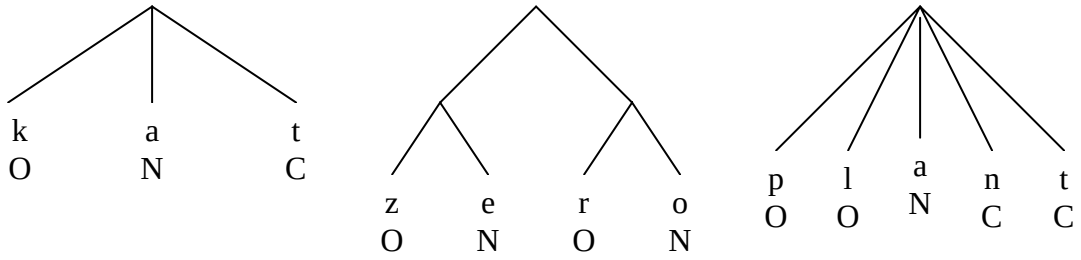
Derived Feature #: _____

Problem 3: Constraint Propagation (20 Points)

Having thoroughly mastered the art of map coloring and rejected job offers from the numerous cartography consulting firms, you try some basic natural language understanding. Your job is to enable an AI system to find the syllabification of a word, given its sequence of phonemes. After consulting with some 6.034 TAs, you learn that a syllable is organized into three elements: an optional onset, the nucleus, and an optional coda.

- Onset: One or more consonants before the vowel in the syllable
- Nucleus : The vowel in the syllable
- Coda: One or more consonants after the vowel in the syllable

Some examples:



You decide to model assignment of the labels O, N, and C to each phoneme using the vocabulary of constraint propagation. (The variables of CSP are the phonemes, and the domain of values is O, N, and C.) You start with a simple example, the word `elektivz` (electives). In the beginning, the domains of the variables looks like this:

e	**l**	**e**	**k**	**t**	**i**	**v**	**z**
O	O	O	O	O	O	O	O
N	N	N	N	N	N	N	N
C	C	C	C	C	C	C	C

Then Your faithful TAs also give you a few unary constraints:

1	A consonant may not be assigned the value N
2	A vowel may not be assigned the value O or C
3	A word cannot begin with a C
4	A word cannot end with an O

Now, using these unary constraints, you can limit the domains of the variables, producing the following result:

		e	**l**	**e**	**k**	**t**	**i**	**v**	**z**
O			O		O	O		O	
N		N		N			N		
C			C		C	C		C	C

Part A : Binary Constraints (8 Points)

"Oops," your TA says while scratching his head. "I forgot to give you a binary constraint…"

5	An O cannot be followed by a C

You race off and immediately try to find a consistent set of variable assignments using depth-first search. You proceed through the word, left to right, attempting assignments in the order O, N, C. After each assignment, you test constraint 5, backing up if the constraint is violated. Show the search tree:

Now, fill in the resulting syllabification:

	e	**l**	**e**	**k**	**t**	**i**	**v**	**z**
Value								

Part B: Continuous Constraints (6 Points)

After racing proudly back to your TA, he scratches his head again. "That syllabification doesn't seem quite right," he says. "In order to prevent assignments that result in ill-formed syllables like `agf` and `rdo`, you need to take into account sonority." Luckily, Professor Winston walks in. "Oh, sonority is just the degree to which a phoneme sounds like a vowel," he says and whips this chart out of his back pocket; note that phonemes that lie in the same box are considered equally sonorous.

Least Sonorous			Most Sonorous
g, d, b, k, t, p	z, s, v, f	r, l, n, m	u, o, i, e, a

Now, your TA explains to you two more constraints about rising onsets and falling codas:

6	O_1-O_2 sequences must have rising sonority $O_1 < O_2$
7	C_1-C_2 sequences must have equal or falling sonority $C_1 \geq C_2$

"Now you should have enough information," he says. Again, you race off and do another search as in Part A, but now with two more constraints. Show the tree:

What is the first variable of `elektivz` that fails an assignment?

How many assignments are later changed?

What is the resulting syllabification?

	e	**l**	**e**	**k**	**t**	**i**	**v**	**z**
Value								

Part C: Conservative Checking (6 Points)

Now Winston suggests a bit of conservative constraint checking. **Conservative checking** means using the constraints associated with a variable to restrict the domain of that variable whenever the domain of a neighbor is reduced to one value. This checking continues from neighbor to neighbor as long as a domain is reduced to one value.

Now you are to syllabify the word `perseptronz` (perceptrons) using conservative checking. First, apply the unary constraints and then do depth-first search, left-to-right, with conservative checking., using all the constraints. Again attempt assignments in the order O, N, C, and show your tree.

How many total checks do you make? ☐

How many assignments are made? ☐

What is the resulting syllabification?

	p	**e**	**r**	**s**	**e**	**p**	**t**	**r**	**o**	**n**	**z**
Value											

Problem 4: Neural nets (20 Points)

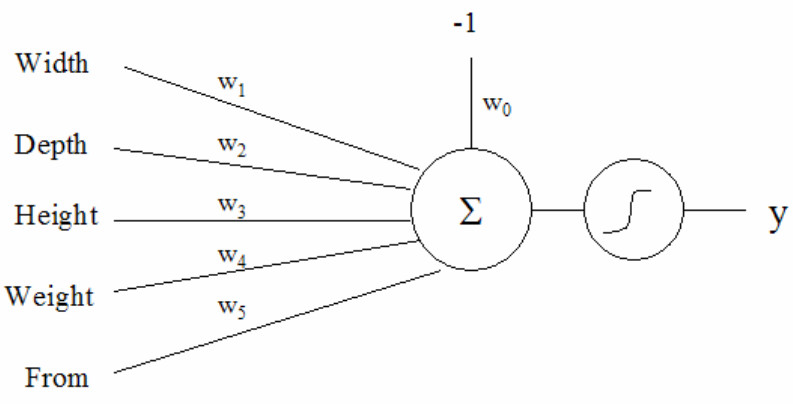
Part A: (2 Points)

After years of receiving crummy holiday and birthday gifts you are determined to find a way to decide if a gift is good or bad before opening it. Because this sounds like a task for machine learning, you decide to use a neural network. The output is close to 0 if it is a bad gift, and close to 1 if it is a good gift. Your inputs are all values between 0 and 1. They are:

Width: Ranging from 0 (narrow) to 1 (wide)
Depth: Ranging from 0 (shallow) to 1 (deep)
Height: Ranging from 0 (short) to 1 (tall)
Weight: Ranging from 0 (light) to 1 (heavy)
From: 0 (from an ordinary relative) or 1 (from a wealthy relative)

You go back and look at hundreds of gifts you received from the last several years, about half good and half bad, and train the neural net using that data.

Your first idea is to construct a one layer neural net with all inputs going into a single sigmoid unit, as such:



Part A.1

You know that anything from a wealthy relative must be a good gift, so after training the corresponding weight should be:

A. Highly positive
B. Close to zero, but positive
C. Close to zero, but negative
D. Highly negative

Part A.2

You finish training the net just in time for the holiday season, and use it to test its accuracy on this year's holiday gifts. Because the output of a sigmoid is a continuous value but you want a discrete value, you say that any outputs below 0.5 predict bad gifts and any above 0.5 predict good gifts. Unfortunately, you discover this first network does a horrible job.

Circle all possible reasons why:

A. Your neural net model doesn't have enough units for the complicated interactions between the input variables
B. You didn't give the net enough training data
C. You initialized all the weights as random numbers between 0 and 1
D. You initialized each weight to a very large positive or negative number.
E. Sigmoid units aren't powerful enough to capture the relation you want, so a perceptron should have been used instead

Part B (6 points)

You know from experience that only the following are good gifts:

- Small envelopes because they contain cash
- Gifts from wealthy relatives, regardless of other characteristics
- Tall, wide, and deep gifts

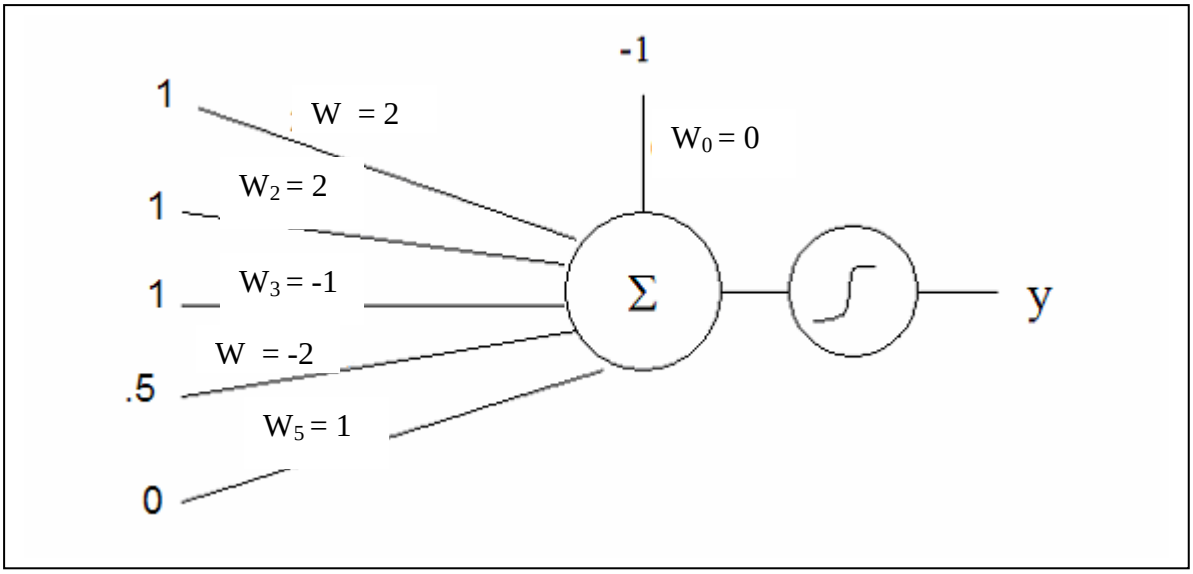
You decide to create three separate neural nets, one for each of the three categories above. Each is identical to the one drawn above, but the desired output is 1 if a sample specifies the category associated with the net. On a test sample, if at least one of the outputs is close to 1, then you know it is a good gift.

One training sample has the following characteristics:

Width = 1, Depth = 1, Height = 1, Weight = .5, From = 0;

It is known to be a good gift, so the desired output $y^* = 1$.

$w_0 = 0$, $w_1 = 2$, $w_2 = 2$, $w_3 = -1$, w_4 -2, $w_5 = 1$.



Run one step of training on the neural net assuming a learning rate r = 100, and initial weights shown above, and the standard error function, $E = \frac{1}{2}(y - y^*)^2$.

To simplify your calculations, use the following values of the sigmoid function **(NOTE: These are not the actual values of the sigmoid function but use them to make your life easier!):**

Sigmoid (0) = 0.5, Sigmoid(1) = .75, Sigmoid(2) = .9, and Sigmoid(-x) = 1 – Sigmoid(x)

$w_0 =$

$w_1 =$

$w_2 =$

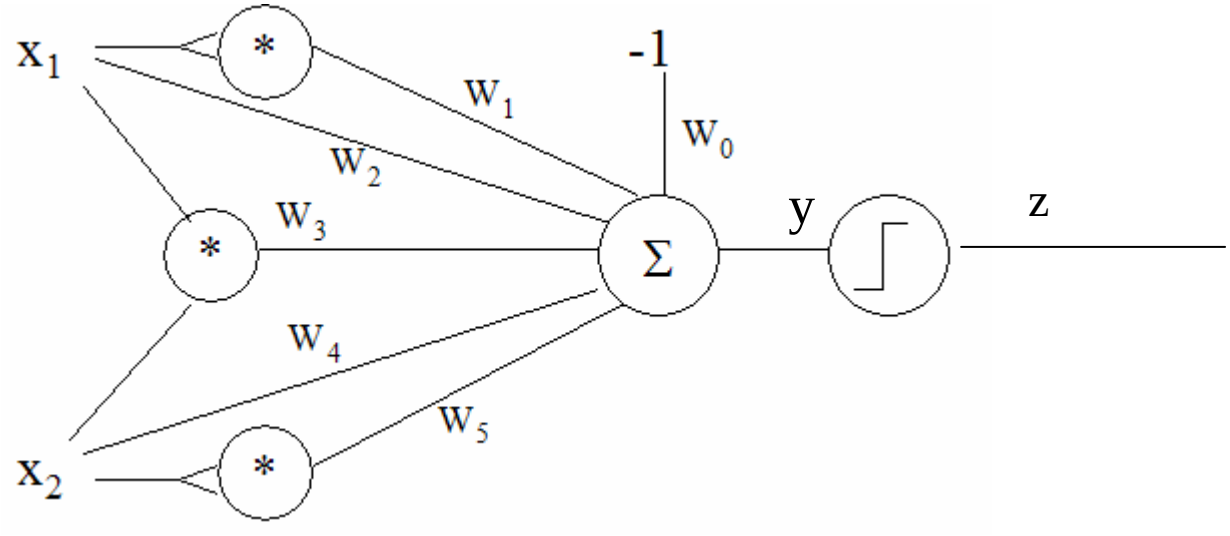
$w_3 =$

$w_4 =$

$w_5 =$

Part C (4 Points)

You are tired of being limited by perceptrons that can only draw straight lines in your input space, so you construct the following net:



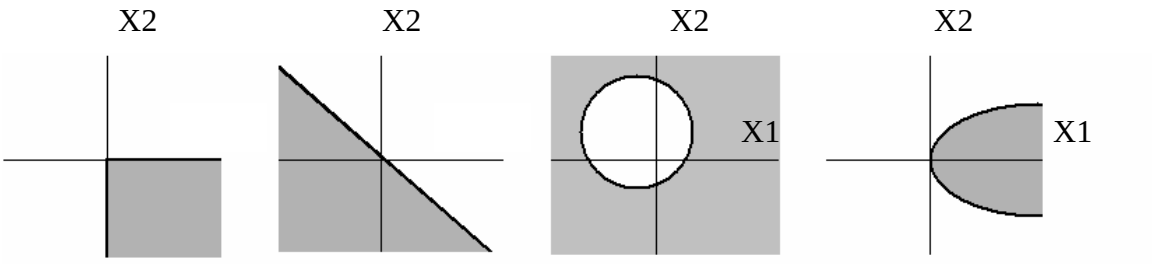
The output of * units are the product of all the inputs, and the output of a Σ unit is the sum of all inputs.

Note that the top and bottom * units have doubled x_1 and x_2 inputs, so they compute the squares of x_1 and x_2 .

So the input to the step-function threshold unit is

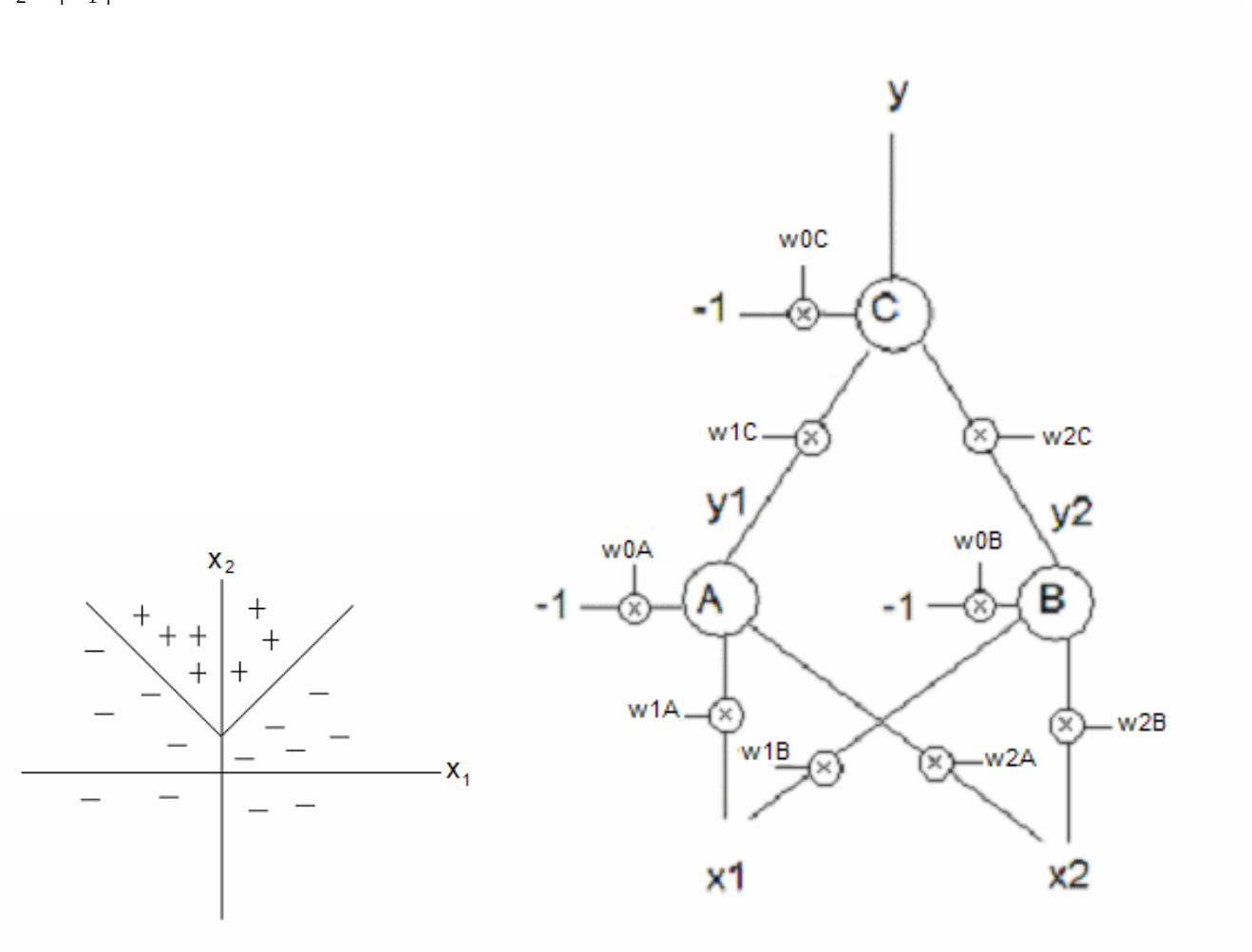
$y = w_1x_1^2 + w_2x_1 + w_3x_1x_2 + w_4x_2 + w_5x_2^2 - w_0$

Circle all of the following on which a single neuron as described above can separate the shaded region from the unshaded region. The region shown in each image covers the entire possibility of inputs.



Part D: (8 Points)

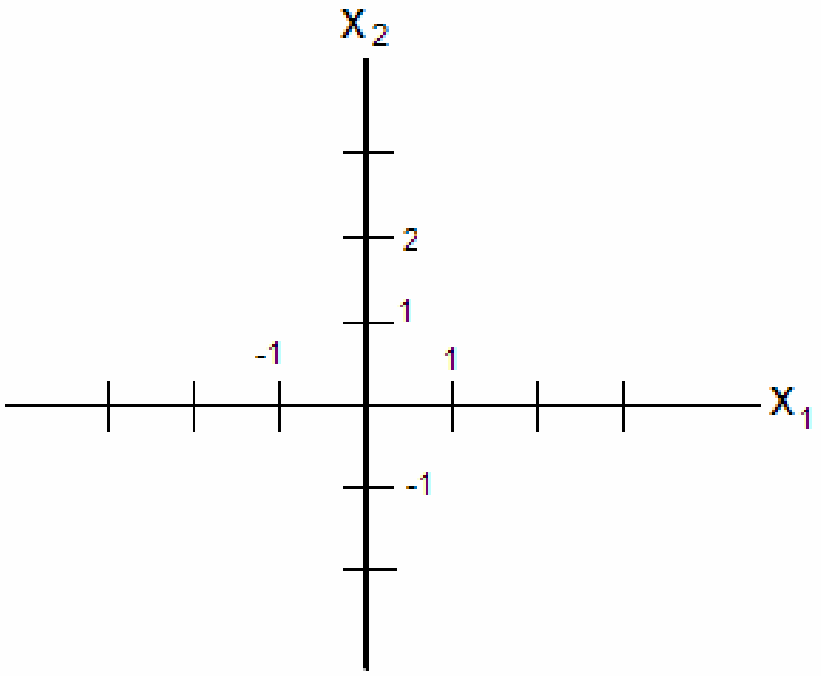
You want to create a neural net using perceptron units that will determine whether $x_2 > |x_1| + 1$.



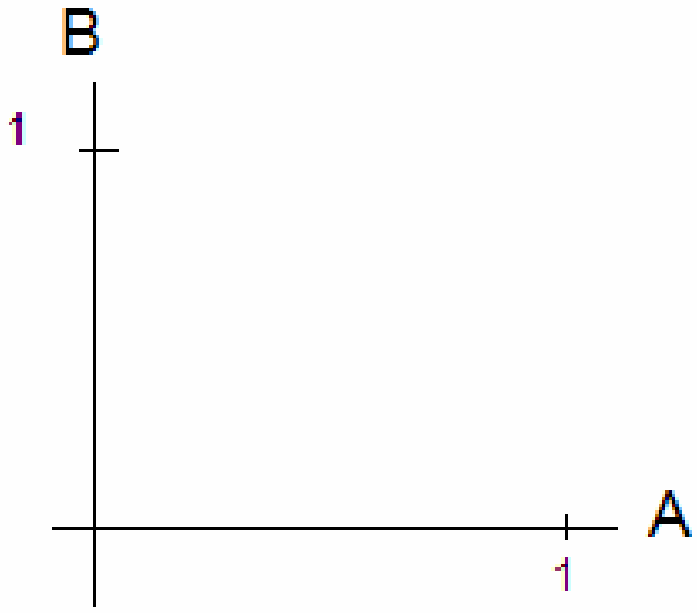
Give values for the missing weights of net A so that it correctly classifies the data according to the function $x_2 > |x_1| + 1$. The output for each instance should be 0 for instances labeled – and 1 for instances labeled +.

w_{0A} =
w_{1A} = 1
w_{2A} = 1
w_{0B} = 1
w_{1B} =
w_{2B} =
w_{0C} =
w_{1C} =
w_{2C} =

On the following chart, show the decision boundaries for perceptrons A and B. Be sure to label the boundary as A or B and identify whether each side gets a 0 or 1 label.



On the following chart, draw in all combinations of outputs for units A and B. Label each combination with the classes associated with that combination (+ or – or both), given the function $x_2 > | x_1 | + 1$.



Problem 5: SVM (18 Points)

Part A (2 Points)

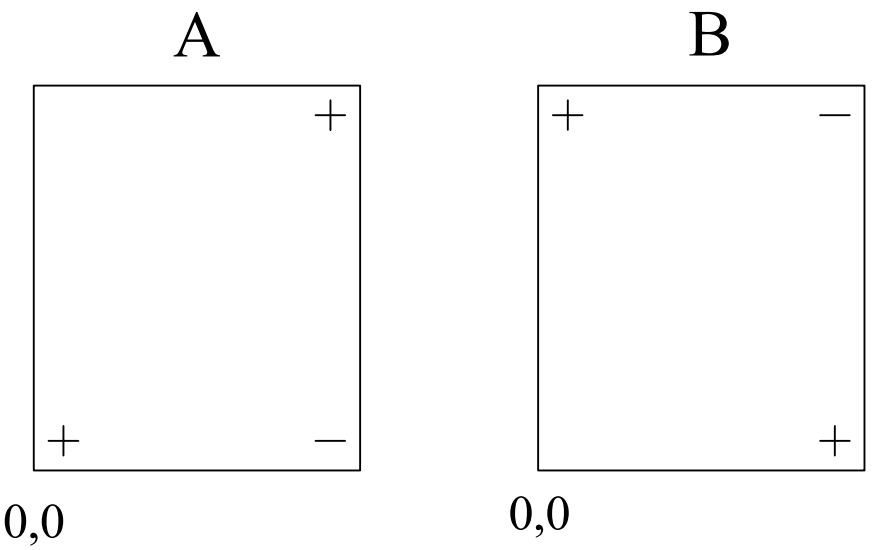
Consider the two arrangements of + and – points in drawings A and B, in which the arrangement in B is produced by rotating the arrangement in A around the center. Then select true or false for the following assertions:

Given a radial-basis kernel, $\kappa(v_1, v_2) = e^{\frac{-\|v_1 - v_2\|^2}{0.5}}$, the boundary separating the + and – points in B can be aligned with the boundary in A by a combination of translation and rotation.

- True
- False

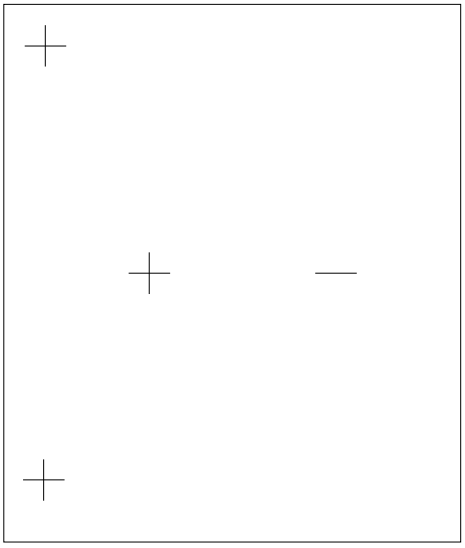
Given a polynomial kernel, $\kappa(v_1, v_2) = (v_1 \cdot v_2)^2$, where the vectors are drawn from the origin, the boundary separating the + and – points in B can be aligned with the boundary in A by a combination of translation and rotation..

- True
- False

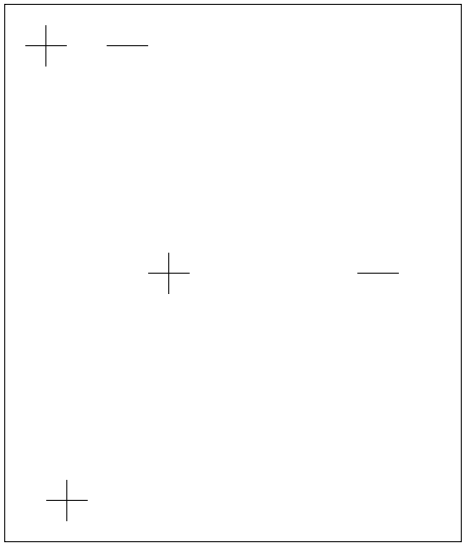


Part B (4 Points)

Assume that a support vector machine is to learn to separate the + and – points in the following diagrams. Sketch the -1 or +1 lines (gutters of the widest street) and circle the points corresponding to support vectors assuming a polynomial kernel, $\kappa(v_1, v_2) = (v_1 \cdot v_2)^1$



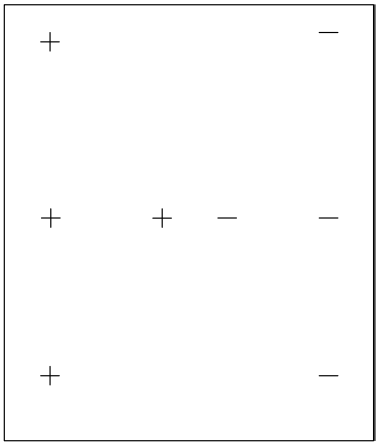
0,0



0,0

Part C.1(2 Points)

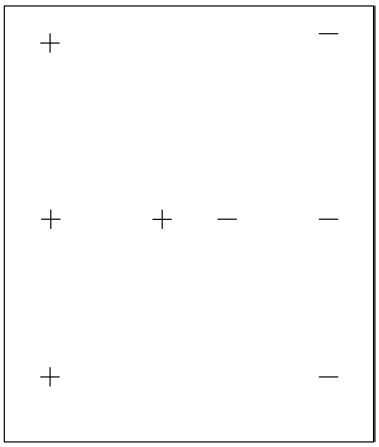
Assume that a support vector machine is to learn to separate the + and – points in the following diagram. Sketch the ***decision boundary*** (the 0 line, not the -1 or +1 lines) and circle the points corresponding to support vectors assuming a radial basis kernel, $\kappa(v_1, v_2) = e^{\frac{-\|v_1 - v_2\|^2}{2\sigma^2}}$, and a ***very large sigma***.



0,0

Part C.2(2 Points)

Assume that a support vector machine is to learn to separate the + and – points in the following diagram. Sketch the ***decision boundary*** (the 0 line, not the -1 or +1 lines) and circle the points corresponding to support vectors assuming a radial basis kernel, $\kappa(v_1, v_2) = e^{\frac{-\|v_1 - v_2\|^2}{2\sigma^2}}$, and a ***very small sigma***.



0,0

Part D (8 Points)

On the separate sheet, there are nine colored diagrams, labeled A through I, representing graphs of SVMs trained to separate pluses (+) from minuses (-). Indicate which diagram results from using which kernel function by writing the letter of the diagram next to the corresponding kernel.

<u>Note that the points are the same in diagrams A, B, C, and D. They are also the same in E, F, and G.</u>

$\kappa(v_1, v_2) = (v_1 \cdot v_2)^1$		$\kappa(v_1, v_2) = e^{\frac{-\|v_1 - v_2\|^2}{0.5}}$	
$\kappa(v_1, v_2) = (v_1 \cdot v_2)^2$		$\kappa(v_1, v_2) = e^{\frac{-\|v_1 - v_2\|^2}{0.22}}$	
$\kappa(v_1, v_2) = (v_1 \cdot v_1)^2$		$\kappa(v_1, v_2) = e^{\frac{-\|v_1 - v_2\|^2}{0.08}}$	

Problem 6: Miscellaneous (10 points)

Circle the **single** phrase that **best** completes the following fragments. All multiple votes will be rejected. No points will be subtracted for incorrect answers.

When running a rule-based system:

- A rete can be used to speed up forward chaining.
- Backward chaining can help keep user questions focused on a current hypothesis.
- Forward chaining deduction systems never delete assertions.
- Backward chaining requires unification matching procedures (variables to variables).
- None of the above
- All of the above

Programs can answer certain how and why questions about their past behavior provided that:

- They use a rete
- They do not do forward chaining
- They build and/or trees
- They are not rule based
- None of the above
- All of the above

Given unknown objects that move in two dimensions (no rotations), then point prediction requires:

- One point and a corresponding point in one model image.
- Two points and corresponding points in two model images
- Three points and corresponding points in two model images.
- Three points and corresponding points in three model images.
- None of the above.

In the Yip-Sussman theory of phonological learning:

- Words are assumed to be stored as sequences of phonemes described as distinctive feature vectors.
- Learning takes place through a generalization search through a buffer of distinctive feature vectors.
- The phonemes of English correspond to less than 10% of the distinctive feature combinations.
- Phonological rules are captured by bi-directional constraints
- All of the above
- None of the above

In the Yip-Sussman theory of phonological learning:

- 100s of examples are required to learn each rule.
- Externally provided near misses are essential.
- The search is exhaustive, thus guaranteeing the best possible generalizations.
- The goal of each search is to match all positives presented while matching the fewest negatives.
- All of the above
- None of the above

Arch learning demonstrates:

- That programs can learn something definite from individual examples
- Learning progresses faster when teachers can formulate good near misses.
- Positive examples generalize; negative examples specialize.
- The importance of representations that make the right details explicit.
- All of the above
- None of the above

Near-miss examples

- Guide specialization
- Prevent specialization
- Enable reification
- Enable learning without knowing whether an example is positive or negative.
- All of the above
- None of the above

Felicity conditions ensure that, in part:

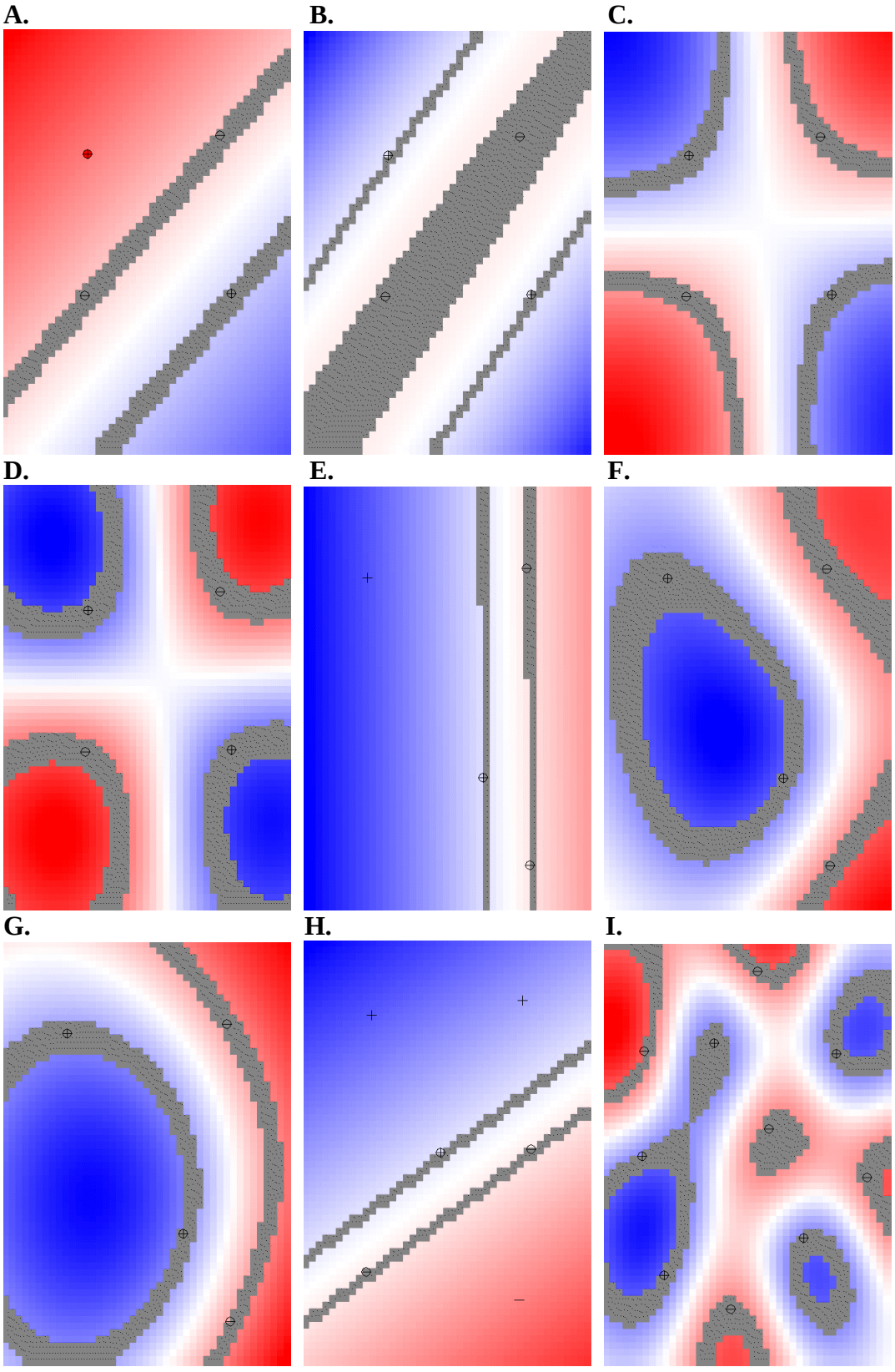
- The teacher has a model of what the student already knows.
- The teacher has a model of what the student is supposed to learn.
- Whatever the student learns can be put to use by the student.
- The student has enough computational capacity to learn what is taught.
- All of the above
- None of the above

In precedent-based learning (that is, the Macbeth and cup learning examples)

- Identification trees determine which features are relevant to classification.
- A teacher provides explanations and a student uses those explanations to form near misses.
- Learning is a by-product of solving problems posed by a teacher.
- The representation problem is eliminated through the use of universal grammar.
- All of the above
- None of the above

Kanwisher's work on fMRI methods demonstrate that fMRI

- Poses a hazard to subjects because of the intensity of X-rays required
- Has located dozens of recognition centers in the brain
- Provides temporal resolution down to the millisecond level
- Demonstrates the existence of a center activated by music
- All of the above
- None of the above


A.
B.
C.
D.
E.
F.
G.
H.
I.

n	-n log n	-n log n - (1-n) log (1-n)		n	-n log n	-n log n - (1-n) log (1-n)
0.00	0.00					
0.01	0.07	0.08		0.51	0.50	1.00
0.02	0.11	0.14		0.52	0.49	1.00
0.03	0.15	0.19		0.53	0.49	1.00
0.04	0.19	0.24		0.54	0.48	1.00
0.05	0.22	0.29		0.55	0.47	0.99
0.06	0.24	0.33		0.56	0.47	0.99
0.07	0.27	0.37		0.57	0.46	0.99
0.08	0.29	0.40		0.58	0.46	0.98
0.09	0.31	0.44		0.59	0.45	0.98
0.10	0.33	0.47		0.60	0.44	0.97
0.11	0.35	0.50		0.61	0.44	0.96
0.12	0.37	0.53		0.62	0.43	0.96
0.13	0.38	0.56		0.63	0.42	0.95
0.14	0.40	0.58		0.64	0.41	0.94
0.15	0.41	0.61		0.65	0.40	0.93
0.16	0.42	0.63		0.66	0.40	0.92
0.17	0.43	0.66		0.67	0.39	0.91
0.18	0.45	0.68		0.68	0.38	0.90
0.19	0.46	0.70		0.69	0.37	0.89
0.20	0.46	0.72		0.70	0.36	0.88
0.21	0.47	0.74		0.71	0.35	0.87
0.22	0.48	0.76		0.72	0.34	0.86
0.23	0.49	0.78		0.73	0.33	0.84
0.24	0.49	0.80		0.74	0.32	0.83
0.25	0.50	0.81		0.75	0.31	0.81
0.26	0.51	0.83		0.76	0.30	0.80
0.27	0.51	0.84		0.77	0.29	0.78
0.28	0.51	0.86		0.78	0.28	0.76
0.29	0.52	0.87		0.79	0.27	0.74
0.30	0.52	0.88		0.80	0.26	0.72
0.31	0.52	0.89		0.81	0.25	0.70
0.32	0.53	0.90		0.82	0.23	0.68
0.33	0.53	0.91		0.83	0.22	0.66
0.34	0.53	0.92		0.84	0.21	0.63
0.35	0.53	0.93		0.85	0.20	0.61
0.36	0.53	0.94		0.86	0.19	0.58
0.37	0.53	0.95		0.87	0.17	0.56
0.38	0.53	0.96		0.88	0.16	0.53
0.39	0.53	0.96		0.89	0.15	0.50
0.40	0.53	0.97		0.90	0.14	0.47
0.41	0.53	0.98		0.91	0.12	0.44
0.42	0.53	0.98		0.92	0.11	0.40
0.43	0.52	0.99		0.93	0.10	0.37
0.44	0.52	0.99		0.94	0.08	0.33
0.45	0.52	0.99		0.95	0.07	0.29
0.46	0.52	1.00		0.96	0.06	0.24
0.47	0.51	1.00		0.97	0.04	0.19
0.48	0.51	1.00		0.98	0.03	0.14
0.49	0.50	1.00		0.99	0.01	0.08
0.50	0.50	1.00		1.00	0.00	0.00