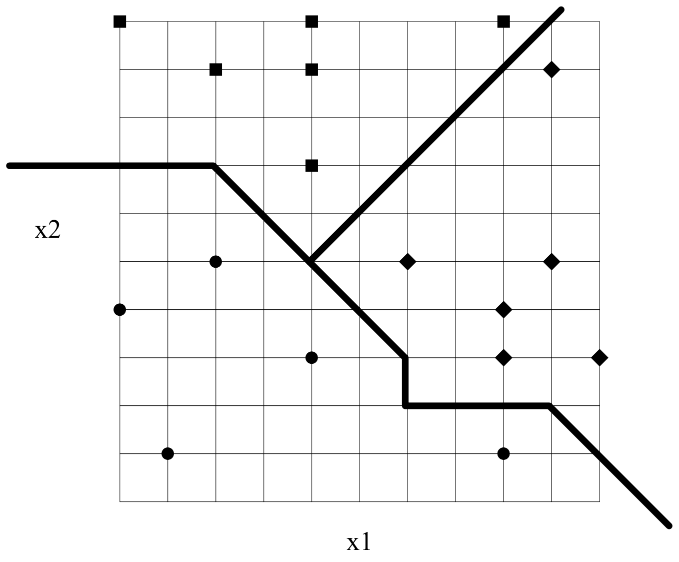
6.034 QUIZ 2

Fall 2001

Problem 1: Decision Boundaries (24 points)

In a nightmare, you find yourself working for a bank. Your boss, a Sloan graduate, tells you that there are three classes of customers. Your boss, a nut, believes that you can predict what sort of customer a prospect will turn out to be on the basis of two features, **x1** (cups of coffee drunk per day) and **x2** (lottery tickets purchased per week). He presents you with the following data. Samples from the three classes are represented as circles, squares, and diamonds.



Many people missed the exit angle on the right and the jog in the middle.

Part A (6 points)

Your first thought is to use the single-nearest-neighbor approach. ***Carefully*** draw the corresponding decision boundaries on the diagram above.

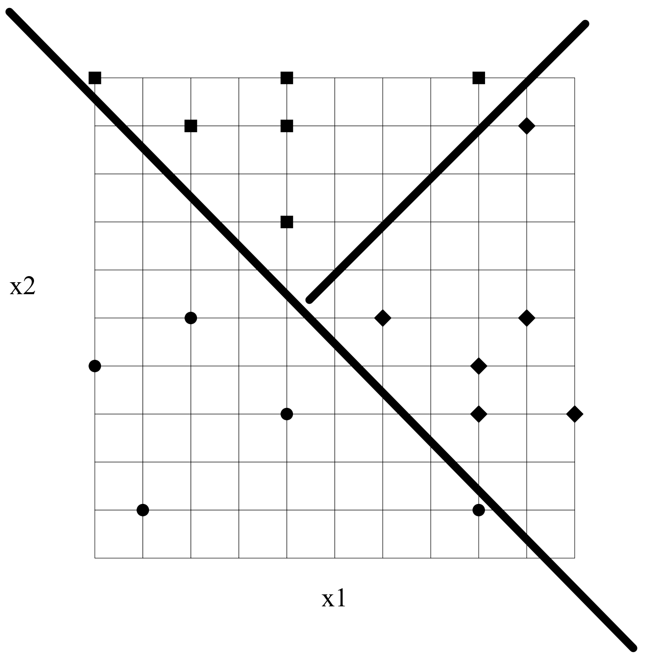
Part B (6 points)

You have never been a big fan of the nearest neighbor approach, so you decide you will have a look at the same data from the identification tree perspective. You decide to allow the standard disorder-based tree-building program to build an identification tree using the following four tests, each of which may appear 0 or more times. Note that the thresholds are to be determined by the standard disorder-based tree-building program:

x1 > Threshold selected by program for x1

x2 > Threshold selected by program for x2
x1 + x2 > Threshold selected by program for x1+x2
x2 – x1 > Threshold selected by program for x2-x1

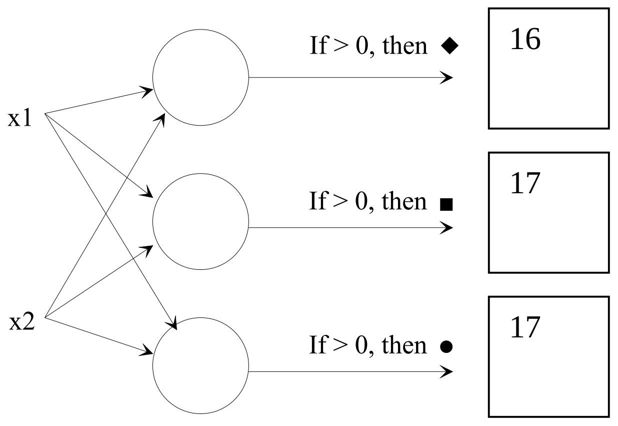
Carefully draw the decision boundaries produced on the diagram below.



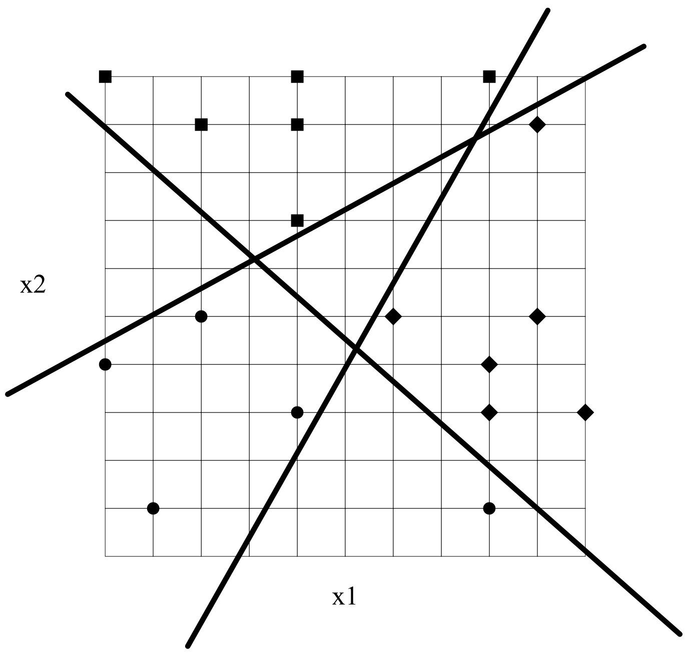
Here, the corrected answer uses only "derived attributes," not the original tests.

Part C (6 points)

Finally, it occurs to you to think about using the following neural net (each neuron has a threshold).



Determine the best the net can possibly do, after training. Express your conclusions in the boxes above by writing down the number of samples correctly analyzed, in the best case, by each of the three sample-identifying outputs. Note that the maximum number you can enter in any box is 17 (all examples correctly identified and all nonexamples correctly identified.)



Each neuron puts a line into the space. Two of the lines separate the sample data perfectly, and the third incorrectly handles just one sample.

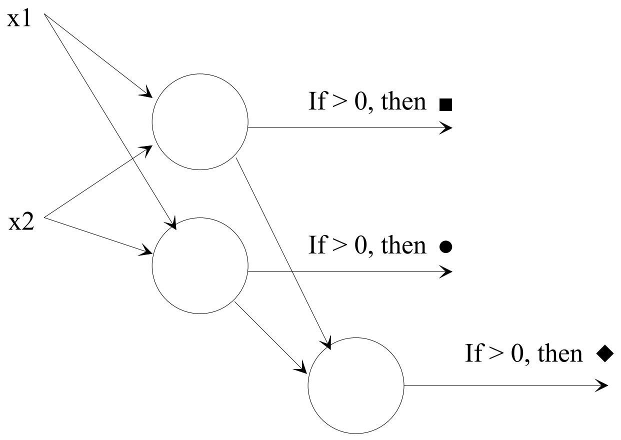
Part D (6 points)

Reflecting on your conclusions in Part C, decide how many ***<u>total</u>*** neurons you need in a network of your own design to handle the sample data perfectly in the sense that

- There is one output associated with each of the three classes.
- The output is > 0 only for samples belonging to the corresponding class; that is, all 17 samples are handled correctly for each of the three neuron outputs associated with a class.

$\boxed{3}$

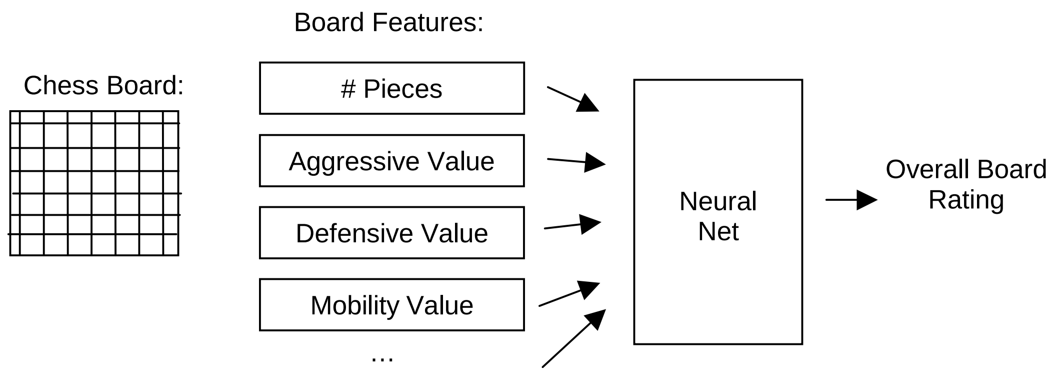
Sketch your network here, but ***do not calculate the weights.***



Because two of the neurons perfectly identify square and circle, diamonds can be handled by a neuron that has a positive output iff neither of its inputs are positive. This is most easily arranged if all neurons have sigmoid functions with extreme-value exponents that make them look like perceptrons.

Problem 2: Neural Nets and Genetic Algorithms (26 points)

While trying to design the ultimate chess-playing program, you come up with a dozen possible heuristic measures for a chessboard. You don't know which of these features are most valuable, or whether certain combinations work well together, but you need a single number telling you how good the board is in order to use alpha-beta. You decide to feed these features as inputs into some sort of neural net, using sigmoid threshold functions, hoping you can teach the net to evaluate chess boards well.



Part A: Backpropagation Problems (8 points)

You get an expert chess player to **rate 100 different chessboards**, and you use standard backpropagation to train the neural net on that data. **Circle all** of the following issues that could, in principle, limit your ability to develop the best possible chess program using this scheme. **Draw an X through all** of the answers that cannot. All choices are to be either circled or Xed.

[✓ A.] Backpropagation is susceptible to overfitting the data points, since you didn't use any cross-validation method.

✗ *B.* A neural network splits its input space using hyper planes, so it can only distinguish between boards that are completely good or completely bad. *Only perceptrons,with step function thresholds, produce "completely good or completely bad" outputs.*

[✓ C.] Backpropagation implements gradient descent, so your network may converge on a set of weights that is only a local optimum rather than the global optimum.

[✓ D.] The topology of your neural net might not be adequate to capture the expertise of your human expert.

Part B: Genetic Algorithm Problems (8 points)

Next, instead of backpropagation, you use a standard genetic algorithm to evolve weights for the network hoping that that your network and your chess expert will eventually agree. You use a population of 50 networks, and run the GA for 1000 generations. You test the result by personally playing chess against some evolved networks every 10 generations, and you find improvement steady from generation 0 to generation 100, but no change in playing ability after generation 100.

Circle all explanations that **could** account for this result. That is, circle all explanations that cannot be ruled out given what you have been told. **Draw an X through all** the answers that can be ruled out. All choices are to be either circled or Xed.

[✓ A.] You forgot to include crossover, so the population is just doing hill climbing to a locally optimal point.

[✗ B.] The mutation level is so high that no useful adaptations can accumulate. *If this were the explanation, there would have been no improvement for the first 10 generations.*

[✓ C.] In this case, the phenotype does not benefit from combining two genotypes with crossover, so crossover provides no benefit beyond mutation.

[✗ *D.*] A GA requires that individuals are represented as binary strings, and this means you cannot evolve neural network weights other than 0 or 1. *Gas not restricted to binary strings (cf human DNA) and anyway, anything can be encoded in 0s and 1s (cf computer.)*

Part C: Fitness Methods (10 points)

In this part of the problem, you continue to use a genetic algorithm, but now, **the fitness method** used to train the network **is entirely different.**

In particular, you decide to experiment with different methods of calculating the fitness for an "individual" in your population of neural networks. In all methods, **individual fitness** is the fraction of chess games that the individual wins, but you try different opponents. Identify which of the four graphs of **average population fitness** makes the most sense for each of the following methods:

Method 1:	10 games against a single **human expert** player.	C or D
Method 2:	10 total games against **10 different human players** in an on-line game system on the Internet. (Whoever is logged on.)	C or D
Method 3:	10 games against a single **commercial computer chess program** set to its highest difficulty mode.	C or D, And same as 1
Method 4:	10 games against **randomly selected individuals** in the current population of neural-net players.	A
Method 5:	49 total games, 1 against **every other individual** in the current population of neural-net players	B

1 and 2: We like D and C, but others with reasonable arguments have the reverse position.
3: Should be same as human expert, whatever the answer chosen was for that.
4: Same as five, but with some noise because of the randomly selected opponents.
5: When a population plays itself, half of the games will be won, and half lost.

Possible graphs for different measures of fitness
Generation (abscissa) ranges from from 0 to 1000, fitness (ordinate) from 0 to 1.0

Average Fitness vs. Time

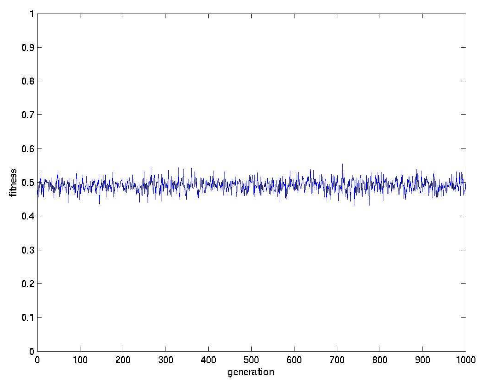


Figure A

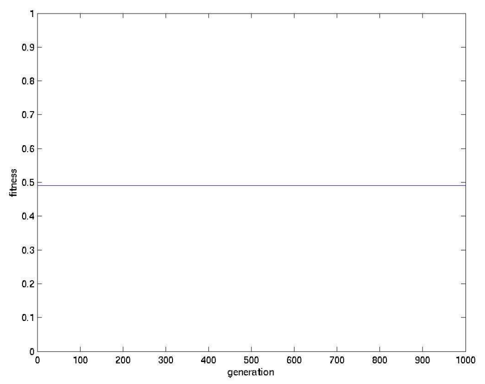


Figure B

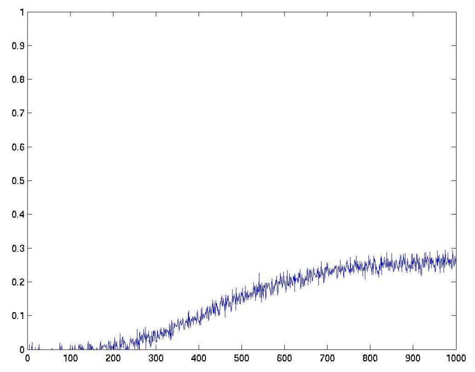


Figure C

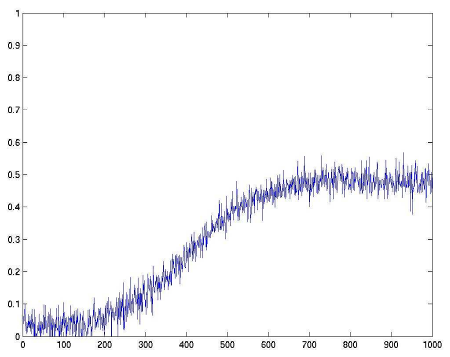


Figure D

Problem 3: Identification Trees (24 points)

You have been appointed as a research assistant at PacMan Institute of Technology, where you work for Pinky, Winky, Binky, and Hinky on research projects.

Binky asks you to build a Ghostbot that can collect and dispose of styrofoam foodtruck containers (a very useful task around P.I.T.). They should go into two different bins **P** (for Powerpellet Kitchen) or **S** (for Strawberry King).

You are given the training set (repeated on a tear off sheet for your convenience at the end of this examination). Note that "Maybe" is a possible result for the Unfinished? test.

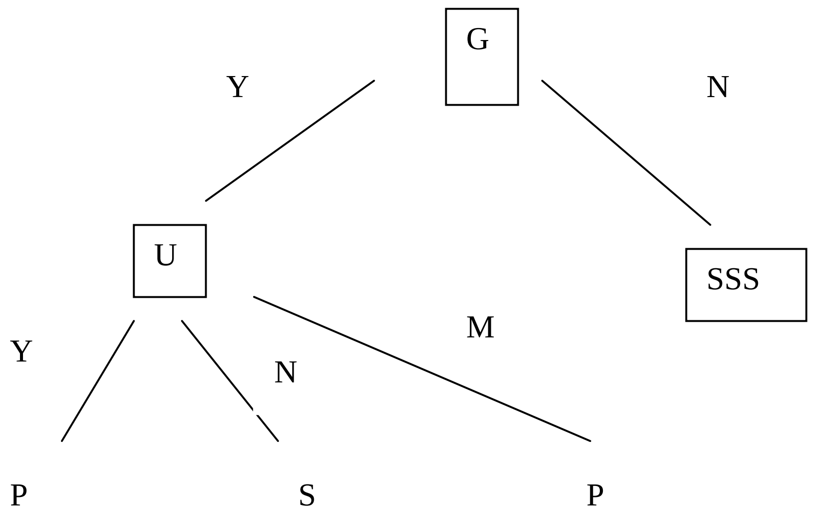
Training #	Colorful?	Unfinished?	Greasy?	Truck
1	Y	Y	N	S
2	N	Y	N	S
3	N	Y	N	S
4	Y	Y	Y	P
5	N	N	Y	S
6	N	Maybe	Y	P

And you are given the following table (also on the tear-off page), in case you forgot your calculator. If the number you need is not in the table, use the table to estimate the number you need.

	n	**$\log_2 n$**	**$- n \log_2 n - (1-n)\log_2(1-n)$**
	0.00	0.00	0.00
	0.05	-4.32	0.29
	0.10	-3.32	0.47
	0.15	-2.74	0.61
	0.20	-2.32	0.72
	0.25	-2.00	0.81
	0.30	-1.74	0.88
	0.35	-1.51	0.93
	0.40	-1.32	0.97
	0.45	-1.15	0.99
	0.50	-1.00	1.00
1/9	0.11	-3.17	0.50
1/8	0.13	-3.00	0.54
1/7	0.14	-2.81	0.59
1/6	0.17	-2.58	0.65
1/5	0.20	-2.32	0.72
1/4	0.25	-2.00	0.81
1/3	0.33	-1.58	0.92

Part A (12 Points)

Exhibit the identification tree produced by the standard disorder-based tree builder in the box below. You may abbreviate the tests as C, U, and G. (You may wish to show your work to ensure maximum partial credit).



Part B (5 Points)

Next, convert the tree from Part A into a set of equivalent IF-THEN rules**.** You may abbreviate the tests as C, U, and G.

If G=Y and U=Y
Then P

If G=Y and U=N
Then S

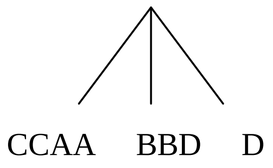
If G=Y and U=M
Then P

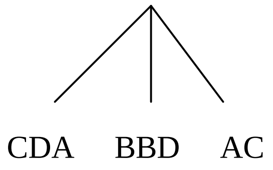
If G=N
Then S

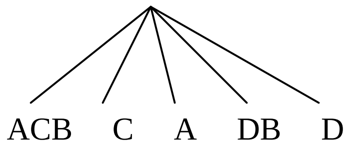
Each path to a leaf provides a rule. We allowed ors in the rules, but not elses, which don't make sense.

Part C (7 points)

Next, you are commanded to decide which the following is the **best** candidate as the **next branch** of a decision tree under a node that receives samples CCAABBDD. **Circle One, calculate the average disorder for the one you circle, and write that average disorder in the box below.**







Average disorder for best test:

First tr~~ee~~: =
1/2 x 1 + 3/8 x 0.92 + 0 = 0.8~~45~~ =

~~S~~eco~~n~~d tree: =
-3/8 x ~~(~~3 x 1/3 ~~log~~$_2$ 1/3) + 3/8 x 0.92 + ~~¼~~ x 1 =
.~~5~~92 + .3~~45~~ + .2~~5~~ =
1.18~~7~~ =

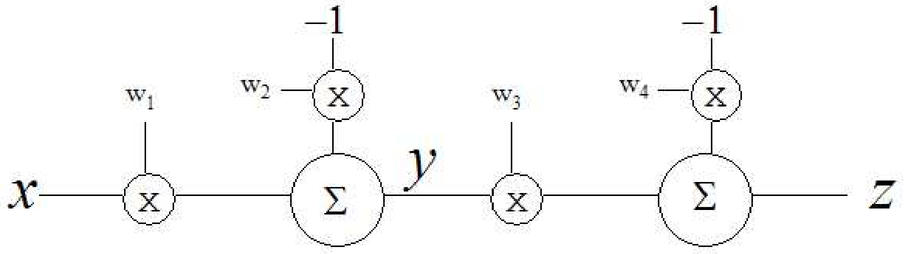
T~~h~~ird tree: =
1/~~4~~ x 1 + 0 + 0 - 3/8 x ~~(~~3 x 1/3 ~~log~~$_2$ 1/3) + 0 =
~~¼~~ - .~~5~~92 =
0.8~~4~~2 =
=

First and third trees have nearly the same disorder; full credit for circling either the first or third tree and getting disorder correct.

Problem 4: Neural Nets (26 points)

Part A (9 points)

You are trying to work out a basic one input, one output neural net problem but can't remember how to construct the neural net. You ask your friend (who took 6.034 two years ago) and his memory is fuzzy but he draws the following diagram.



This is similar to the neural net originally drawn in lecture, but ***your friend has missed the sigmoid thresholds***! You don't realize his mistake at first, and start to work through the mathematics ***with no threshold functions***.

- Write z in terms of y, w_3, and w_4.

$z = yw_3 – w_4$

- Write z in terms of x, w_1, w_2, w_3, and w_4.

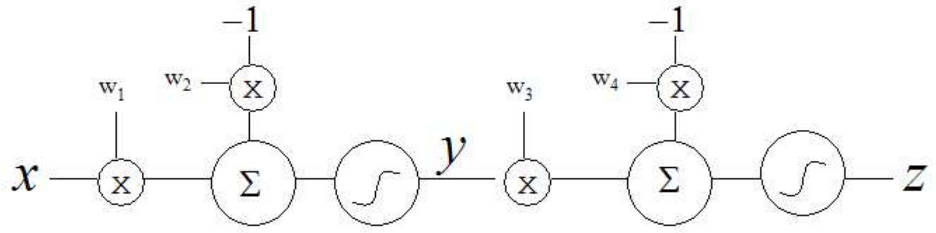
$z = (w_1\ x – w_2)\ w_3 – w_4$

- Find dE/dw_1, if the performance function, $E = ½ * (z – z^*)^2$, with z^* being the desired output.

$w_3\ x\ \ (z – z^*)$

Part B (7 points)

You realize the net in part A is wrong and consult with another friend who took the class during the past year. She realizes immediately that the sigmoid blocks are missing. She modifies the diagram above to include them.



Apparently she didn't do well in 6.034 either, because she tells you that the performance function, $E = ¼ * (z – z^*)^4$, with z^* being the desired output.

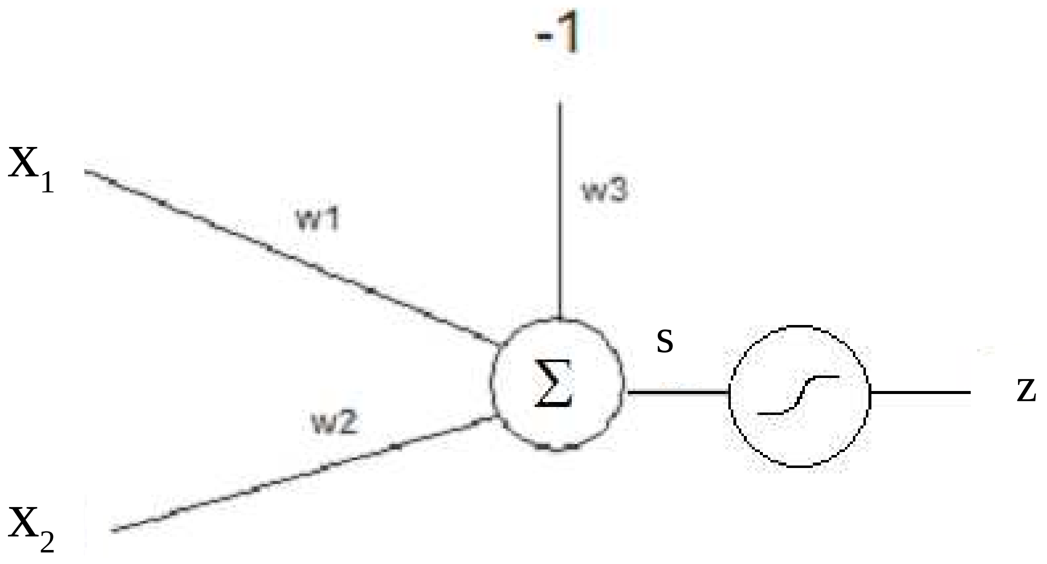
Find dE/dw_3, based on what she has told you.

Chain rule produces

$y\, z\, (1 – z)\ \ (z – z^*)^3$

Part C (10 points)

You have the following neural net with one sigmoid unit. After running a few steps, you have the weights set at $w_1 = 2$, $w_2 = 1$, $w_3 = 2$. You are using the ordinary performance function $E = ½ (z – z^*)^2$, with z^* being the desired output, and a learning rate of 4.



Let $s = x_1 * w_1 + x_2 * w_2 – w_3$, and then z = Sigmoid(s). A few values of the sigmoid function are:

S(-5) = 0	S(-4) = 0.02	S(-3) = 0.04	S(-2) = 0.12	S(-1) = 0.27	S(0) = 0.5
S(5) = 1	S(4) = 0.98	S(3) = 0.96	S(2) = 0.88	S(1) = .73	

Your next piece of training data is $x_1 = .6$, $x_2 = .8$, and $z^* = 0$.

What is the value of z that you get using forward propagation with the current set of weights and inputs.

0.8 x 1 + 0.6 x 2 – 1 x 2 = 0, *which goes into sigmoid, which produces* z = 1/2

Find δ= dE/ds

$z (1 – z) (z – z^*) = 1/8$

The rate, 4, did not enter into the calculations, because we did not ask you to compute weight changes.