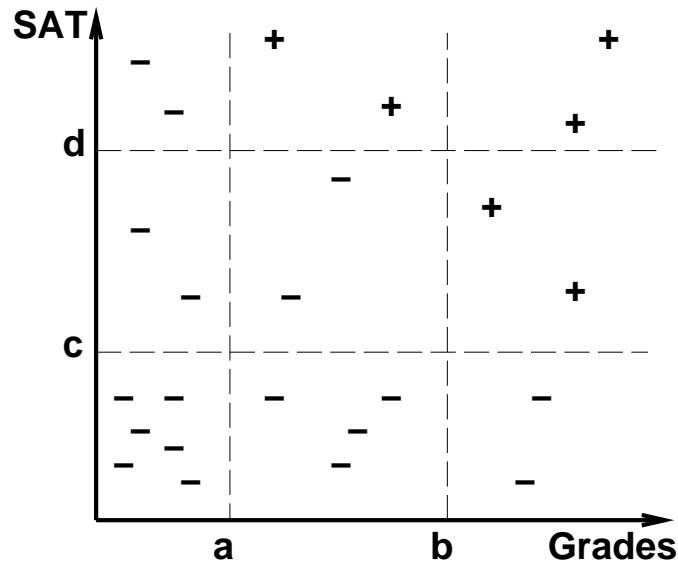


Problem 3 Identification Trees (20 points)

This problem contains independent parts, so **do not give up on the problem as a whole just because you give up on part of the problem.**

You are given the following admissions data from the MIT admissions office. A “+” indicates someone admitted and a “-” indicates someone not admitted. The x axis is grades (on a 0-100 scale) and the other is average SAT score (on a 0-800 scale):



In building an identification tree for this data, we will consider only the following four tests as candidates for the nodes:

1. $\text{SAT} > c$
2. $\text{SAT} > d$
3. $\text{Grades} > a$
4. $\text{Grades} > b$

Part A (4 points)

Compute the value of the average disorder measure for the test $\text{SAT} > c$:

Part B (4 points)

Assume that the test $\mathbf{SAT} > \mathbf{c}$ is chosen as the test at the top of the identification tree. Consider the remaining tests that could be used at the next level of the tree:

- $\mathbf{SAT} > \mathbf{d}$
- $\mathbf{Grades} > \mathbf{a}$
- $\mathbf{Grades} > \mathbf{b}$

1. Which one of these tests, if any, should be used at the next level of the tree for the branch in which the top-level test ($\mathbf{SAT} > \mathbf{c}$) is *false*. Indicate in one sentence the reason for your answer.

2. Now, for the branch in which the top level test ($\mathbf{SAT} > \mathbf{c}$) evaluates to *true*. Next to each test below, write its rank based on average disorder. That is, write a 1 next to the test with least average disorder, a 2 next to the next best, etc. If there is a tie, use the same rank for the tied tests. **Hint: you need not do detailed calculations to determine the correct answer.**

Test	Rank
$\mathbf{SAT} > \mathbf{d}$	
$\mathbf{Grades} > \mathbf{a}$	
$\mathbf{Grades} > \mathbf{b}$	

Part C (8 points)

Given your success so far, you decide to turn your horizons past college, to the financial future, and use ID trees to work on bankruptcy prediction.

From 10 features (synonym for tests), measured surreptitiously from credit card usage patterns, you propose to a big bank to “guarantee” that you can predict consumer bankruptcies 18 months in advance.

For some reason, the tenth feature, which you thought would be used prominently in the identification tree built by your ID tree program, is not deployed at all for the sample data you provide. Naturally, you suspect a bug, but in fact, when you pay your 6.034 recitation instructor big bucks for consulting, he provides an alternative explanation.

Check all of the following that your recitation instructor might have said, without destroying your faith in his grip on the material. That is, check all of the following that could explain why the tenth feature is not used:

The tenth feature usefully splits the sample data into groups, but it splits the sample data much like the ninth feature, but not quite as well. Therefore, the tenth feature is always dominated by the ninth feature.

The tenth feature is useful, but only in combination with the ninth feature. Because ID tree construction does no look ahead or test combinations, the value of the tenth feature is overlooked.

The values of the tenth feature are all tightly clustered around the mean. The feature can be useful, but you must first divide all values by the standard deviation of the values for the tenth feature in the sample set.

The tenth feature is not useful. However, a new, eleventh feature can be derived from the tenth feature, and that new eleventh feature is useful.

Part D (4 points)

You fail to find much use for this ID program in bankruptcy detection, but you decide to use the same system to do stock selection. Again you start with 20 features, presuming to divide “good” stocks from “bad” stocks.

There are two groups of stocks, Group A and Group B, representing two different company types (such as electronics and health care).

The features are binary. There are about 1000 samples for each of the two groups, Group A and Group B.

For stocks in Group A, you discover that about 10 features are used in just about every path leading from the root node to a terminal node containing “good” stocks or “bad stocks.” All 20 features are found in equal proportions in the tree.

For stocks in Group B, you discover that no more than 10 features are used in any path leading from the root node to a terminal node, and in many paths, the number of features is far fewer. The 20 features are found in **unequal** proportions in the tree.

“Hmmm,” says your expensive consultant. Reminds me of the overfitting problem that you often see arise in neural networks.

Check the correct answer:

He is more likely to be reacting to the tree built for Group A.

He is more likely to be reacting to the tree built for Group B.

He is equally likely to be reacting either tree.

He is not likely to be reacting either tree.

Problem 4 How they work (23 points)

The purpose of this problem is to test your understanding of various mechanisms at the level required to predict performance in the face of various perturbations.

This problem contains independent parts, so **do not give up on the problem as a whole just because you give up on part of the problem.**

Part A Neural Nets

A.1 (5 points)

You have purchased neural-net training software from MicroMonopoly. Not surprisingly, it is screwed up.

Instead of the *standard* threshold function,

$$l = \frac{1}{1 + e^{-m}}$$

they have provided an *altered* function,

$$l = \frac{1}{1 + e^{+m}}$$

which, when plotted, gives you the mirror image of the *standard* threshold function.

You are told that a network with the *standard* threshold function can be trained to perform a certain recognition task successfully with a given training set.

After a bit of thought, you conclude that there is a set of weights that will enable an identically arranged network to perform the recognition task with the *altered* threshold function.

Next, you learn that the MicroMonopoly training software uses exactly the training formulas given in the book:

$$\begin{aligned}\Delta w_{ij} &= r o_i o_j (1 - o_j) \beta_j, \\ \beta_j &= \sum_k w_{jk} o_k (1 - o_k) \beta_k \text{ for nodes in hidden layers,} \\ \beta_z &= d_z - o_z \text{ for nodes in the output layer.}\end{aligned}$$

Check the box next to the best expectation; place an X in the other boxes:

The training software won't work; in principle, networks that use the *altered* threshold function can't be trained using the given training formulas.

The training software may work, but, on the other hand, using initial weights and a rate constant that would be appropriate for the correct threshold function, it may take much longer to train the network, and the training may not converge on a suitable set of weights at all.

Using the initial weights and a rate constant that would be appropriate for the correct threshold function, the software will work just about as it would if there were no bug.

A.2 (5 points)

You call customer "support" and the person who you talk to recommends that you save all files, reformat your disk, and then reload the operating system and all your applications.

Fortunately, before you descend into that hell, "Network 98" is released. Now, the threshold function they provide is:

$$l = \frac{1}{1 + e^{-1,000,000 \times m}}$$

Check the box next to the best expectation; place an X in the other boxes:

The training software won't work; in principle, networks that use the *altered* threshold function can't be trained using the given training formulas.

The training software may work, but, on the other hand, using initial weights and a rate constant that would be appropriate for the correct threshold function, it may take much longer to train the network, and the training may not converge on a suitable set of weights at all.

Using the initial weights and a rate constant that would be appropriate for the correct threshold function, the software will work just about as it would if there were no bug.

A.3 (5 points)

Finally, you do a work around that enables you to train the network with the following, normal, threshold function:

$$l = \frac{1}{1 + e^{-m}}$$

Your trained network is to be used in a handheld computer game, so you hand the training results over to a circuit designer who burns the network into a VLSI chip.

Unfortunately, the designer works for a MicroMonopoly spinoff. The threshold functions burned into the chip reflect a previous release of MicroMonopoly Network 98:

$$l = \frac{1}{1 + e^{-1000m}}$$

Even more unfortunately, you learn it would cost a fortune to alter most of the characteristics of the chip, but a few things can be done inexpensively.

Check the box next to the right thing to do; place an X in the other boxes:

Multiply all inputs to first-layer nodes by 10^3 .

Multiply all weights by 10^3 .

Multiply all final-layer outputs by 10^3 .

Multiply all inputs to first-layer nodes by 10^{-3} .

Multiply all weights by 10^{-3} .

Multiply all final-layer outputs by 10^{-3} .

None of the above.