Question 1:

a. Your target concept is an element of your hypothesis space

a. False. Target concept maybe an element of your hypothesis space. You hope it is an element of your hypothesis space usually, but it doesn’t have to be.

b. Suppose we are given a decision tree, let’s name it Fred, we generate a training set consistent with that tree and then apply decision tree learning to build a new tree. As the training set size goes to infinity, the learning algorithm new tree will be the same as the original tree

a. False. Didn’t say where the tree came from, so you don’t even know if it is generated from the same algorithm that you would use to do the tree. Example: A or B. Two different trees can match that extremely well.

c. In the algorithms we showed in class, we never tested the same attribute twice along one path of the decision tree. Someone argued that testing the same attribute twice is inefficient and unnecessarily. This argument is correct whether our attribute is discrete or continuous.

a. False. Continuous attributes, we don’t just test them and have a branch on every possible continuous value. Instead we make some kind of tests on it, the variable is less than 5, true or false? And we can later test the same variable, not with the same test, but have another test like X less than 2.

d. The output of a boosting algorithm that learn using decision strums, i.e., a decision tree with only 1 node, can be converted to an ordinary decision tree in a straightforward and boring way.

a. False. Linear vs Non linear. They are not related to each other.

e. In general, when choosing a hypothesis space, a very large hypothesis space is preferable to a smaller one.

a. False. Bias variance. Big hypothesis class means we have low bias because what we are trying to learn is most likely in there, which also means it will have very high variance. We need a ton of data to figure out which hypothesis in that class is the one we want.

b. Small hypothesis class has low variance but very high bias.

c. Best hypothesis space: The one just has the target function in it.

f. Given a learning problem with over one thousand attributes, only a few of which are probably relevant. You should use a nearest neighbor method instead of a decision tree learning method.

a. False. Amount of data we would need to discover the variables are irrelevant for knn would be pretty big because the fundamental assumption of knn is that all the variables are equally important, assuming you have a normal born eculidean function.

g. One of the powerful advantages of the boosting algorithm is that it does not overfit.

a. False. (kind of true). Really resistant to overfit, but there is a case to force boosting to overfit. Usually it doesn’t, but you can make it do so. Example: You have a learning algorithm that is self-overfit and perfectly capture the training data in the very first round.

h. Bayesian network can represent any propositional sentence (Booleans symbols and operators).

a. True. You can represent AND, you can represent NOT, you can represent OR, with that you can represent anything.

i. Given a network of perceptron, each perceptron uses linear activation function, i.e. each unit j, the output is some constant c sub j, times the weighted sum of its input, with no threshold. One can construct a single unit perceptron that computes the same function.

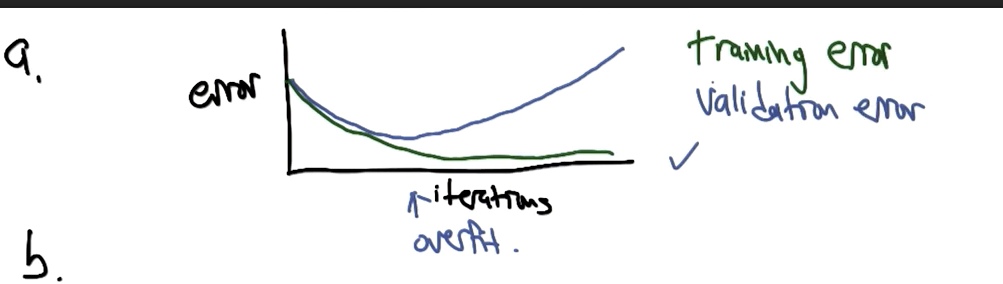
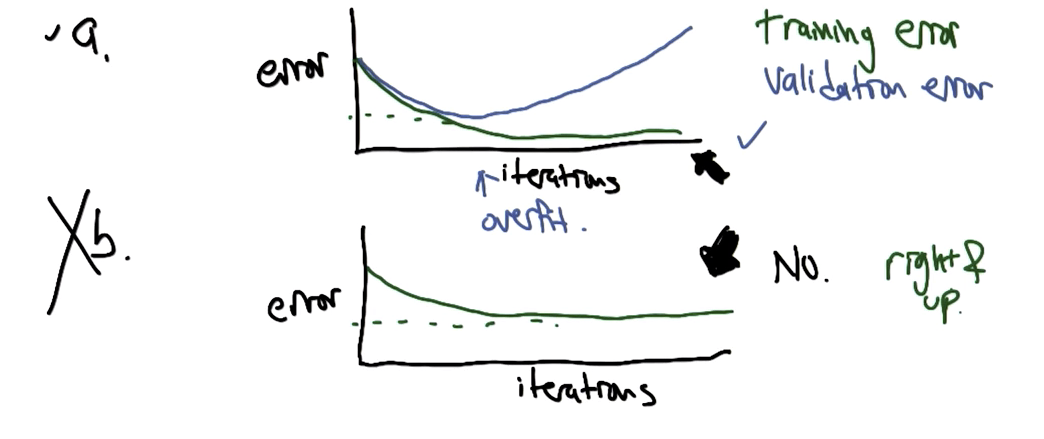
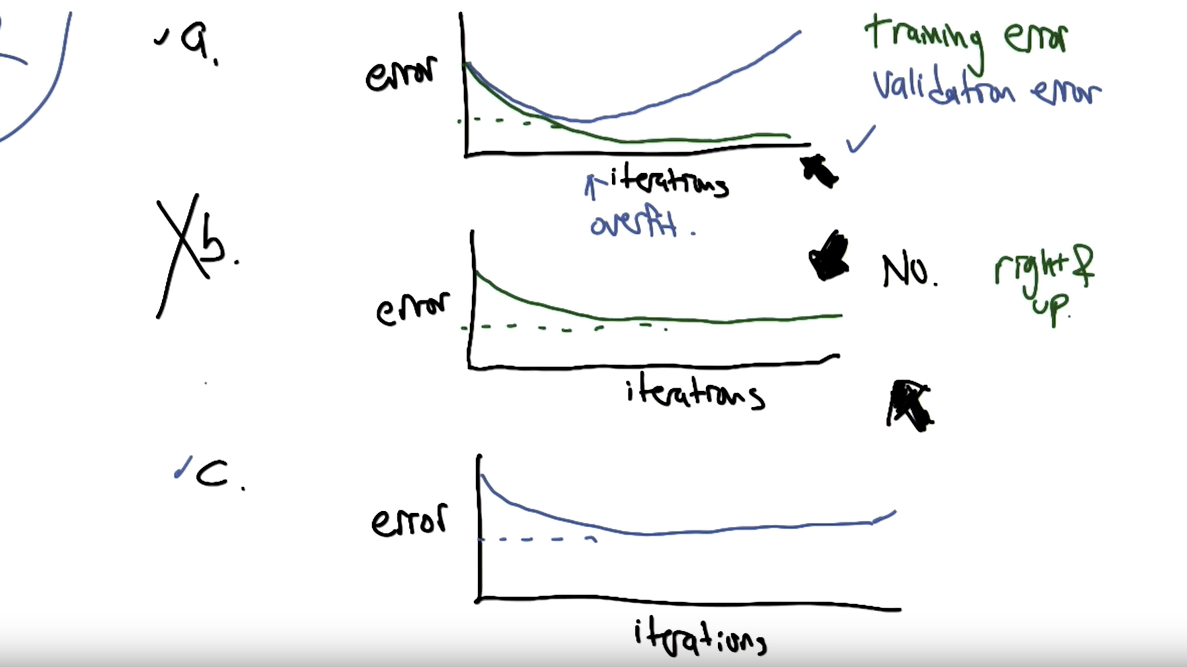
a. True. A linear combination of linear things is still linear so we can collapse it all down to one linear layer.

j. Despite superficial similarly graphs, Bayesian network and dependency trees are otherwise unrelated.

a. False. A dependency tree is just a special case of Bayesian network. It is a Bayesian network that is a tree.

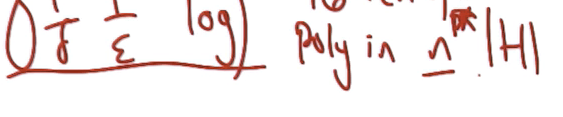
Question 2:

Imagine we want to train a neural network using back propagation, in order to determine the amount of training a simple validation technique is employed. Specifically, we randomly split the available samples into two equal sizes apart, one for training and one for validating. (We have a testing set and training set).

1. Sketch a plot showing the qualitative behavior of the network on the training and validation sets. Make the X axis the number of training iteration and the y axis the error. Presumably you should have two curves, one for the error on the validation set, and one on the training set. Describe the features of the curves.
   1. Training error early on will be high, the more iterations we run, the better we do, error goes down. In some kind of epsondote. Epsontoding is one important thing, starting high and getting low. Maybe go to zero, may be it doesnt, depends on the amount of data.
   2. Validation error also gonna start high, but will drop too, and then at some point, it is gonna start going up. Unfortunately we have overfit where we are changing the network in the way that it is very appropriate for the training data but that certainly not generalize well for the validation error.
   3. Probably would have a slightly bigger gap between the two lines, and probably will start overfitting a bit farther to the right.
   4. 
2. Now imagine that you suddenly have ten times as much as data you had before, but you are using the same network structure and learning algorithm. Basically just increasing data, would you expect the training curve to be different? If so, sketch out the old and the new curve and explain your reasoning.
   1. Yes. With more training data, we have more training error. Draw something epsidoe higher. It will epsidoe at the same level where the other one diverged. Because that was the one actually fitting the real data.
   2. Qualitatively you would expect the training error, it depends on your data, but in general you expect basically the training curve to be shifted over to the right and up a little. The way to think is, what if I have all the data, then I am going to get more error, because I get more data, and eventually will get to the best you can get to, you can’t really overfit.
   3. 
3. Would you expect the validation curve to be different as you got more data? And if so sketch both the old and new curves and explain your reasoning.
   1. Because we have so much data, we are not going to be overfitting, so we are not going to get the initial turn around, certainly not so soon. So I expect to stick with the green line. Get more data, your validation error will do better, and you will overfit later if at all.
   2. I would expect Graph A the validation curve to much more closely follow the training curve. It can still overfit, but as the amount of data goes to infinity, it won’t. But more to the point the point in which you overfit will happen later and later and later.
   3. 

Question 3:

Consider the hypothesis class H sub S sub 2, that is symmetric depth to decision trees over N boolean variables. Each hypothesis in H sub S sub 2 is a decision tree of depth 2 that is a tree with 4 leaves all distance 2 from the root, where the left and right child of the root must be the same variable. You have the example of there right for you. Assume that there were only 2 classes, (output Yes or No).

1. As a function of N, how many semantically distinct trees are there in H sub S sub 2. Two trees are semantically distinct if they look different, even if they compute the same function. Explain your answer.
   1. N different choices for the top node. n(n-1) for the next level of node. Each leaf can be true or false, so there are 16 n(n-1). (or 2^4 n(n-1))
2. Give a bound for the number of examples needed in the Pac Model to learn any target concept in H sub S sub 2 for some error absalone and confidence delta.
   1. 
3. As a function of N, how many syntactically distinct trees are there in H sub S sub 2. Two trees are semantically distinct only if they compute different functions. Explain your answer.
   1. This is really functions of no more than 2 variables, as opposed to function of 2 variables. This is then all functions of 0 variables, all functions of 1 variables and all functions of 2 variables. And you add all those together. There are 2 functions of 0 variables: True and False. There are 2n functions of 1 variables: Either keep it or negate it. There are n truce 2 ways of functions of exactly 2 variables (where they both actually matter): Ands and Ors. 16 possible outputs from 2 variables. There are not all True, not all False, not matching the leaf nodes on both sides (4), rule out TTTT, FFFF, TTFF, FFTT.
   2. Get full credit if you say 16. The answer is actually 10. If you look at the set of variables of all those functions, and eliminate the ones and ignore them, you end up with 0. Figure out the answer of a and the answer of c is different.
4. Give a tighter bound for the number of examples needed to learn any target concepts \*\*\*\*\* (???)
   1. You just have to remember it is polynomial, 1 over delta, 1 over absalon, and the log of the size of the hypothesis space. (Using the C’s bound)

Question 4:

We have used a greedy one step lookahead for growing decision trees. We pick a new node to add by looking at the remaining attributes and choosing the one with the greatest information gain. Lets call it gain of a, d. A is the attribute and d is the dataset. Imagine that we instead use two step lookahead, rather than pick a single node to add to our tree, we allow to choose a node and its children simultaneously, we define a new information gain measure, gain sub 2 (a, l, r, d) that computes the information gain attribute combine with the information gain of the best children that could follow it. In other words, we find a, l, r that maximizes that function which is just the sum of the gain of (a, d) + gain of (l \* positive data) + gain of (r \* negative data). Gain 2 is reasonably defined when a, l, r, d are empty. This defines a mini tree, that we add to our decision tree. So we are adding mini trees now instead of single nodes, so

1. Is this a richer hypothesis class, why or why not?
   1. No. Same hypothesis class. It is just describing another algorithm. Any decision trees could be returned as opposed to all decision trees whose lengths are even or something.
2. What are some advantages or disadvantages of this alternative approach?
   1. Advantage: lookahead can search pass local optima, like parity issue. Regular decision tree can get stuck in an XOR. This would have no trouble at all because it would be testing all the possible combinations that discover. Less likely to get stuck
   2. Disadvantage: Before we look at all the attributes, now we look at all the attributes squared. Computationally expensive.
   3. Gain n -> Better for lookahead, Bad for Computationally expensive.

Question 5: Briefly compare Genetic algorithm, Simulated Annealing, Randomized Hill Climbing, and MIMIC. What does each one do, for what sort of problems does each suited, what are their strengths and weaknesses. Be comprehensive but be concise.

1. They are all randomized. They all take a fitness function or optimization function as input. They try to protect the things that give the biggest output as output.
2. GA is model after evolution, natural selection and looks at a population of individuals. Measure their fitnesses, and then decide which one to cross over to make individuals based on which ones are successful. Like a reproduction phase, reproduce the successful things and combine them. The big thing that comes out from reproduction of cross over. Can be mutation without crossover, because we have a population.
3. SA: Similar to hill climbing. you have an individual value, X. Look at neighbors of X to see what the fitness values are for them. If they are better, if you find one that is better, then move to the neighbor. If it is worse, then we have some probabilistic function that we say flip a coin, with some probability, we go, we move to that state even though it is worse, and otherwise we just don’t move, we stay where we were. You always go to better places, and you only sometimes go to worse ones. The annealing part is saying that you wanna change the probability over time, started off rather high and over time it gets lower so you are doing something more like straight up hill climbing but only after you have kind of browser around the space little bit to find out where the good things are.
4. RHC: similar to hill climbing. Take uphill steps, so look at neighbors if neighbors have a better score you go there, until you can’t go any further. If all the moves from the current state are downhill, then we rerandomize, we basically pick a new starting point, some places in our state space. It is like hill climbing but you never go some places worse. Randomize restart, Start in a random place.
5. MIMIC: kind of simulated to GA but with a single individual, represent each moment of time is actually a probability distribution, the population is represented as a probability distribution. We generate some individuals from that probability distribution and we find which ones have high scores, and to generate the next probability distribution, we can either estimate the probability, in particular, you can represent the probability distribution one attribute at a time, but you can actually do better by finding a tree of conditional probabilities and that turns out to be something you can do efficiently by running a shortest path algorithm. Oh no sorry, minimum spending tree.
6. Hill Climbing is good when it is nearly the base of relatively tall but relatively narrow hill. The diameter of the transition graph is relatively small. You can actually get to the peak with a relatively few steps and there is not a lot of misleadings, local optima.
7. SA is better getting unstuck from things without having to completely rerandomized. Stay in the same basic zone but actually shake itself loose and hook something nearby. Randomize hill climbing gets stuck, it just forgets everything, SA can remember where that it is in the region of something good and it can continue to search there.
8. MIMIC allows you to actually explicitly represent the neighborhood where SA, the neighbors are the things that are one move away from you. MIMIC has the probability distribution over all possible individuals and so that probability distribution can be relatively narrow, so it tends to produce the same kind of things or it could be broader in some places. So it actually kind of can capture a variable size of basin of attractions. All these expressive powers you get from MIMIC has a computational cost, it is worth it exactly in the cases where the evaluation function, evaluating a fitness function is really expensive. So it worths putting all these computation energy into learning as much as you can for every single fitness functions evaluation. By contrast, evaluating the fitness function is free, then you might as well do something like SA because you can evaluate every infinite points for zero cost. Have to be more than free, but less than expensive, maybe SA makes more sense than MIMIC.
9. SA has a temperature and anneals every time and tells you how likely you are to go to some places that dont get stuck.
10. GA has a population as of doing parallel bean search at the same time, sharing info across the searches.
11. MIMIC is like GA, instead of having a population, you have a probability distribution.
12. Start explaining from RHC , move to SA, move to GA , move to MIMIC

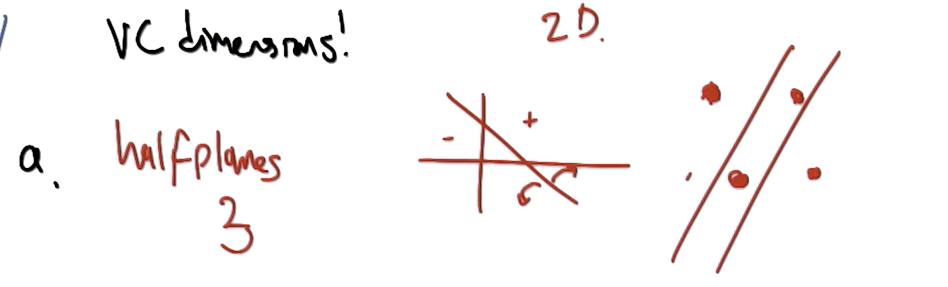
Question 6: Several algorithms that we studied are binary classifiers, i.e., they tell you whether an instance belongs to a specific class or not. This is in contrast to a multi way classifier that which of many classes an instance in.

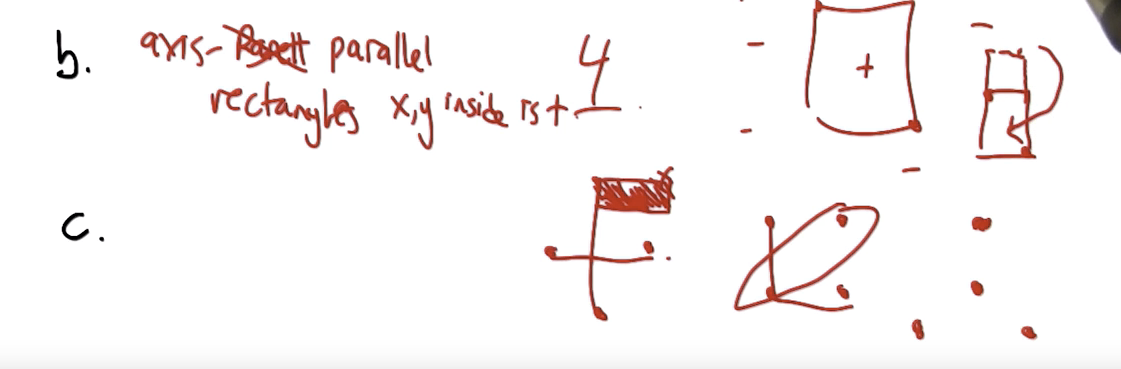
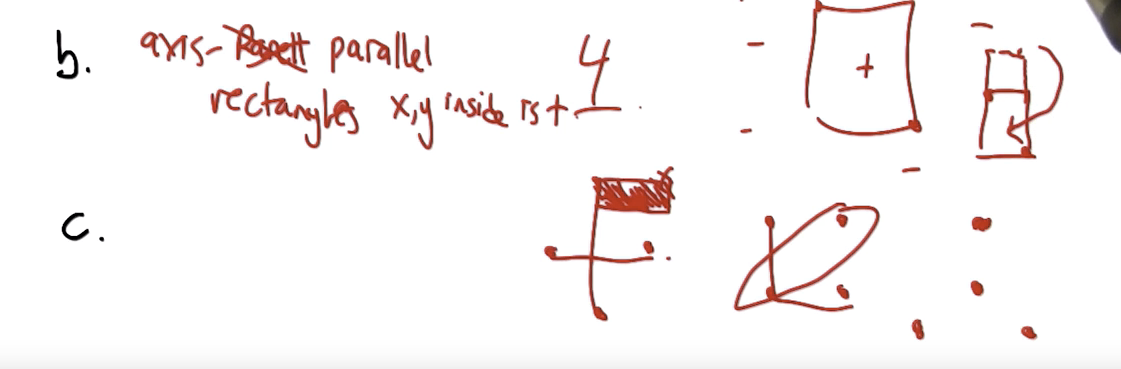
1. How might you use these binary classifiers to distinguish multiple classes? Note: You should not modify the algorithms itself, instead consider how you might use it as a component inside a larger algorithm. Do you think your approach would work well?
   1. I am gonna do black vs the other 3. And if I say black, then I will take black, if it is the other 3, then I will do brown vs the other 2, then I will do red vs green, which you are making assumptions of what is similar or what is dissimilar. Another version which might be worse, black or not black, brown or not brown, red or not red, green or not green. If your output is something other than just a straight 0, 1, but is a number like a confidence, then you could basically then do voting. There are gonna be problems with it that you have to deal with.
2. Suppose you wanted to test the performance of one of these classifiers you come up with, you write some code and come up with their predictive accuracy in whatever way you like, you get a very high accuracy but chipmunk points out that the results may not actually be that impressive. What are some baseline algorithms to compare the algorithms performance against and convince chipmunk? Describe two such baselines and justify the conclusion in your evaluation.
   1. Go with base rates, the majority classifiers. (Said whether the problem is easy or hard, different from whether your algorithm is doing well or not, or doing better than some alternatives)
   2. Pick any stated algorithms that tends to work well (known to be really good) and compare. Compare to SVM or boosting or whatever is currently trending.

Question 7:

Brief explanations on VC dimension questions to justify the answer. It is sufficient to have a simple explanations. Consider the space of instances corresponding to all points in the plane, x , y. Give the VC dimensions of the following hypothesis spaces. Briefly explain your answers.

1. The set of all lines, by which half planes.
   1. 3. The definitions of VC dimensions. The size of the largest set that we can shatter. I feel like we can shatter 3. It is a half plane by the way. You can be on either side. You can choose which side to shatter. It can’t be shatter 4.
   2. 3 degrees of freedom, slope, intercept and which side is positive.



1. The set of all axis parallel rectangles in the plane, where points x and y inside the rectangle, and let’s include the edges being inside the rectangle are classified as positive.
   1. 4. I feel like, in terms of degrees and freedom, we have got 4 of them. Try to avoid making any pair of axis.
   2. 4 degrees of freedom is the bottom line of the answer.
   3. If you have a 5th point, some point is going to be the interior of some axises parallel complex hole, you never can label that one negative and label other ones positive. If you have a fifth point, one of those 5 points must be in the interior of the axises parallel rectangle that you draw to cover another 4.
   4. 
2. It is the set of all rectangles in the plane, where point x, y inside the recentagle are classified as positive. This is exactly like the hypothesis space that we just talked about, except that you are allowed to rotate the rectangle anyway you wish. For this problem, an argument for your lower bound is acceptable. You don’t need to come up with the argument for the upper bound.
   1. degrees of freedom. (x, y) in one of the corner. Same as the other one, except there is also an angle. So it is 5 degrees of freedom.
   2. You no longer stuck on the axises, as you can turn the rectangle. Basically you make a pentagon, and you just rotating them around and make it work for five. I get another degree of freedom.
   3. Answer is actually 7, but I don’t suppose anyone to figure this out.
   4. 

Question 8:

1. Why use kernel functions in conjunction with a support vector machine?
   1. SVM alone are problematic because they only can do linear separations. You want to have a more expressive class, kernels give you that. Give you effective of having additional dimensions that can use to split up data. SVM needs it
   2. Kernels fits because you don’t have to redo the algorithm.
   3. You have a really nice way to stick all your domain knowledge there.
2. Consider the dataset that is below, place a y somewhere on the diagram indicate the position of a new positive instance that would change the current set of support vectors.
3. Now draw somewhere for me, a new negative point that would not change the support vectors.
   1. 