

10-701 Final Exam, Spring 2007

1. Personal info:
 - Name:
 - Andrew account:
 - E-mail address:
2. There should be 16 numbered pages in this exam (including this cover sheet).
3. You can use any material you brought: any book, class notes, your print outs of class materials that are on the class website, including my annotated slides and relevant readings, and Andrew Moore's tutorials. You cannot use materials brought by other students. Calculators are allowed, but no laptops, PDAs, phones or Internet access.
4. If you need more room to work out your answer to a question, use the back of the page and clearly mark on the front of the page if we are to look at what's on the back.
5. Work efficiently. Some questions are easier, some more difficult. Be sure to give yourself time to answer all of the easy ones, and avoid getting bogged down in the more difficult ones before you have answered the easier ones.
6. Note there are extra-credit sub-questions. The grade curve will be made without considering students' extra credit points. The extra credit will then be used to try to bump your grade up without affecting anyone else's grade.
7. You have 180 minutes.
8. Good luck!

Question	Topic	Max. score	Score
1	Short questions	21 + 0.911 extra	
2	SVM and slacks	16	
3	GNB	8	
4	Feature Selection	10	
5	Irrelevant Features	14 + 3 extra	
6	Neural Nets	16 + 5 extra	
7	Learning theory	15	

1 [Points] Short Questions

The following short questions should be answered with at most two sentences, and/or a picture. For the (true/false) questions, answer true or false. If you answer true, provide a short justification, if false explain why or provide a small counterexample.

1. [points] Your billionaire friend needs your help. She needs to classify job applications into good/bad categories, and also to detect job applicants who lie in their applications using density estimation to detect outliers. To meet these needs, do you recommend using a discriminative or generative classifier? Why?

for density estimation,
need $p(x|y)$

2. [points] Your billionaire friend also wants to classify software applications to detect bug-prone applications using features of the source code. This pilot project only has a few applications to be used as training data, though. To create the most accurate classifier, do you recommend using a discriminative or generative classifier? Why?

based on convergence properties and some experimental observations

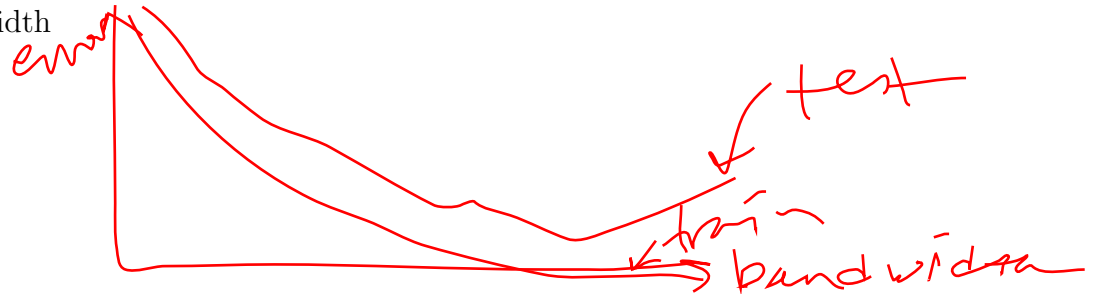
3. [points] Finally, your billionaire friend also wants to classify companies to decide which one to acquire. This project has lots of training data based on several decades of research. To create the most accurate classifier, do you recommend using a discriminative or generative classifier? Why?

4. [points] Assume that we are using some classifier of fixed complexity. Draw a graph showing two curves: test error vs. the number of training examples and cross-validation

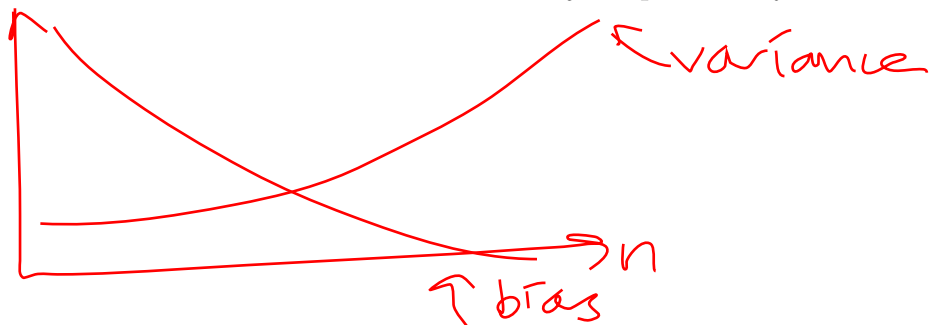
error vs. the number of training examples.



5. [points] Assume that we are using an SVM classifier with a Gaussian kernel. Draw a graph showing two curves: training error vs. kernel bandwidth and test error vs. kernel bandwidth



6. [points] Assume that we are modeling a number of random variables using a Bayesian Network with n edges. Draw a graph showing two curves: Bias of the estimate of the joint probability vs. n and variance of the estimate of the joint probability vs. n .



7. [points]

(a) Both PCA and linear regression can be thought of as algorithms for minimizing a sum of squared errors. Explain which error is being minimized in each algorithm.

PCA: $\arg \min_{\mathbf{u}} \left(\mathbf{x} - \sum_{i=1}^k (\mathbf{x} \cdot \mathbf{u}_i) \mathbf{u}_i \right)^2$ "reconstruction error"

Lin reg: $\arg \min_{\beta} (\mathbf{y} - \mathbf{x}\beta)^2$ "residual" error

8. [points] A long time ago there was a village amidst hundreds of lakes. Two types of fish lived in the region, but only one type in each lake. These types of fish both looked exactly the same, smelled exactly the same when cooked, and had the exact same delicious taste - except one was poisonous and would kill any villager who ate it. The only other difference between the fish was their effect on the pH (acidity) of the lake they occupy. The pH for lakes occupied by the non-poisonous type of fish was distributed according to a Gaussian with unknown mean (μ_{safe}) and variance (σ_{safe}^2)

and the pH for lakes occupied by the poisonous type was distributed according to a different Gaussian with unknown mean (μ_{deadly}) and variance (σ_{deadly}^2). (Poisonous fish tended to cause slightly more acidic conditions).

Naturally, the villagers turned to machine learning for help. However, there was much debate about the right way to apply EM to their problem. For each of the following procedures, indicate whether it is an accurate implementation of Expectation-Maximization and will provide a reasonable estimate for parameters μ and σ^2 for each class.

- (a) Guess initial values of μ and σ^2 for each class. (1) For each lake, find the most likely class of fish for the lake. (2) Update the μ and σ^2 values using their maximum likelihood estimates based on these predictions. Iterate (1) and (2) until convergence. *It'll do OK, if we give sensible enough μ, σ^2 initial values*
- (b) For each lake, guess an initial probability that it is safe. (1) Using these probabilities, find the maximum likelihood estimates for the μ and σ values for each class. (2) Use these estimates of μ and σ to reestimate lake safety probabilities. Iterate (1) and (2) until convergence. *OK. This is the same as (a) after the first M-step*
- (c) Compute the mean and variance of the pH levels across all lakes. Use these values for the μ and σ^2 value of each class of fish. (1) Use the μ and σ^2 values of each class to compute the belief that each lake contains poisonous fish. (2) Find the maximum likelihood values for μ and σ^2 . Iterate (1) and (2) until convergence.

This can be stuck at the initial μ, σ^2 :
In the E-step, we'll get

$$P(\text{safe} | x) = \frac{P(x|s)P(s)}{P(x|s)P(s) + P(x|poison)P(p)}$$

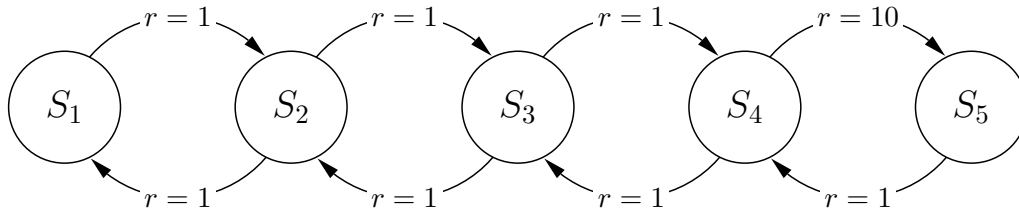
same since $\mu_s = \mu_p, \sigma_s = \sigma_p$ *assume equal*

$$= \frac{1}{2}$$

In the M-step, μ, σ will not change since we're again letting them be calculated from all lakes (weighted equally)

2 [points] Reinforcement Learning - *few!*

Consider the following Markov Decision Process:



We have states S_1 , S_2 , S_3 , S_4 , and S_5 . We have actions *Left* and *Right*, and the chosen action happens with probability 1. In S_1 the only option is to go back to S_2 , and similarly in S_5 we can only go back to S_4 . The reward for taking any action is $r = 1$, except for taking action *Right* from state S_4 , which has a reward $r = 10$. For all parts of this problem, assume that $\gamma = 0.8$.

1. What is the optimal policy for this MDP?
2. What is $V^*(S_5)$? It is acceptable to state it in terms of γ , but not in terms of state values.
3. Consider executing Q -learning on this MDP. Assume that the Q values for all $(state, action)$ pairs are initialized to 0, that $\alpha = 0.5$, and that Q -learning uses a greedy exploration policy, meaning that it always chooses the action with maximum Q value. The algorithm breaks ties by choosing *Left*. What are the first 10 $(state, action)$ pairs if our

robot learns using Q -learning and starts in state S_3 (e.g. $(S_3, Left), (S_2, Right), (S_3, Right), \dots$)?

4. Now consider executing R_{max} on this MDP. Assume that we trust an observed $P(x'|x, a)$ transition probability after a single observation, that the value of $R_{max} = 100$, and that we update our policy each time we observe a transition. Also, assume that R_{max} breaks ties by choosing a policy of $Left$. What are the first 10 $(state, action)$ pairs if our robot learns using R_{max} and starts in state S_3 (e.g. $(S_3, Left), (S_2, Right), (S_3, Right), \dots$)?

3 [Points] Bayes Net Structure Learning

Finding the most likely Bayes Net structure given data is generally intractable. However, if certain restrictions are imposed on the structure, the most likely one can be found efficiently. One such restriction imposes a fixed ordering on the variables of the Bayes Net. This ordering restricts all edges to be directed forward in the ordering. For example, an edge $X \rightarrow Y$ can only exist if X comes before Y in the ordering.

- We'll now explore the effect that the ordering has on the number of parameters and independence assumptions of Bayes Nets. In each box you are given a Bayes Net that obeys a fixed ordering ABCD (1A and 1B).

Draw a Bayes Net (part 2A) for the fixed ordering DCBA that can model the same distribution as the Bayes Net of part 1A. It should have no additional independence assumptions that are not present in part 1A, but also no unnecessary edges. Repeat for 1B and 2B.

Count the number of parameters in each Bayes Net. Each variable is **binary** - it can take on 2 values.

Identify an independence assumption of Bayes Net 1A that doesn't exist in Bayes Net 2A, if such an independence assumption exists. Repeat for Bayes Nets 1B and 2B.

Hint: Pay close attention to V-structures - both existing ones and ones you create!!!

<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>1A</p> </div> <div style="text-align: center;"> <p>2A</p> </div> </div>	<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>1B</p> </div> <div style="text-align: center;"> <p>2B</p> </div> </div>
<p>Number of parameters for Bayes Net 1A</p> <p style="text-align: center;"><u>7</u></p>	<p>Number of parameters for Bayes Net 1B</p> <p style="text-align: center;"><u>11</u></p>
<p>Number of parameters for Bayes Net 2A</p> <p style="text-align: center;"><u>9</u></p>	<p>Number of parameters for Bayes Net 2B</p> <p style="text-align: center;"><u>15</u></p>
<p>List an independence assumption of 1A not present in 2A (if there is one)</p> <p style="text-align: center;"><u>C ⊥ D B</u></p>	<p>List an independence assumption of 1B not present in 2B (if there is one)</p> <p style="text-align: center;"><u>A ⊥ B</u></p>

2. Given a fixed ordering over variables: $X_1, X_2, X_3, \dots, X_n$, show that the choice of parents π_n is independent of the choice of other parents π_1, \dots, π_{n-1} . In other words, show that:

$$\max_{\pi_1, \dots, \pi_n} \log P(X_1, \dots, X_n | \pi_1, \dots, \pi_n) = \max_{\pi_n} f(X_1, \dots, X_n, \pi_n) + \max_{\pi_1, \dots, \pi_{n-1}} g(X_1, \dots, X_n, \pi_1, \dots, \pi_{n-1})$$

for some functions f and g .

$$\log P(X_1, \dots, X_n | \pi_1, \dots, \pi_n) = \log \pi_i \cdot P(X_i | X_1, \dots, X_{i-1}, \pi_1, \dots, \pi_n)$$

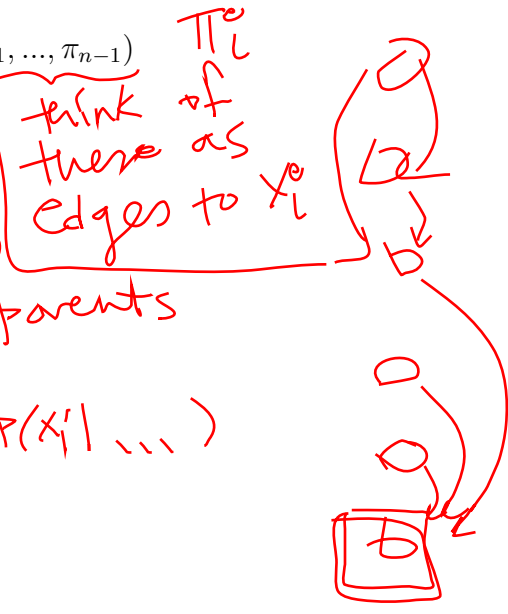
↑ since X_{i+1}, \dots, X_n can't be parents of X_i

$$= \log P(X_n | X_1, \dots, X_{n-1}, \pi_1, \dots, \pi_n) + \sum_{i=1}^{n-1} P(X_i | \dots)$$

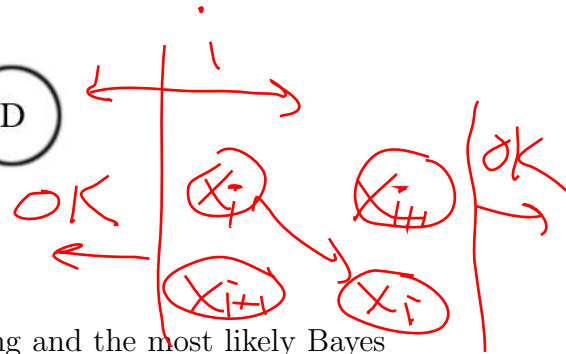
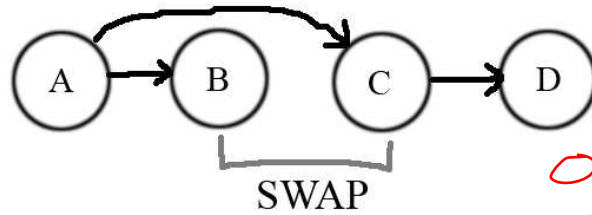
only need the subset that is the parent set of X_i , only π_n has that info

$$= \underbrace{\log P(X_n | X_1, \dots, X_{n-1}, \pi_n)}_{f(X_1, \dots, X_n, \pi_n)} + \underbrace{\sum_{i=1}^{n-1} P(X_i | X_1, \dots, X_{i-1}, \pi_i)}_{g(X_1, \dots, X_n, \pi_1, \dots, \pi_{n-1})}$$

(maybe handwavy)



3. For fixed orderings with a limit of k on the number of parents for each node, the best structure can be obtained by combinatoric search. For each variable, all subsets of variables from earlier in the ordering of size k or less are considered. For each set, $\log P(\text{child}|\text{parents})$ is computed. We saw in part 1 of this question that the ordering can change the number of parameters required to model the joint probability. In this question we'll consider the efficiency of modifying the ordering. This approach can be used to greedily search for a good ordering of variables.



Consider the scenario where you are given a fixed ordering and the most likely Bayes Net structure for that fixed ordering. We would like to find the most likely structure after we switch two **adjacent** variables in the ordering. How many calculations of $\log P(\text{child}|\text{parent})$ would this require in the worst case? Explain.

$X_i = \dots + \binom{i-1}{k-1} + \binom{i-1}{k} = \sum_{l=0}^{k-1} \binom{i-1}{l}$ if you saved values from before

$X_{i+1} =$ only recompute if $X_i \rightarrow X_{i+1}$ originally

Local swapping of variables is prone to getting stuck in local maxima. Instead, let's consider changing the fixed ordering so that the two variables we swap have j **variables in between**. How many $\log P(\text{child}|\text{parent})$ calculations are required to find the most likely structure for this new ordering in the worst case? Explain.

X_i X_{i+1} ... X_{i+j} X_{i+j+1}

Swap

$\sum_{l=1}^{i+j+1} \left[\sum_{d=1}^{k-1} \binom{l-1}{d} \right]$

may need to recompute parent set for all of them (but only the sets containing X_{i+j+1}) by caching

Perhaps you can do better but I don't think this decomposes in a way

4 [Points] Decision Trees

In class, we discussed greedy algorithms for learning decision trees from training data. These algorithms partition the feature space into labeled regions by greedily optimizing some metric (information gain) in hope of producing simple trees that partition the feature space into regions that perfectly classify the training data. As with most greedy approaches, if we consider finding a good tree with a limited depth, this approach is not guaranteed to produce the set of regions that best maximize this metric.

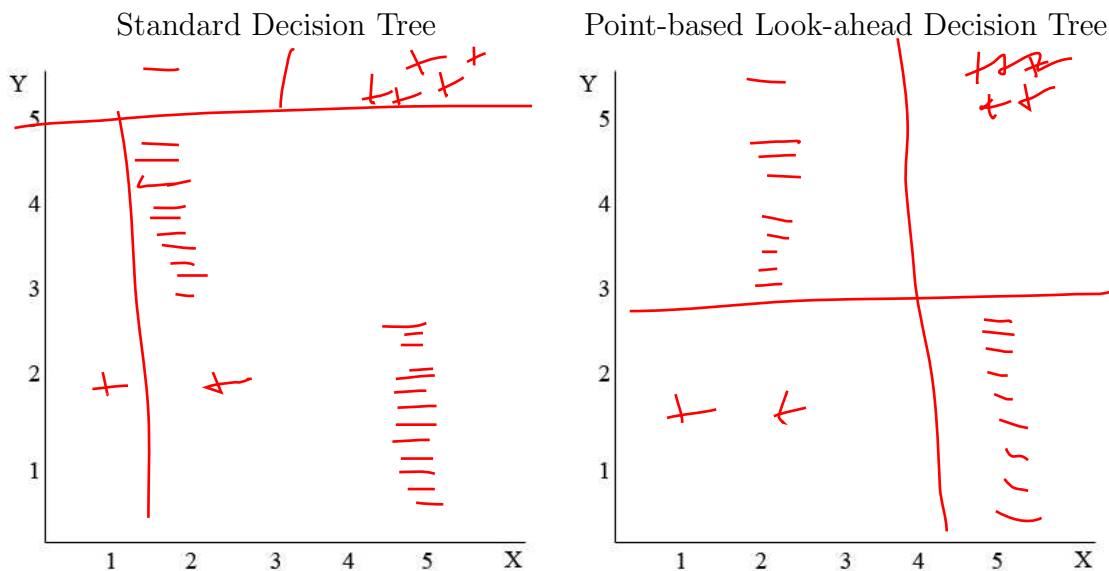
We can always be less greedy. Instead of greedily making one decision and then greedily making the next decision, we can consider the outcome of all possible pairs of those two decisions and choose the best of those. We'll now explore the benefits and costs of being less greedy.

In a **standard decision tree**, each level of the recursion will find one decision boundary (e.g., $X=3$) that partitions the feature space into two regions (e.g., $X > 3$, $X \leq 3$) so to maximize the metric. Each region is then partitioned recursively using the same procedure.

In a **point-based look-ahead decision tree**, the feature space is partitioned into four regions by a single point (e.g., $X, Y = (3, 4)$ gives regions $[X > 3, Y > 4]$, $[X > 3, Y \leq 4]$, $[X \leq 3, Y > 4]$, and $[X \leq 3, Y \leq 4]$).

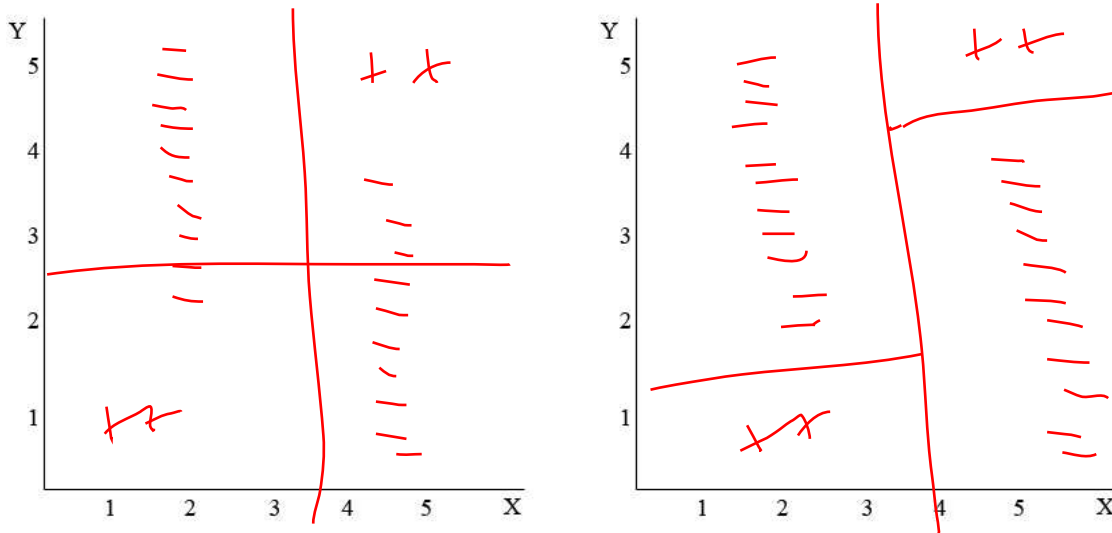
In a **boundary-based look-ahead decision tree**, three decision boundaries are considered in each level of the recursive decision-tree construction. The first decision boundary splits the feature space into two regions, and the two additional decision boundaries split those two regions for a total of 4 regions (e.g., $X = 3, Y = 4$ for $X < 3, Y = 2$ for $X > 3$) which yields regions $[X > 3, Y > 4]$, $[X > 3, Y \leq 4]$, $[X \leq 3, Y > 2]$, and $[X \leq 3, Y \leq 2]$.

1. Draw a dataset on the following 2 plots so that a standard decision tree with two levels (4 regions) will poorly classify the data, but a point-based look-ahead decision tree with one level (4 regions) will perfectly classify the data. Use '+' and '-' to indicate the class of each point and draw in the decision region boundaries of each decision tree.



2. Now draw a dataset on the following 2 plots so that a point-based look-ahead decision tree with one level (4 regions) will poorly classify the data, but a boundary-based look-ahead decision tree with one level (4 regions) will perfectly classify the data. Use '+' and '-' to indicate the class of each point and draw in the decision region boundaries of each decision tree.

Point-based Look-ahead Decision Tree Boundary-based Look-ahead Decision Tree



3. Now provide the running time required for one level of the partitioning in the various decision tree variants. Assume there are D points in the training set all with unique X and Y values. Explain your reasoning.

Standard Decision Tree $X, Y \rightarrow 2D$ possible splits
 Sort $D \log D$ - save time for computing scores
 $O(D \log D)$

Point-Based Look-ahead Decision Tree D^2 splits
 $O(D^2 + D \log D) = O(D^2)$
 again, sort first

Boundary-Based Look-ahead Decision Tree
 for each 1D split, $\leftarrow 2D$ of them
 consider two splits (one on each side)
 $\leftarrow D \log D$
 $O(D^2 \log D)$

5 Neural Networks

Recall the two types of Neural Network activation functions from Homework 2, the linear activation function and the hard threshold:

- linear $y = w_0 + \sum_i w_i x_i$,
- hard threshold

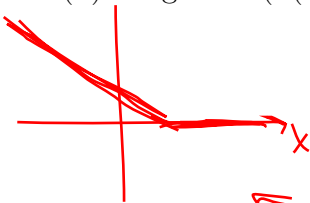
$$y = \begin{cases} 1 & \text{if } w_0 + \sum_i w_i x_i \geq 0, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

1. Which of the following functions can be exactly represented by a neural network with one hidden layer which uses linear and/or hard threshold activation functions? For each case, justify your answer.

(a) polynomials of degree one *Yes.* $y = ax + b$



(b) hinge loss ($h(x) = \max(1-x, 0)$)



No. can't combine linear + threshold

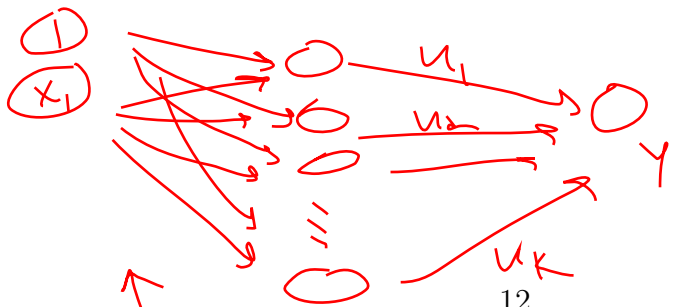


(c) polynomials of degree two

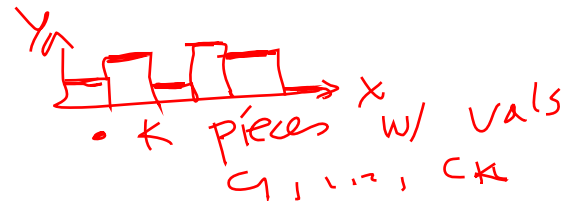
No. $y = ax^2 + bx + c$
can't get this as a linear comb of x_i 's

(d) piecewise constant functions

Yes (in 1-D)



threshold pieces (intervals)



• k pieces w/ vals c_1, \dots, c_k
 • each piece j
 $w_{j0} + w_j x \geq 0$
 $x \geq -\frac{w_{j0}}{w_j}$
 • \exists a set of w_j 's to match c_j 's

6 [points] VC Dimentia

Given a hypothesis class \mathcal{H} , the VC dimension, $VC(\mathcal{H})$ is defined to be the size of the largest set that is shattered by \mathcal{H} . If \mathcal{H} can shatter arbitrarily large sets, then we say that $VC(\mathcal{H}) = \infty$.

1. It is sometimes useful to think of VC dimension as being related to the number of parameters needed to specify an element of \mathcal{H} . For example, what is the VC dimension of the set of hypotheses of the following form?

$$h_{\alpha}(x) = \begin{cases} 1 & \text{if } \alpha_d x^d + \alpha_{d-1} x^{d-1} + \dots + \alpha_0 > 0 \\ 0 & \text{otherwise} \end{cases}$$

Justify your answer.

Hint: think polynomial basis functions

$$X^T \alpha = y \rightarrow \alpha = (X^T)^{-1} y$$

2. Despite the result from part (1), the VC dimension is not always so nicely related to the number of parameters. For any positive integer M , can you come up with a hypothesis class which takes M parameters but has VC dimension 1?

Hint: Think of how you might encode several parameters with just one parameter.

parameters: $\alpha_1, \dots, \alpha_m$

$$h(x) = I(\alpha_1 > 0)$$

or

$$h(x) = I\left(\sum_{i=1}^m \alpha_i > 0\right)$$

dims are linearly indep

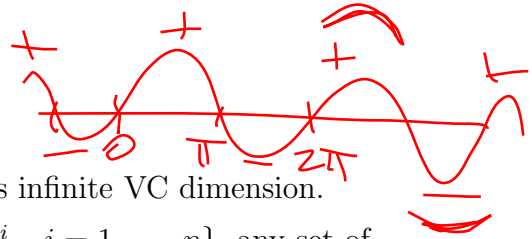
$n = d + 1$ can do it.

if $n > d + 1$, overdetermined system (#var < #constraints) get α working for $d + 1$ pts. label of $d + 2^{\text{th}}$ pt is deterministic

adversary chooses the other label for $d + 2^{\text{th}}$ pt.

3. Consider the class of hypotheses of the form:

$$h_\alpha(x) = \begin{cases} 1 & \text{if } \sin(\alpha x) > 0 \\ 0 & \text{otherwise} \end{cases}$$



You will show that this one-parameter hypothesis class has infinite VC dimension.

To do this, show that given the datapoints $X = \{x_i = 10^{-i}, i = 1, \dots, n\}$, any set of labels $y_i \in \{0, 1\}$ can be realized by h_α by setting

$$\frac{1}{10}, \frac{1}{100}, \frac{1}{1000}, \dots$$

$$\alpha = \left(1 + \sum_{i=1}^n (1 - y_i) 10^i \right) \cdot \pi$$

$y_i \in \{0, 1\}$

For example, if $n = 5$ and $y_i = (1, 1, 1, 1, 0)$, then $\alpha = (100001)\pi$.

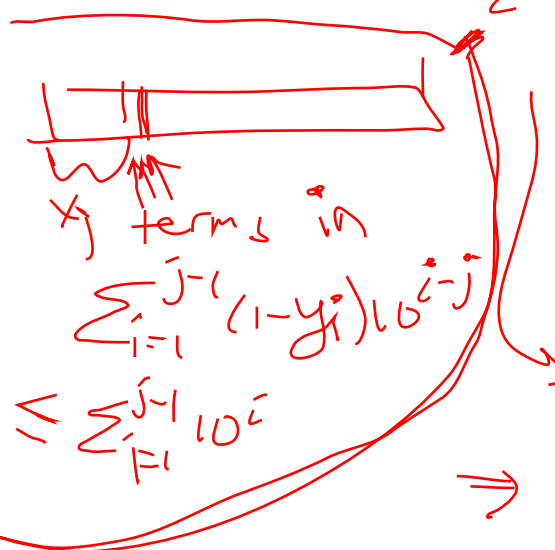
Hint: On intervals of the form $(m\pi, (m+1)\pi)$, the sine function takes positive values if m is even and negative values if m is odd.

$\sin(\alpha x) > 0 \quad y_i = 1$
 for $m\pi < \alpha x_j < (m+1)\pi$, m is even if $y_j = 1$
 m is odd if $y_j = 0$

$$m < \frac{\alpha x_j}{\pi} = \frac{\left(1 + \sum_{i=1}^n (1 - y_i) 10^i \right) \pi x_j}{\pi} = 10^{-j}$$

$$= x_j + \sum_{i=1}^{j-1} (1 - y_i) 10^i 10^{-j} + (1 - y_j) 10^j 10^{-j}$$

$$= x_j + \sum_{i=1}^{j-1} (1 - y_i) 10^{i-j} + (1 - y_j)$$



$0 < \epsilon < 1$

$\sum_{i=1}^{j-1} (1 - y_i) 10^{i-j}$

$\sum_{i=1}^{j-1} 10^i$

\Rightarrow

m is odd if $y = 0$
 m is even if $y = 1$

powers of 10 \Rightarrow even

$\left\{ \begin{array}{ll} \text{even} + \epsilon + 1 & \text{if } y = 0 \\ \text{even} + \epsilon + 0 & \text{if } y = 1 \end{array} \right.$