CS 188 Spring 2015 Introduction to Artificial Intelligence Practice Final

To earn the extra credit, one of the following has to hold true. Please circle and sign.

A I spent 3 or more hours on the practice final.

B I spent fewer than 3 hours on the practice final, but I believe I have solved all the questions.

Signature: Kevin Chau

Follow the directions on the website to submit the practice final and receive the extra credit. The normal instructions for the final follow on the next page.

Exam Instructions:

- You have approximately 2 hours and 50 minutes.
- The exam is closed book, closed calculators, and closed notes except up to three pages of crib sheets.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a brief explanation. All short answer sections can be successfully answered in a few sentences AT MOST.

First name	
Last name	
SID	
edX username	

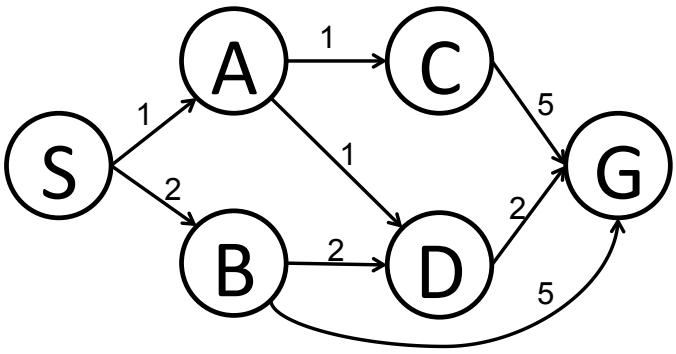
First and last name of student to your left	
First and last name of student to your right	

For staff use only:

Q1.	Search: Short Questions	/4
Q2.	MDPs: Short Questions	/8
Q3.	Other Short Questions	/13
Q4.	Bayes' Nets: Conditional Independence	/8
Q5.	Hidden Markov Models	/13
Q6.	Variable Elimination Ordering	/18
Q7.	Bidirectional A* Search	/14
Q8.	Dual Perceptron CSPs	/11
Q9.	Classification and Separating Hyperplanes	/11
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [4 pts] Search: Short Questions



Answer the following questions about the search problem shown above. S is the start-state, G is the (only) goal-state. Break any ties alphabetically. For the questions that ask for a path, please give your answers in the form '$S - A - D - G$.'

(a) [1 pt] What path would breadth-first graph search return for this search problem?

S-B-G

(b) [1 pt] What path would uniform cost graph search return for this search problem?

S-A-D-G

(c) [1 pt] What path would depth-first graph search return for this search problem?

S-A-C-G

(d) [1 pt] What path would A* graph search, using a consistent heuristic, return for this search problem?

S-A-D-G

Q2. [8 pts] MDPs: Short Questions

(a) Each True/False question is worth 2 points. Leaving a question blank is worth 0 points. **Answering incorrectly is worth −2 points.**

For the questions that are not True/False, justify your answer concisely.

(i) [2 pts] [*true* or *false*] If the only difference between two MDPs is the value of the discount factor then they must have the same optimal policy.

False. Counter example: consider an MDP with two sink states. Transistion into sink state A gives s reward of 1, transitioning into sink state B gives a reward of 10. All other transitions have zero rewards. Let A be one step North from the start state. Let B be two steps South from the start state. Assume actions always succeed. Then if the discount factor gamma < 0.1 the optimal policy takes the agent one step North from the start state into A. If the discount factor gamma > 0.1 the optimal policy akes the agent two steps South from the start state into B.

(ii) [2 pts] [*true* or *false*] When using features to represent the Q-function it is guaranteed that this feature-based Q-learning finds the same Q-function, Q^*, as would be found when using a tabular representation for the Q-function.

False. whenever the optimal Q-function, Q*, canot be represented as a weighted combination of features, then the feature-based representation would not even have the expressiveness to find the optimal Q-function, Q*.

(iii) [2 pts] [*true* or *false*] For an infinite horizon MDP with a finite number of states and actions and with a discount factor γ, with $0 < \gamma < 1$, value iteration is guaranteed to converge.

True

(iv) [2 pts] [*true* or *false*] When getting to act only for a finite number of steps in an MDP, the optimal policy is stationary. (A stationary policy is a policy that takes the same action in a given state, independent of at what time the agent is in that state.)

False

Q3. [13 pts] Other Short Questions

(a) CSPs

(i) [4 pts] **CSP Formulation.**

PacStudent (S), PacBaby (B), PacMom (M), PacDad (D), GrandPac (P), and a friendly Ghost (G) are lining up next to each other. The positions are numbered 1, 2, 3, 4, 5, 6, where 1 neighbors 2, 2 neighbors 1 and 3, 3 neighbors 2 and 4, 4 neighbors 3 and 5, 5 neighbors 4 and 6, and 6 neighbors 5. Each one of them takes up exactly one spot. PacBaby (B) needs to be next to PacMom (M) one one side and PacDad (D) on the other side. GrandPac (P) needs to be next to the Ghost (G). PacStudent (S) needs to be at 1 or 2. Formulate this problem as a CSP: list the variables, their domains, and the constraints. Encode unary constraints as a constraint rather than pruning the domain. (No need to solve the problem, just provide variables, domains and implicit constraints.)

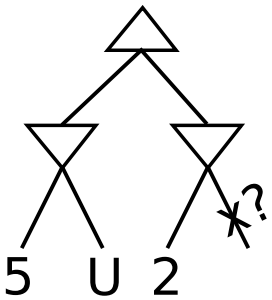
- Variables: S, B, M, D, P, G
- Domains:
  {1,2,3,4,5,6}, {1,2,3,4,5,6}, {1,2,3,4,5,6}, {1,2,3,4,5,6}, {1,2,3,4,5,6}, {1,2,3,4,5,6}
- Constraints:
  S =/= B =/= M =/= D =/= P =/= G
  abs(B-M) = 1
  abs(B-D) = 1
  abs(P-G) = 1
  S = 1 or S = 2

(ii) [2 pts] Consider a CSP with variables X, Y with domains $\{1, 2, 3, 4, 5, 6\}$ for X and $\{2, 4, 6\}$ for Y, and constraints $X < Y$ and $X + Y > 8$. List the values that will remain in the domain of X after enforcing arc consistency for the arc $X \to Y$ (recall arc consistency for a specific arc only prunes the domain of the tail variable, in this case X).

X = {3,4,5}

(b) [2 pts] $\alpha - \beta$ **Pruning: Short Questions**

Consider the game tree shown below. For what range of U will the indicated pruning take place?



U >= 2 (if pruning with equality)

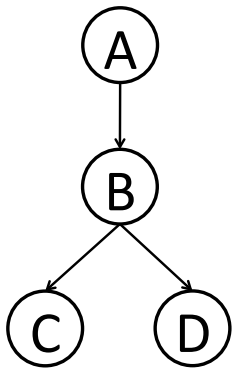
U > 2 if pruning with inequality

Seing 2 has to be worse or at least as bad as seing U

(c) [3 pts] **Bayes' Nets: Representation**

Consider the joint distribution $P(A, B, C, D)$ defined by the Bayes' net below.

A	P(A)
+a	0.8
-a	0.2



A	B	P(B|A)
+a	+b	0.8
+a	-b	0.2
-a	+b	0.5
-a	-b	0.5

B	C	P(C|B)
+b	+c	0.8
+b	-c	0.2
-b	+c	0.5
-b	-c	0.5

B	D	P(D|B)
+b	+d	0.7
+b	-d	0.3
-b	+d	0.1
-b	-d	0.9

Compute the following quantities:

$P(A = +a) =$ .8

$P(A = +a, B = -b, C = -c, D = +d) =$ .8*.2*.5*.1 = .008

$P(A = +a | B = -b, C = -c, D = +d) =$ P(+a,-b,-c,+d)/ [P(+a,-b,-c,+d)+P(-a,-b,-c,+d)] = .008/ (.008+.2*.5*.5*.1) = 0.6153846154

(d) [2 pts] **Naive Bayes**

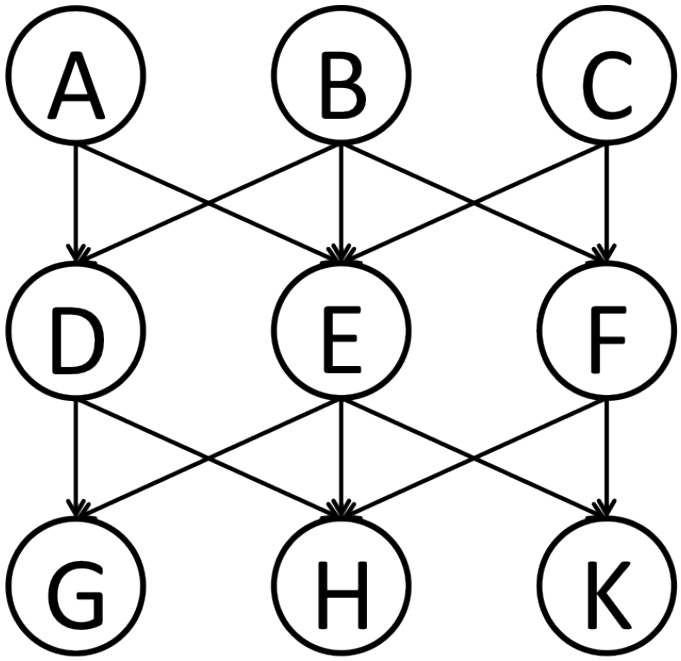
Describe the naive Bayes bag-of-words model for document classification. Draw the Bayes net graph, annotate the class label node and the feature nodes, describe what the domain of the features is, describe any properties of the conditional probability tables that are specific to the bag-of-words (and not necessarily true in all naive Bayes models). For simplicity it is OK to assume that every document in consideration has exactly N words and that words come from a dictionary of D words.

For document classification, the Bayes net has a node for the classification variable Y which points to a weight w_i for each word in the document. In other words, the naive Bayes model has a particular feature choice: there is one feature for each word in the document .The value of the feature is the word at that position. The bag-of-words aspect lies in assuming that the conditional
probability tables are the same for each feature—i.e., it assumes that where a word appears in a document is
irrelevant. The Bayes net graph is shown in the figure above, Y is the class label, W1, . . . , Wn are the features.

Q4. [8 pts] Bayes' Nets: Conditional Independence

(a) [8 pts]

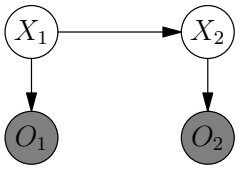
Based only on the structure of the (new) Bayes' Net given below, circle whether the following conditional independence assertions are guaranteed to be true, guaranteed to be false, or cannot be determined by the structure alone.Note: The ordering of the three answer columns might have been switched relative to previous exams!



1	$A \perp\!\!\!\perp C$	Guaranteed false	Cannot be determined	Guaranteed true
2	$A \perp\!\!\!\perp C \mid E$	Guaranteed false	Cannot be determined	Guaranteed true
3	$A \perp\!\!\!\perp C \mid G$	Guaranteed false	Cannot be determined	Guaranteed true
4	$A \perp\!\!\!\perp K$	Guaranteed false	Cannot be determined	Guaranteed true
5	$A \perp\!\!\!\perp G \mid D, E, F$	Guaranteed false	Cannot be determined	Guaranteed true
6	$A \perp\!\!\!\perp B \mid D, E, F$	Guaranteed false	Cannot be determined	Guaranteed true
7	$A \perp\!\!\!\perp C \mid D, F, K$	Guaranteed false	Cannot be determined	Guaranteed true
8	$A \perp\!\!\!\perp G \mid D$	Guaranteed false	Cannot be determined	Guaranteed true

Q5. [13 pts] Hidden Markov Models

Consider the following Hidden Markov Model.



X_1	$\Pr(X_1)$
0	0.3
1	0.7

X_t	X_{t+1}	$\Pr(X_{t+1}\|X_t)$
0	0	0.4
0	1	0.6
1	0	0.8
1	1	0.2

X_t	O_t	$\Pr(O_t\|X_t)$
0	A	0.9
0	B	0.1
1	A	0.5
1	B	0.5

Suppose that $O_1 = A$ and $O_2 = B$ is observed.

(a) [2 pts] Use the Forward algorithm to compute the probability distribution $\Pr(X_2, O_1 = A, O_2 = B)$. Show your work. You do not need to evaluate arithmetic expressions involving only numbers.

P(X2 = 0, O1 = A, O2 = B) = .1[.4*.3*.9+.5*.7*.8]

P(X2 = 0, O1 = A, O2 = B) = .5[.6*.3*.9+.5*.7*.2]

(b) [2 pts] Compute the probability $\Pr(X_1 = 1|O_1 = A, O_2 = B)$. Show your work.

Use Bayes Rule. Compute the probabilities for X1 = 1 and X1 = 0. Then take the conditional probability.

P(x1=0,O1=A,O2=B) = .3*.9*.4*.1+.3*.9*.6*.5 = .0918
P(x1=1,O1=A,O2=B) = .7*.5*.8*.1+.7*.5*.2*.5 =.063
P(x1=1 | O1 = A, O2 = B) = .063/ (.063+.0918) = .407

For the next two questions, use the specified sequence of random numbers $\{a_i\}$ generated independently and uniformly at random from $[0, 1)$ to perform sampling. Specifically, to obtain a sample from a distribution over a variable $Y \in \{0, 1\}$ using the random number a_i, pick $Y = 0$ if $a_i < \Pr(Y = 0)$, and pick $Y = 1$ if $a_i \geq \Pr(Y = 0)$. Similarly, to obtain a sample from a distribution over a variable $Z \in \{A, B\}$ using the random number a_i, pick $Z = A$ if $a_i < \Pr(Z = A)$, and pick $Z = B$ if $a_i \geq \Pr(Z = A)$. Use the random numbers $\{a_i\}$ in order starting from a_1, using a new random number each time a sample needs to be obtained.

(c) [3 pts] Use likelihood-weighted sampling to obtain 2 samples from the distribution $\Pr(X_1, X_2 | O_1 = A, O_2 = B)$, and then use these samples to estimate $\mathrm{E}[\sqrt{X_1 + 3X_2} | O_1 = A, O_2 = B]$.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}
0.134	0.847	0.764	0.255	0.495	0.449	0.652	0.789	0.094	0.028

Sample 1: Sampling X1 = 0 since a1 = 0.134 < 0.3. Sampling X2 = 1 since a2 = 0.847 ≥ 0.4. Weight is 0.9 · 0.5 = 0.45.
Sample 2: Sampling X1 = 1 since a3 = 0.764 ≥ 0.3. Sampling X2 = 0 since a4 = 0.255 < 0.8. Weight is 0.5 · 0.1 = 0.05.
E[√
X1 + 3X2] = 0.45/0.5 ·
√
3 + 0.05/0.5 · 1.0.

(d) [2 pts] [*true* *false*] In the case that there is no evidence, particle filtering using a single particle is equivalent to rejection sampling. Explain your answer.

(e) [2 pts] [*true* *false*] Performing particle filtering twice, each time with 50 particles, is equivalent to performing particle filtering once with 100 particles. Explain your answer.

(f) [2 pts] [*true* *false*] Variable elimination is generally more accurate than the Forward algorithm. Explain your answer.

Q6. [18 pts] Variable Elimination Ordering

Assume all random variables are binary valued.

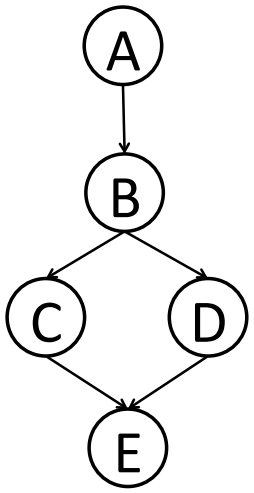
(a) [4 pts] **The Ordering Matters.** Consider the sequence of graphs below. For each, regardless of the elimination ordering, the largest factor produced in finding $p(X)$ will have a table with 2^2 entries.

Now draw a sequence of graphs such that, if you used the best elimination ordering for each graph, the largest factor table produced in variable elimination would have a constant number of entries, but if you used the worst elimination ordering for each graph, the number of entries in the largest factor table would grow exponentially as you move down the sequence. Provide (i) the sequence of graphs, (ii) the sequence of queries for which variable elimination is done, (iii) the best ordering, (iv) the worst ordering.

(b) **Search Space Formulation for Finding an Ordering.** Having established that ordering matters, let's investigate search methods that can find a good elimination ordering. The idea is to step through the process of variable elimination for various orderings of elimination of the hidden variables—and while doing so, only keep track of (i) which factors are present in each step, and (ii) for each factor which variables participate; but not actually compute and store the tables corresponding to each factor. (It is the join and the summation that are the expensive steps in variable elimination—computing which variables would participate in the new factor formed after the join and summation is relatively cheap.) We will use the following search-space formulation. We assume the hidden variables are called $H_1, H_2, \ldots, H_n$, and that all variables are binary.

- set of states S: a state s consists of the current set of factors, including the variables participating in each factor but not the corresponding tables, and any subset of $\{H_1, H_2, \ldots, H_n\}$ to track which variables yet have to be eliminated.
- successor function: choose any of the not yet eliminated variables, and update factors and list of not yet eliminated variables to account for the new elimination.
- cost function: the number of entries in the table representation of the new factor that is generated from the elimination of the current variable
- goal test: test whether the set of not yet eliminated hidden variables is empty.
- start state: set of conditional probability tables and set of all hidden variables.

(i) [4 pts] **Complete Search Tree.** Consider the query $P(D| + e, +c)$. Draw the complete search tree for this problem. Annotate nodes with states, and annotate costs and actions on the edges. Hint: the start state is $(\{P(A), P(B|A), P(+c|B), P(D|B), P(+e| + c, D)\}, \{A, B\})$.



(ii) [2 pts] **Solving the Search for the Example Problem.**

(a) Clearly mark all optimal plans in the search tree above.

(b) What is the cost of an optimal plan to a goal state?

(iii) [8 pts] **Questions about this Search Formulation in General.**

For each of the following heuristics state whether they are admissible or not. Justify your answer. (No credit if there is no justification.) Notation: $\mathcal{H}$ is the set of hidden variables not yet eliminated. $\mathcal{Q}$ is the set of query variables. $\#\mathcal{H}$ is the number of hidden variables not yet eliminated. $\#\mathcal{Q}$ is the number of query variables. Again we assume that all variables are binary-valued.

(a) h_1: $\max_{H_i \in \mathcal{H}}\{$ size of factor generated when eliminating H_i next $\}$

Admissible Not Admissible

(b) h_2: $\min_{H_i \in \mathcal{H}}\{$ size of factor generated when eliminating H_i next $\}$

Admissible Not Admissible

(c) h_3: $2^{\#\mathcal{H}-1}$

Admissible Not Admissible

(d) h_4: if the current largest factor is of size 2^k and $k > \#Q$, then $2^{k-1} + 2^{k-2} + \ldots 2^{\#Q}$; otherwise, 0.

Admissible Not Admissible

Q7. [14 pts] Bidirectional A* Search

If a search problem has only a single goal state, it is common to perform bidirectional search. In bidirectional search you build two search trees at the same time: the "forward" search tree is the one we have always worked with in CS188, the "backward" search tree is one that starts from the goal state, and calls a predecessor (rather than successor) function to work its way back to the start state. Both searches use the same cost function for transitioning between two states. There will now also be a backward heuristic, which for each state estimates the distance to the start state. Bidirectional search can result in significant computational advantages: the size of the search tree built grows exponentially with the depth of the search tree. If growing a tree from start and goal to each other, these two trees could meet in the middle, and one ends up with a computational complexity of just twice searching a tree of half the depth, which are very significant savings.

Recall the pseudo-code for a standard A* graph search

```
function Graph-Search(problem)

  forward-closed   <-- empty set
  forward-priority-queue  <-- Insert(Make-Node(Start-State(problem)), forward-priority-queue)

  LOOP DO
    IF forward-priority-queue is empty THEN return failure

    IF forward-priority-queue is not empty THEN
      node <-- pop(forward-priority-queue)
      IF (State(node) == Goal-State(problem) ) THEN return node

    IF State(node) is not in forward-closed THEN
      add State(node) to forward-closed
      forward-priority-queue <-- Insert-All(ExpandForward(node, problem), forward-priority-queue)

  END // LOOP
```

Now consider the following tentative pseudo-code for bidirectional A* search. We assume a consistent forward heuristic, and a consistent backward heuristic. Concatenation is a function that builds a plan that goes from start state to goal state by combining a forward partial plan and a backward partial plan that end in the same state.

```
function Bidirectional-Graph-Search(problem)

  forward-closed   <-- empty set
  backward-closed  <-- empty set
  forward-priority-queue  <-- Insert(Make-Node(Start-State(problem)), forward-priority-queue)
  backward-priority-queue <-- Insert(Make-Node(Goal-State(problem)), backward-priority-queue)

  LOOP DO
    IF forward-priority-queue is empty AND backward-priority-queue is empty THEN return failure

1   IF there exist a node n1 in forward-priority-queue and a node n2 in backward priority queue ...
1     such that State(n1) == State(n2) THEN
1       return Concatenation of n1 and n2

    IF forward-priority-queue is not empty THEN
      node <-- pop(forward-priority-queue)
      IF ( State(node) == Goal-State(problem) ) THEN return node

2     IF ( State(node) is in backward-priority-queue ) THEN
2       return Concatenation of node and matching node in backward-priority-queue

3     IF ( State(node) is in backward-closed ) THEN
3       return Concatenation of node and matching node in backward-closed

      IF State(node) is not in forward-closed THEN
        add State(node) to forward-closed
        forward-priority-queue <-- Insert-All(ExpandForward(node, problem), forward-priority-queue)

    IF backward-priority-queue is not empty THEN
      node <-- pop(backward-priority-queue)
      IF ( State(node) == Start-State(problem) ) THEN return node

4     IF ( State(node) is in forward-priority-queue ) THEN
4       return Concatenation of node and matching node in forward-priority-queue

5     IF ( State(node) is in forward-closed ) THEN
5       return Concatenation of node and matching node in forward-closed

      IF State(node) is not in backward-closed THEN
        add State(node) to backward-closed
        backward-priority-queue <-- Insert-All(ExpandBackward(node, problem), backward-priority-queue)

  END // LOOP
```

(a) The IF statements labeled 1, 2, 3, 4, 5 are modifications to try to connect both search trees.

(i) [2 pts] If cutting out all lines of code labeled 1, 2, 3, 4, or 5, will Bidirectional-Graph-Search return an optimal solution? Briefly justify your answer.

(ii) [2 pts] If amongst the numbered lines of code we only retain 1, is Bidirectional-Graph-Search guaranteed to be optimal? Briefly justify your answer.

(iii) [2 pts] If amongst the numbered lines of code we only retain 2, is Bidirectional-Graph-Search guaranteed to be optimal? Briefly justify your answer.

(iv) [2 pts] If amongst the numbered lines of code we only retain 3, is Bidirectional-Graph-Search guaranteed to be optimal? Briefly justify your answer.

(v) [2 pts] If amongst the numbered lines of code we only retain 4, is Bidirectional-Graph-Search guaranteed to be optimal? Briefly justify your answer.

(vi) [2 pts] If amongst the numbered lines of code we only retain 5, is Bidirectional-Graph-Search guaranteed to be optimal? Briefly justify your answer.

(vii) [2 pts] Which numbered code section(s) should be retained to maximally benefit from the bidirectional search and at the same time retain optimality guarantees?

Q8. [11 pts] Dual Perceptron CSPs

In this question, we formulate the dual perceptron algorithm as a constraint satisfaction problem (CSP). We have a binary classification problem with classes $+1$ and -1 and a set of n training points, $x_1, x_2, ..., x_n$ with labels $y_i \in \{-1, +1\}$.

Recall that the dual perceptron algorithm takes as input a kernel function $K(x_i, x_j)$ defined on all pairs of training points x_i and x_j, estimates an α_i for each training point x_i, and predicts the class of a point z using $h_\alpha(z) = \sum_{i=1}^{n} \alpha_i K(x_i, z)$, classifying z as positive $(+1)$ if $h_\alpha(a) \geq 0$, and negative (-1) otherwise.

Let the α_i variables of the dual perceptron be the variables in a CSP, with domains restricted to $\{-1, 0, 1\}$. Each training point x_i induces a constraint c_i requiring that it is correctly classified with a margin of at least 1; i.e., $y_i h_\alpha(x_i) \geq 1$.

For this problem, we work with a predefined kernel function $K(x_i, x_j)$. The value of the kernel function (left) for the training points and their labels (right) are given in the tables below. In the kernel table, the j^{th} entry in the i^{th} row is the value of $K(x_i, x_j)$.

	x_1	x_2	x_3	x_4
x_1	1	0	0	-1
x_2	0	4	-2	-2
x_3	0	-2	1	1
x_4	-1	-2	1	2

i	1	2	3	4
y_i	-1	-1	+1	+1

(a) [2 pts] Write each constraint c_i as an inequality in terms of the variables α. (c_1 has been completed for you.)

c_1	$\alpha_1 - \alpha_4 \leq -1$	c_3	
c_2		c_4	

(b) We now randomly initialize to the full assignment $\alpha = (1, -1, 0, -1)$.

(i) [3 pts] For a constraint of the form $a \geq b$, define the *constraint violation margin* (CVM) as the difference $b - a$. For each of the above constraints, circle either *Satisfied* or *Violated* and compute the CVM.

	Satisfied?		CVM		Satisfied?		CVM
c_1	Satisfied	Violated		c_3	Satisfied	Violated	
c_2	Satisfied	Violated		c_4	Satisfied	Violated	

(ii) [4 pts] We decide to run a variation of the min-conflicts algorithm. Recall that min-conflicts begins with a full assignment of all variables and tries to get all constraints satisfied by iteratively modifying the assignment.

In our variant of the algorithm, a single iteration consists of selecting the currently most violated constraint—i.e., the constraint with the highest CVM—and then reassigning all variables that are part of the constraint to values such that the new CVM for the selected constraint is minimized.

Starting from the assignment above ($\alpha = (1, -1, 0, -1)$), run a single iteration of this algorithm. Indicate which constraint c_i is selected, then compute the updated assignment α' and the updated CVM for the selected constraint c_i. Finally, indicate whether or not after this single iteration all constraints have been satisfied (and the algorithm terminates).

Selected c_i	α'_1	α'_2	α'_3	α'_4	Updated CVM	Terminated?
						Yes No

(iii) [2 pts] Suppose we are given a solution to this CSP of $\alpha^* = (-1, -1, +1, +1)$. For each test point z_i whose kernel values with each training point are given in the table below, compute $h_{\alpha^*}(z_i)$ and the predicted classification for z_i.

	$K(x_1, z_i)$	$K(x_2, z_i)$	$K(x_3, z_i)$	$K(x_4, z_i)$	$h_{\alpha^*}(z_i)$	Class prediction?
z_1	3	0	-2	1		
z_2	-2	1	2	-2		

Q9. [11 pts] Classification and Separating Hyperplanes

For this first part, we will be deciding what makes a good feature-mapping for different datasets, as well as finding feature weights that make the data separable.

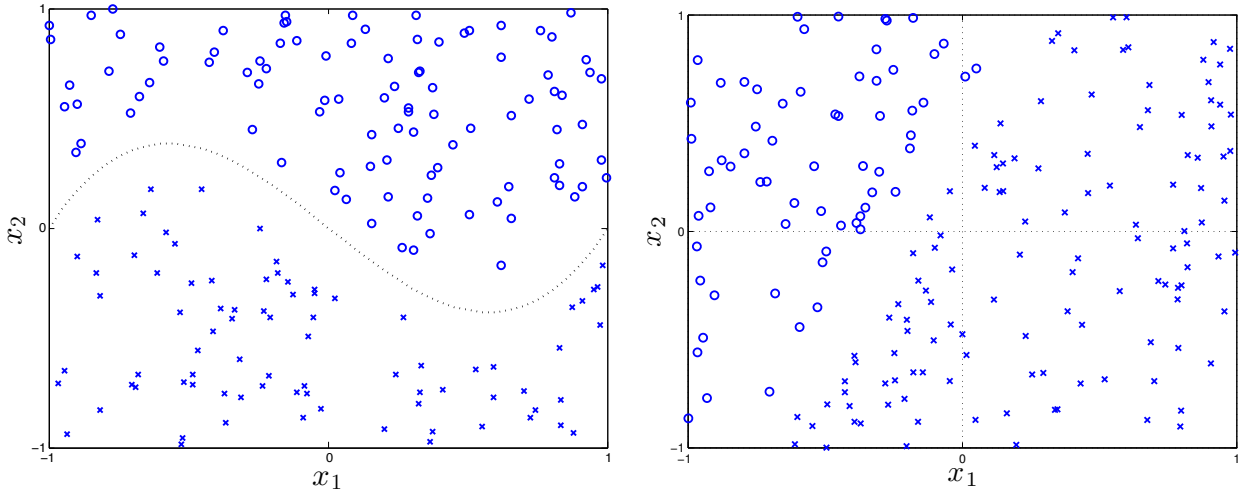


Figure 1: Sets of points separated into positive examples (x's) and negative examples (o's). In plot (a), the dotted line is given by $f(x_1) = x_1^3 - x_1$.

We begin with a series of true/false questions on what kernels can separate the datasets given. We always assume a point x is represented without a bias term, so that $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top$. We will consider the following four kernels:

A. The linear kernel $K_{\text{lin}}(x, z) = x^\top z = x \cdot z$.

B. The shifted linear kernel $K_{\text{bias}}(x, z) = 1 + x^\top z = 1 + x \cdot z$.

C. The quadratic kernel $K_{\text{quad}}(x, z) = (1 + x^\top z)^2 = (1 + x \cdot z)^2$.

D. The cubic kernel $K_{\text{cub}}(x, z) = (1 + x^\top z)^3 = (1 + x \cdot z)^3$.

(a) **(i)** [1 pt] [*true* *false*] The kernel K_{lin} can separate the dataset in Fig. 1(b).

(ii) [1 pt] [*true* *false*] The kernel K_{bias} can separate the dataset in Fig. 1(b).

(iii) [1 pt] [*true* *false*] The kernel K_{cub} can separate the dataset in Fig. 1(b).

(iv) [1 pt] [*true* *false*] The kernel K_{lin} can separate the datset in Fig. 1(a).

(v) [1 pt] [*true* *false*] The kernel K_{quad} can separate the datset in Fig. 1(a).

(vi) [1 pt] [*true* *false*] The kernel K_{cub} can separate the datset in Fig. 1(a).

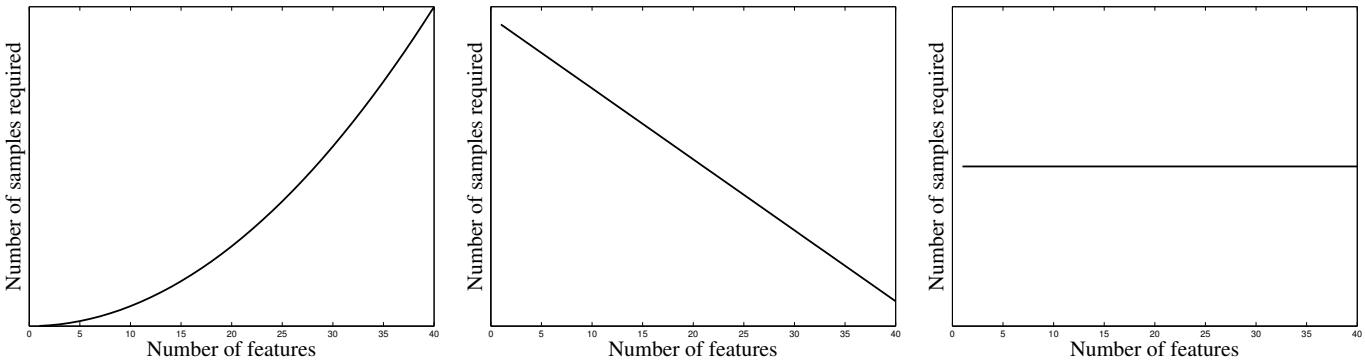
(b) [2 pts] Now imagine that instead of simply using $x \in \mathbb{R}^2$ as input to our learning algorithm, we use a feature mapping $\phi : x \mapsto \phi(x) \in \mathbb{R}^k$, where $k \gg 2$, so that we can learn more powerful classifiers. Specifically, suppose that we use the feature mapping

$$\phi(x) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_2^2 & x_1^3 & x_2^3 \end{bmatrix}^\top \tag{1}$$

so that $\phi(x) \in \mathbb{R}^7$. Give a weight vector w that separates the x points from the o points in Fig. 1(a), that is, $w^\top \phi(x) = w \cdot \phi(x)$ should be > 0 for x points and < 0 for o points.

(c) [1 pt] Using the feature mapping (1), give a weight vector w that separates the x points from the o points in Fig. 1(b), assuming that the line given by $f(x_1) = ax_1 + b$ lies completely between the two sets of points.

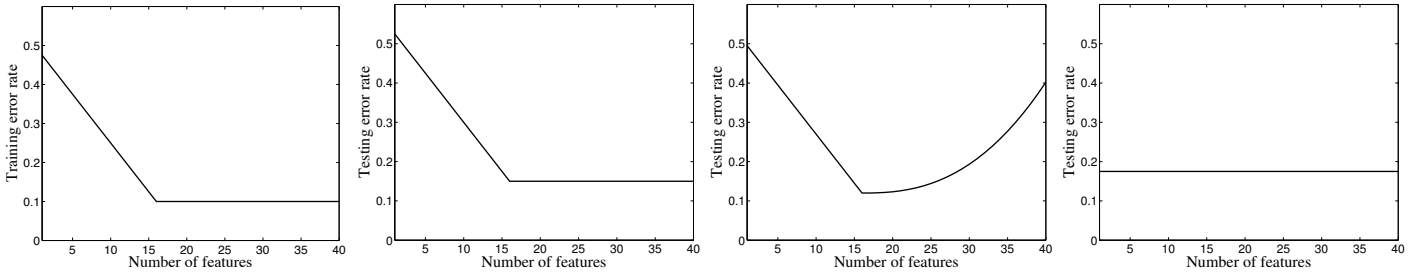
Now it's time to test your understanding of training error, test error, and the number of samples required to learn a classifier. Imagine you are learning a linear classifier of the form $\text{sign}(w^\top \phi(x))$, as in the binary Perceptron or SVM, and you are trying to decide how many features to use in your feature mapping $\phi(x)$.



(a) (b) (c)

Figure 2: Number of samples required to learn a linear classifier $\text{sign}(w^\top \phi(x))$ as a function of the number of features used in the feature mapping $\phi(x)$.

(d) [1 pt] Which of the plots (a), (b), and (c) in Fig. 2 is most likely to reflect the number of samples necessary to learn a classifier with good generalization properties as a function of the number of features used in $\phi(x)$?



(a) (b) (c)

Figure 3: Leftmost plot: training error of your classifier as a function of the number of features.

(e) [1 pt] You notice in training your classifier that the training error rate you achieve, as a function of the number of features, looks like the left-most plot in Fig. 3. Which of the plots (a), (b), or (c) in Fig. 3 is most likely to reflect the error rate of your classifier on a held-out validation set (as a function of the number of features)?