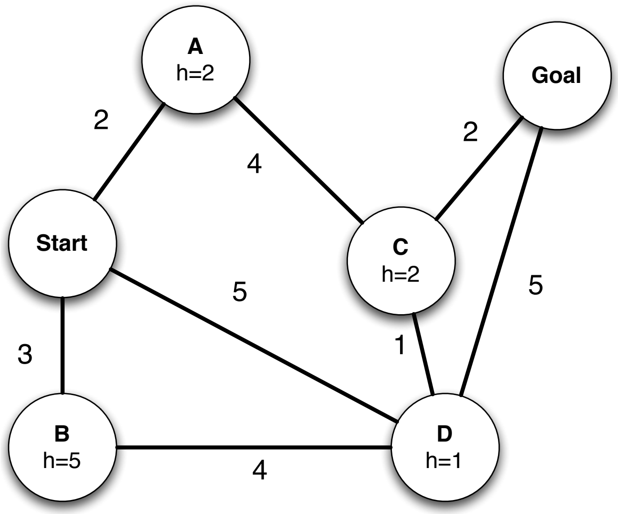
CS188 Spring 2014 Section 0: Search

1 Search algorithms in action



For each of the following graph search strategies, work out the order in which states are expanded, as well as the path returned by graph search. In all cases, assume ties resolve in such a way that states with earlier alphabetical order are expanded first. The start and goal state are S and G, respectively. Remember that in graph search, a state is expanded only once.

a) Depth-first search.

b) Breadth-first search.

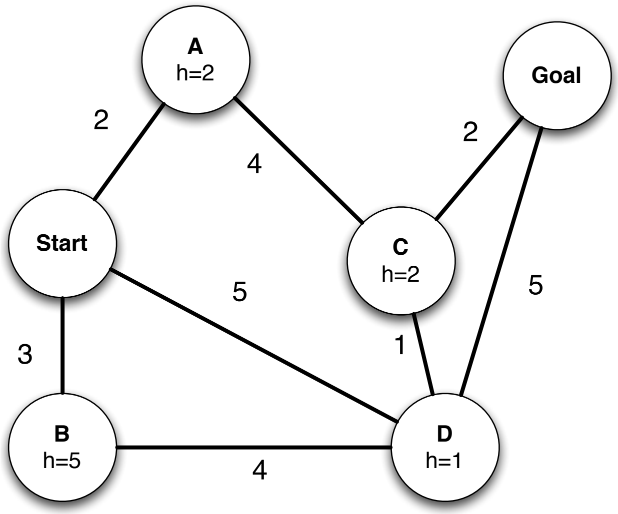
c) Uniform cost search.

d) Greedy search with the heuristic h shown on the graph.

e) $A^\star$ search with the same heuristic.

CS188 Spring 2014 Section 0: Search

1 Search algorithms in action



For each of the following graph search strategies, work out the order in which states are expanded, as well as the path returned by graph search. In all cases, assume ties resolve in such a way that states with earlier alphabetical order are expanded first. The start and goal state are S and G, respectively. Remember that in graph search, a state is expanded only once.

a) Depth-first search.
States Expanded: Start, A, C, D, B, Goal
Path Returned: Start-A-C-D-Goal

b) Breadth-first search.
States Expanded: Start, A, B, D, C, Goal
Path Returned: Start-D-Goal

c) Uniform cost search.
States Expanded: Start, A, B, D, C, Goal
Path Returned: Start-A-C-Goal

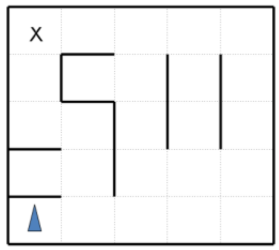
d) Greedy search with the heuristic h shown on the graph.
States Expanded: Start, D, Goal
Path Returned: Start-D-Goal

e) $A^{\star}$ search with the same heuristic.
States Expanded: Start, A, D, C, Goal
Path Returned: Start-A-C-Goal

CS188 Spring 2014 Section 1: Search

1 Search and Heuristics

Imagine a car-like agent wishes to exit a maze like the one shown below:



The agent is directional and at all times faces some direction $d \in (N, S, E, W)$. With a single action, the agent can *either* move forward at an adjustable velocity v *or* turn. The turning actions are *left* and *right*, which change the agent's direction by 90 degrees. Turning is only permitted when the velocity is zero (and leaves it at zero). The moving actions are *fast* and *slow*. *Fast* increments the velocity by 1 and *slow* decrements the velocity by 1; in both cases the agent then moves a number of squares equal to its NEW adjusted velocity. Any action that would result in a collision with a wall crashes the agent and is illegal. Any action that would reduce v below 0 or above a maximum speed $V_{\max}$ is also illegal. The agent's goal is to find a plan which parks it (stationary) on the exit square using as few actions (time steps) as possible.

As an example: if the agent shown were initially stationary, it might first turn to the east using (*right*), then move one square east using *fast*, then two more squares east using *fast* again. The agent will of course have to *slow* to turn.

1. If the grid is M by N, what is the size of the state space? Justify your answer. You should assume that all configurations are reachable from the start state.

2. What is the maximum branching factor of this problem? You may assume that illegal actions are simply not returned by the successor function. Briefly justify your answer.

3. Is the Manhattan distance from the agent's location to the exit's location admissible? Why or why not?

4. State and justify a non-trivial admissible heuristic for this problem which is not the Manhattan distance to the exit.

5. If we used an inadmissible heuristic in A* tree search, could it change the completeness of the search?

6. If we used an inadmissible heuristic in A* tree search, could it change the optimality of the search?

7. Give a general advantage that an inadmissible heuristic might have over an admissible one.

2 Expanded Nodes

Consider tree search (i.e. no closed set) on an arbitrary search problem with max branching factor b. Each search node n has a backward (cumulative) cost of $g(n)$, an admissible heuristic of $h(n)$, and a depth of $d(n)$. Let c be a minimum-cost goal node, and let s be a shallowest goal node.

For each of the following, you will give an expression that characterizes the set of nodes that are expanded before the search terminates. For instance, if we asked for the set of nodes with positive heuristic value, you could say $h(n) \geq 0$. Don't worry about ties (so you won't need to worry about $>$ versus $\geq$). If there are no nodes for which the expression is true, you must write "none."

1. Give an expression (i.e. an inequality in terms of the above quantities) for which nodes n will be expanded in a breadth-first tree search.

2. Give an expression for which nodes n will be expanded in a uniform cost search.

3. Give an expression for which nodes n will be expanded in an A* tree search with heuristic $h(n)$.

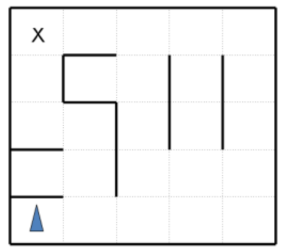
4. Let h_1 and h_2 be two admissible heuristics such that $\forall n, h_1(n) \geq h_2(n)$. Give an expression for the nodes which will be expanded in an A* tree search using h_1 but not when using h_2.

5. Give an expression for the nodes which will be expanded in an A* tree search using h_2 but not when using h_1.

CS188 Spring 2014 Section 1: Search

1 Search and Heuristics

Imagine a car-like agent wishes to exit a maze like the one shown below:



The agent is directional and at all times faces some direction $d \in (N, S, E, W)$. With a single action, the agent can *either* move forward at an adjustable velocity v *or* turn. The turning actions are *left* and *right*, which change the agent's direction by 90 degrees. Turning is only permitted when the velocity is zero (and leaves it at zero). The moving actions are *fast* and *slow*. *Fast* increments the velocity by 1 and *slow* decrements the velocity by 1; in both cases the agent then moves a number of squares equal to its NEW adjusted velocity. Any action that would result in a collision with a wall crashes the agent and is illegal. Any action that would reduce v below 0 or above a maximum speed $V_{\max}$ is also illegal. The agent's goal is to find a plan which parks it (stationary) on the exit square using as few actions (time steps) as possible.

As an example: if the agent shown were initially stationary, it might first turn to the east using (*right*), then move one square east using *fast*, then two more squares east using *fast* again. The agent will of course have to *slow* to turn.

1. If the grid is M by N, what is the size of the state space? Justify your answer. You should assume that all configurations are reachable from the start state.

   The size of the state space is $4MN(V_{max} + 1)$. The state representation is (direction facing, x, y, speed). Note that the speed can take any value in $\{0, ..., V_{max}\}$.

2. What is the maximum branching factor of this problem? You may assume that illegal actions are simply not returned by the successor function. Briefly justify your answer.

   The maximum branching factor is 3, and this happens when the agent is stationary. While stationary it can take the following 3 actions - $fast$, $left$, $right$.

3. Is the Manhattan distance from the agent's location to the exit's location admissible? Why or why not?

   No, Manhattan distance is not an admissible heuristic. The agent can move at an average speed of greater than 1 (by first speeding up to V_{max} and then slowing down to 0 as it reaches the goal), and so can reach the goal in less time steps than there are squares between it and the goal. A specific example: the target is 6 squares away, and the agent's velocity is already 4. By taking only 4 *slow* actions, it reaches the goal with a velocity of 0.

4. State and justify a non-trivial admissible heuristic for this problem which is not the Manhattan distance to the exit.

   There are many answers to this question. Here are a few, in order of weakest to strongest:

   (a) The number of turns required for the agent to face the goal.

   (b) Consider a relaxation of the problem where there are no walls, the agent can turn and change speed arbitrarily. In this relaxed problem, the agent would move with V_{max}, and then suddenly stop at the goal, thus taking $d_{manhattan}/V_{max}$ time.

   (c) We can improve the above relaxation by accounting for the deceleration dynamics. In this case the agent will have to slow down to 0 when it is about to reach the goal. Note that this heuristic will always return a greater value than the previous one, but is still not an overestimate of the true cost to reach the goal. We can say that this heuristic *dominates* the previous one.

5. If we used an inadmissible heuristic in A* tree search, could it change the completeness of the search?

   No! If the heuristic function is bounded, then A* tree search would visit all the nodes eventually, and would find a path to the goal state if there exists one.

6. If we used an inadmissible heuristic in A* tree search, could it change the optimality of the search?

   Yes! It can make the good optimal goal look as though it is very far off, and take you to a suboptimal goal.

7. Give a general advantage that an inadmissible heuristic might have over an admissible one.

   The time to solve an A* tree search problem is a function of two factors: the number of nodes expanded, and the time spent per node.

   An inadmissible heuristic may be faster to compute, leading to a solution that is obtained faster due to less time spent per node. It can also be a closer estimate to the actual cost function (even though at times it will overestimate!), thus expanding less nodes.

   We lose the guarantee of optimality by using an inadmissible heuristic. But sometimes we may be okay with finding a suboptimal solution to a search problem.

2 Expanded Nodes

Consider tree search (i.e. no closed set) on an arbitrary search problem with max branching factor b. Each search node n has a backward (cumulative) cost of $g(n)$, an admissible heuristic of $h(n)$, and a depth of $d(n)$. Let c be a minimum-cost goal node, and let s be a shallowest goal node.

For each of the following, you will give an expression that characterizes the set of nodes that are expanded before the search terminates. For instance, if we asked for the set of nodes with positive heuristic value, you could say $h(n) \geq 0$. Don't worry about ties (so you won't need to worry about $>$ versus $\geq$). If there are no nodes for which the expression is true, you must write "none."

1. Give an expression (i.e. an inequality in terms of the above quantities) for which nodes n will be expanded in a breadth-first tree search.

   $d(n) \leq d(s)$: BFS expands all nodes which are shallower than the shallowest goal. Recall that our search performs the $goal - test$ after popping nodes from the fringe, so we typically expand some nodes at depth s, before we expand the optimal goal node.

2. Give an expression for which nodes n will be expanded in a uniform cost search.

   $g(n) \leq g(c)$: Uniform cost search expands all nodes that are closer than the closest goal node. Recall that our search performs the $goal - test$ after popping nodes from the fringe (this ensures optimality!), so we might expand some nodes of cost $g(c)$, before we expand the optimal goal node.

3. Give an expression for which nodes n will be expanded in an A* tree search with heuristic $h(n)$.

   $g(n) + h(n) \leq g(c)$: All nodes with a total cost of less than g(c), get expanded before the goal node is expanded. This can be proved by induction on the cost $g(n) + h(n)$. Consider a node n_1 which satisfies this property. Note that its parent n_0, will also satisfy this inequality, and by the induction hypothesis, n_0 will be expanded before the goal is expanded, which means that it will put n_1 on the fringe, which will get expanded before the goal node is expanded.

4. Let h_1 and h_2 be two admissible heuristics such that $\forall n, h_1(n) \geq h_2(n)$. Give an expression for the nodes which will be expanded in an A* tree search using h_1 but not when using h_2.

   Let S, be the set of all the nodes. Using the above part, set of nodes expanded by h_1 is $N_1 = \{n : g(n) + h_1(n) \leq g(c)\}$, and, set of nodes expanded by h_2 is $N_2 = \{n : g(n) + h_2(n) \leq g(c)\}$. The set of nodes expanded using h_1 but not using h_2, is $N_1 \cap (S - N_2)$. Since, $h_1 \geq h_2$, $N_1 \subseteq N_2$, hence $N_1 \cap (S - N_2) = \phi$.

5. Give an expression for the nodes which will be expanded in an A* tree search using h_2 but not when using h_1.

   As above, set of nodes expanded using h_2 but not using h_1, is $N_2 \cap (S - N_1) = \{n : g(n) + h_2(n) \leq g(c)$ and $g(c) \leq g(n) + h_1(n)\}$.

CS188 Spring 2014 Section 2: CSPs

1 Course Scheduling

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 5 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time.

The classes are:

1. Class 1 - Intro to Programming: meets from 8:00-9:00am
2. Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
3. Class 3 - Natural Language Processing: meets from 9:00-10:00am
4. Class 4 - Computer Vision: meets from 9:00-10:00am
5. Class 5 - Machine Learning: meets from 10:30-11:30am

The professors are:

1. Professor A, who is qualified to teach Classes 1, 2, and 5.
2. Professor B, who is qualified to teach Classes 3, 4, and 5.
3. Professor C, who is qualified to teach Classes 1, 3, and 4.

1. Formulate this problem as a CSP problem in which there is one variable per class, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit.

2. Draw the constraint graph associated with your CSP.

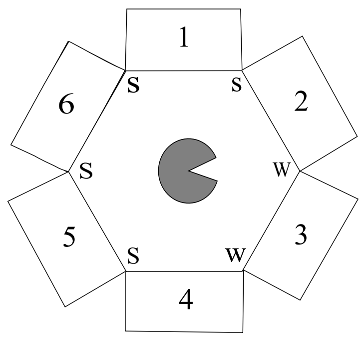
3. Your CSP should look nearly tree-structured. Briefly explain (one sentence or less) why we might prefer to solve tree-structured CSPs.

2 CSPs: Trapped Pacman

Pacman is trapped! He is surrounded by mysterious corridors, each of which leads to either a pit (P), a ghost (G), or an exit (E). In order to escape, he needs to figure out which corridors, if any, lead to an exit and freedom, rather than the certain doom of a pit or a ghost.

The one sign of what lies behind the corridors is the wind: a pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all. Unfortunately, Pacman cannot measure the strength of the breeze at a specific corridor. Instead, he can stand *between* two adjacent corridors and feel the max of the two breezes. For example, if he stands between a pit and an exit he will sense a strong (S) breeze, while if he stands between an exit and a ghost, he will sense a weak (W) breeze. The measurements for all intersections are shown in the figure below.

Also, while the total number of exits might be zero, one, or more, Pacman knows that two neighboring squares will *not* both be exits.



Pacman models this problem using variables X_i for each corridor i and domains P, G, and E.

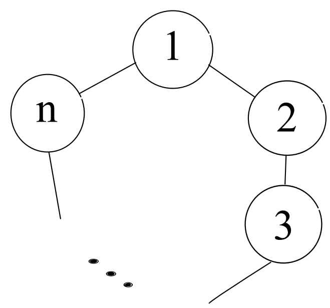
1. State the binary and/or unary constraints for this CSP (either implicitly or explicitly).

2. Cross out the values from the domains of the variables that will be deleted in enforcing arc consistency.

X_1	P	G	E
X_2	P	G	E
X_3	P	G	E
X_4	P	G	E
X_5	P	G	E
X_6	P	G	E

3. According to MRV, which variable or variables could the solver assign first?

4. Assume that Pacman knows that $X_6 = G$. List all the solutions of this CSP or write *none* if no solutions exist.



5. The CSP described above has a circular structure with 6 variables. Now consider a CSP forming a circular structure that has n variables ($n > 2$), as shown below. Also assume that the domain of each variable has cardinality d. Explain precisely how to solve this general class of circle-structured CSPs efficiently (i.e. in time linear in the number of variables), using methods covered in class. Your answer should be at most two sentences.

6. If standard backtracking search were run on a circle-structured graph, enforcing arc consistency at every step, what, if anything, can be said about the worst-case backtracking behavior (e.g. number of times the search could backtrack)?

CS188 Spring 2014 Section 2: CSPs

1 Course Scheduling

You are in charge of scheduling for computer science classes that meet Mondays, Wednesdays and Fridays. There are 5 classes that meet on these days and 3 professors who will be teaching these classes. You are constrained by the fact that each professor can only teach one class at a time.

The classes are:

1. Class 1 - Intro to Programming: meets from 8:00-9:00am
2. Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:30am
3. Class 3 - Natural Language Processing: meets from 9:00-10:00am
4. Class 4 - Computer Vision: meets from 9:00-10:00am
5. Class 5 - Machine Learning: meets from 10:30-11:30am

The professors are:

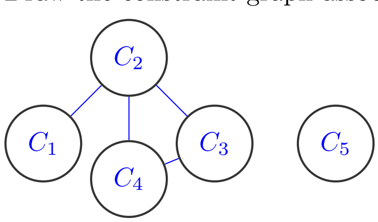
1. Professor A, who is qualified to teach Classes 1, 2, and 5.
2. Professor B, who is qualified to teach Classes 3, 4, and 5.
3. Professor C, who is qualified to teach Classes 1, 3, and 4.

1. Formulate this problem as a CSP problem in which there is one variable per class, stating the domains, and constraints. Constraints should be specified formally and precisely, but may be implicit rather than explicit.

Variables	Domains (or unary constraints)
C_1	{A, C}
C_2	{A}
C_3	{B, C}
C_4	{B, C}
C_5	{A, B}

Binary Constraints
$C_1 \neq C_2$
$C_2 \neq C_3$
$C_2 \neq C_4$
$C_3 \neq C_4$

2. Draw the constraint graph associated with your CSP.



3. Your CSP should look nearly tree-structured. Briefly explain (one sentence or less) why we might prefer to solve tree-structured CSPs.

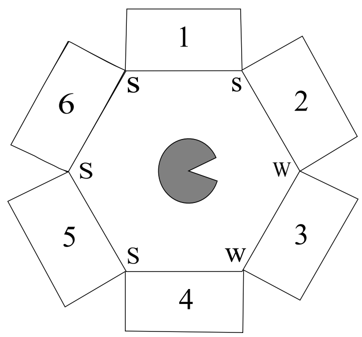
   Minimal answer: we can solve them in polynomial time. If a graph is tree structured (i.e. has no loops), then the CSP can be solved in $O(nd^2)$ time as compared to general CSPs, where worst-case time is $O(d^n)$. For tree-structured CSPs you can choose an ordering such that every node's parent precedes it in the ordering. Then after enforcing arc consistency you can greedily assign the nodes in order, starting from the root, and will find a consistent assignment without backtracking.

2 CSPs: Trapped Pacman

Pacman is trapped! He is surrounded by mysterious corridors, each of which leads to either a pit (P), a ghost (G), or an exit (E). In order to escape, he needs to figure out which corridors, if any, lead to an exit and freedom, rather than the certain doom of a pit or a ghost.

The one sign of what lies behind the corridors is the wind: a pit produces a strong breeze (S) and an exit produces a weak breeze (W), while a ghost doesn't produce any breeze at all. Unfortunately, Pacman cannot measure the strength of the breeze at a specific corridor. Instead, he can stand *between* two adjacent corridors and feel the max of the two breezes. For example, if he stands between a pit and an exit he will sense a strong (S) breeze, while if he stands between an exit and a ghost, he will sense a weak (W) breeze. The measurements for all intersections are shown in the figure below.

Also, while the total number of exits might be zero, one, or more, Pacman knows that two neighboring squares will *not* both be exits.



Pacman models this problem using variables X_i for each corridor i and domains P, G, and E.

1. State the binary and/or unary constraints for this CSP (either implicitly or explicitly).

Binary:

$X_1 = P$ or $X_2 = P$, $X_2 = E$ or $X_3 = E$,
$X_3 = E$ or $X_4 = E$, $X_4 = P$ or $X_5 = P$,
$X_5 = P$ or $X_6 = P$, $X_1 = P$ or $X_6 = P$,
$\forall i, j$ s.t. Adj(i, j) $\neg(X_i = E$ and $X_j = E)$

Unary:

$X_2 \neq P$,
$X_3 \neq P$,
$X_4 \neq P$

2. Cross out the values from the domains of the variables that will be deleted in enforcing arc consistency.

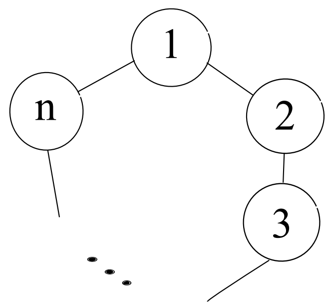
X_1	P		
X_2		G	E
X_3		G	E
X_4		G	E
X_5	P		
X_6	P	G	E

3. According to MRV, which variable or variables could the solver assign first?

   X_1 or X_5 (tie breaking)

4. Assume that Pacman knows that $X_6 = G$. List all the solutions of this CSP or write *none* if no solutions exist.

   (P,E,G,E,P,G)
   (P,G,E,G,P,G)



5. The CSP described above has a circular structure with 6 variables. Now consider a CSP forming a circular structure that has n variables ($n > 2$), as shown below. Also assume that the domain of each variable has cardinality d. Explain precisely how to solve this general class of circle-structured CSPs efficiently (i.e. in time linear in the number of variables), using methods covered in class. Your answer should be at most two sentences.

   We fix X_j for some j and assign it a value from its domain (i.e. use cutset conditioning on one variable). The rest of the CSP now forms a tree structure, which can be efficiently solved without backtracking by enforcing arc consistency. We try all possible values for our selected variable X_j until we find a solution.

6. If standard backtracking search were run on a circle-structured graph, enforcing arc consistency at every step, what, if anything, can be said about the worst-case backtracking behavior (e.g. number of times the search could backtrack)?

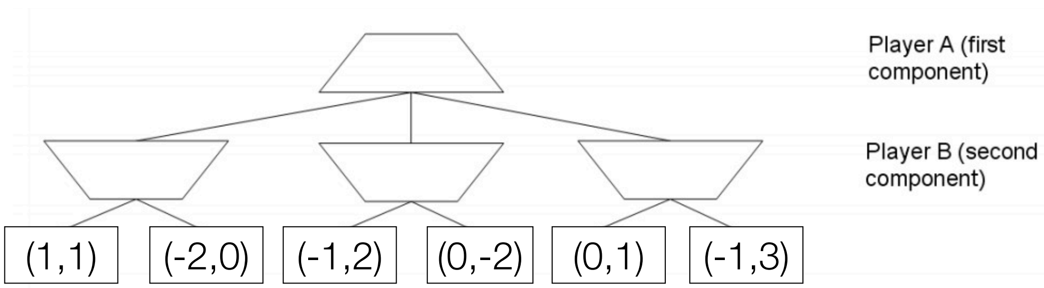
   A tree structured CSP can be solved without any backtracking. Thus, the above circle-structured CSP can be solved after backtracking at most d times, since we might have to try up to d values for X_j before finding a solution.

CS188 Spring 2014 Section 3: Games

1 Nearly Zero Sum Games

The standard Minimax algorithm calculates worst-case values in a *zero-sum* two player game, i.e. a game in which for all terminal states s, the utilities for players A (MAX) and B (MIN) obey $U_A(s) + U_B(s) = 0$. In the zero sum case, we know that $U_A(s) = -U_B(s)$ and so we can think of player B as simply minimizing $U_A(s)$.

In this problem, you will consider the *non zero-sum* generalization in which the sum of the two players' utilities are not necessarily zero. Because player A's utility no longer determines player B's utility exactly, the leaf utilities are written as pairs $(U_A; U_B)$, with the first and second component indicating the utility of that leaf to A and B respectively. In this generalized setting, A seeks to maximize U_A, the first component, while B seeks to maximize U_B, the second component.



1. Propagate the terminal utility pairs up the tree using the appropriate generalization of the minimax algorithm on this game tree. Fill in the values (as pairs) at each of the internal node. Assume that each player maximizes their own utility.

2. Briefly explain why no alpha-beta style pruning is possible in the general non-zero sum case.
   Hint: think first about the case where $U_A(s) = U_B(s)$ for all nodes.

3. For minimax, we know that the value v computed at the root (say for player A = MAX) is a worst-case value. This means that if the opponent MIN doesn't act optimally, the actual outcome v' for MAX can only be better, never worse than v.

   In the general non-zero sum setup, can we say that the value U_A computed at the root for player A is also a worst-case value in this sense, or can A's outcome be worse than the computed U_A if B plays sub-optimally? Briefly justify.

4. Now consider the nearly zero sum case, in which $|U_A(s) + U_B(s)| \leq \epsilon$ at all terminal nodes s for some ϵ which is known in advance. For example, the previous game tree is nearly zero sum for $\epsilon = 2$.

   In the nearly zero sum case, pruning is possible. Draw an X in each node in this game tree which could be pruned with the appropriate generalization of alpha-beta pruning. Assume that the exploration is being done in the standard left to right depth-first order and the value of ϵ is known to be 2. Make sure you make use of ϵ in your reasoning.

5. Give a general condition under which a child n of a B node (MIN node) b can be pruned. Your condition should generalize α-pruning and should be stated in terms of quantities such as the utilities $U_A(s)$ and/or $U_B(s)$ of relevant nodes s in the game tree, the bound ϵ, and so on. Do not worry about ties.

6. In the nearly zero sum case with bound ϵ, what guarantee, if any, can we make for the actual outcome u' for player A (in terms of the value U_A of the root) in the case where player B acts sub-optimally?

2 Minimax and Expectimax

In this problem, you will investigate the relationship between expectimax trees and minimax trees for zero-sum two player games. Imagine you have a game which alternates between player 1 (max) and player 2. The game begins in state s_0, with player 1 to move. Player 1 can either choose a move using minimax search, or expectimax search, where player 2's nodes are chance rather than min nodes.

1. Draw a (small) game tree in which the root node has a larger value if expectimax search is used than if minimax is used, or argue why it is not possible.

2. Draw a (small) game tree in which the root node has a larger value if minimax search is used than if expectimax is used, or argue why it is not possible.

3. Under what assumptions about player 2 should player 1 use minimax search rather than expectimax search to select a move?

4. Under what assumptions about player 2 should player 1 use expectimax search rather than minimax search?

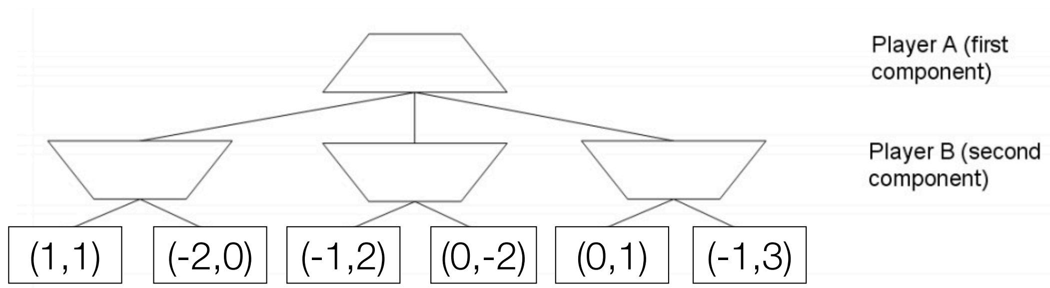
5. Imagine that player 1 wishes to act optimally (rationally), and player 1 knows that player 2 also intends to act optimally. However, player 1 also knows that player 2 (mistakenly) believes that player 1 is moving uniformly at random rather than optimally. Explain how player 1 should use this knowledge to select a move. Your answer should be a precise algorithm involving a game tree search, and should include a sketch of an appropriate game tree with player 1's move at the root. Be clear what type of nodes are at each ply and whose turn each ply represents.

CS188 Spring 2014 Section 3: Games

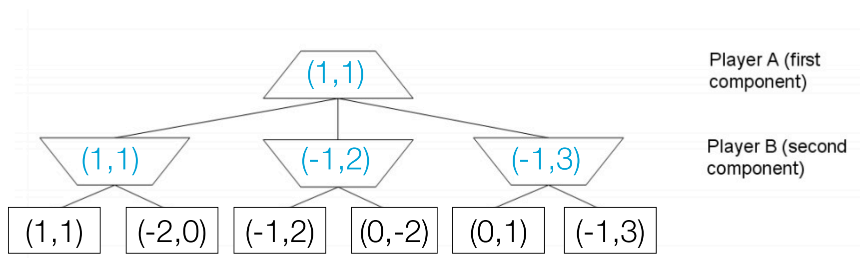
1 Nearly Zero Sum Games

The standard Minimax algorithm calculates worst-case values in a *zero-sum* two player game, i.e. a game in which for all terminal states s, the utilities for players A (MAX) and B (MIN) obey $U_A(s) + U_B(s) = 0$. In the zero sum case, we know that $U_A(s) = -U_B(s)$ and so we can think of player B as simply minimizing $U_A(s)$.

In this problem, you will consider the *non zero-sum* generalization in which the sum of the two players' utilities are not necessarily zero. Because player A's utility no longer determines player B's utility exactly, the leaf utilities are written as pairs $(U_A; U_B)$, with the first and second component indicating the utility of that leaf to A and B respectively. In this generalized setting, A seeks to maximize U_A, the first component, while B seeks to maximize U_B, the second component.



1. Propagate the terminal utility pairs up the tree using the appropriate generalization of the minimax algorithm on this game tree. Fill in the values (as pairs) at each of the internal node. Assume that each player maximizes their own utility.



2. Briefly explain why no alpha-beta style pruning is possible in the general non-zero sum case. *Hint*: think first about the case where $U_A(s) = U_B(s)$ for all nodes.

   The values that the first and second player are trying to maximize are independent, so we no longer have situations where we know that one player will never let the other player down a particular branch of the game tree.

   For instance, in the case where $U_A = U_B$, the problem reduces to searching for the max-valued leaf, which could appear anywhere in the tree.

3. For minimax, we know that the value v computed at the root (say for player A = MAX) is a worst-case value. This means that if the opponent MIN doesn't act optimally, the actual outcome v' for MAX can only be better, never worse than v.

   In the general non-zero sum setup, can we say that the value U_A computed at the root for player A is also a worst-case value in this sense, or can A's outcome be worse than the computed U_A if B plays sub-optimally? Briefly justify.

   A's outcome can be worse than the computed v_A. For instance, in the example game, if B chooses $(-2, 0)$ over $(1, 1)$, then A's outcome will decrease from 1 to 0.

4. Now consider the nearly zero sum case, in which $|U_A(s) + U_B(s)| \leq \epsilon$ at all terminal nodes s for some ϵ which is known in advance. For example, the previous game tree is nearly zero sum for $\epsilon = 2$.

   In the nearly zero sum case, pruning is possible. Draw an X in each node in this game tree which could be pruned with the appropriate generalization of alpha-beta pruning. Assume that the exploration is being done in the standard left to right depth-first order and the value of ϵ is known to be 2. Make sure you make use of ϵ in your reasoning.

   We can prune the node $(0, -2)$ and if we allow pruning on equality then we can also prune $(-1, 3)$. See answers to the next two problems for the reasoning.

5. Give a general condition under which a child n of a B node (MIN node) b can be pruned. Your condition should generalize α-pruning and should be stated in terms of quantities such as the utilities $U_A(s)$ and/or $U_B(s)$ of relevant nodes s in the game tree, the bound ϵ, and so on. Do not worry about ties.

   The pruning condition is $\boxed{U_B > \epsilon - \alpha}$.

   Consider the standard minimax algorithm (zero-sum game) written in this more general 2 agent framework. The maximizer agent tries to maximize its utility, U_A, while the second agent (B) tries to minimize player A's value. This is equivalent to saying that player B wants to maximize $-U_A$. Therefore we say that the utility of player B is $U_B = -U_A$ in the standard minimax situation.

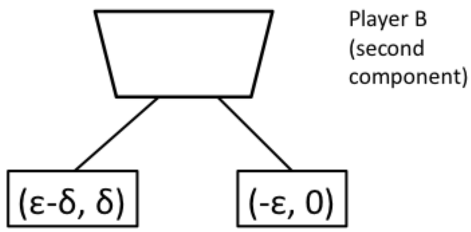
   Recall from lecture that in standard $\alpha - \beta$ pruning we allow a pruning action to occur under a minimizer (player B) node if $v < \alpha$. Under our more general 2 agent framework this condition is equivalent to saying you can prune under player B if $U_A = -U_B < \alpha \Rightarrow U_B > -\alpha$.

   For this question we have an ϵ-sum game so we need to add an additional requirement of ϵ on U_B before pruning can occur. In particular, we know that $|U_A + U_B| \leq \epsilon \Rightarrow U_A \leq \epsilon - U_B$. We want to prune if this upper bound is less than α because then we guarantee that max has an better alternative elsewhere in the tree. Therefore, in order to prune we must satisfy $\epsilon - U_B < \alpha \Rightarrow U_B > \epsilon - \alpha$.

6. In the nearly zero sum case with bound ϵ, what guarantee, if any, can we make for the actual outcome u' for player A (in terms of the value U_A of the root) in the case where player B acts sub-optimally?

$\boxed{u' \geq U_A - 2\epsilon}$

To get intuition about this problem we will first think about the worst case scenario that can occur for player A. Consider the small game tree below for $\delta > 0$:



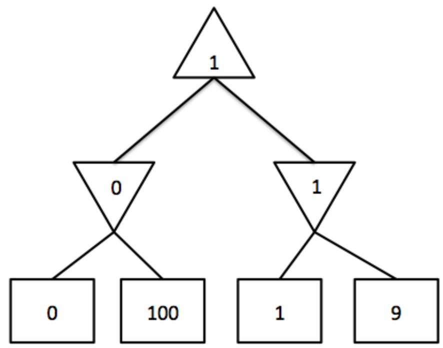
The optimal action for player B to take would be $(\epsilon - \delta, \delta)$. If player B acts optimally then player A will end up with a value of $U_A = \epsilon - \delta$. Now, consider what would happen if player B acted suboptimally, namely if player B chose $(-\epsilon, 0)$. Then player A would receive an actual outcome of $u' = -\epsilon$. So, we see that $u' \geq U_A - 2\epsilon + \delta$. Now let δ be arbitrarily small and you converge to the bound boxed above.

Thus far we have just shown (by example) that we cannot hope for a better guarantee than $u' \geq U_A - 2\epsilon$ (if someone claimed a better guarantee, the above would be a counter-example to that (faulty) claim). We are left with showing that this bound actually holds true. To do so, consider what happens when Player B plays suboptimally. By definition of suboptimality, that means the outcome of the game for player B is not the optimal U_B but some lower value $U'_B = U_B x$ with $x > 0$. This will have the maximum effect on player A's pay-off when for the optimal outcome we had $U_A + U_B = \epsilon$, but for the suboptimal outcome we have $U'_A + U'_B = U'_A + U_B x = \epsilon$. From the first equation we have $U_B = \epsilon U_A$, substituting into the second equation gives us: $U'_A = \epsilon\epsilon + U_A = U_A 2\epsilon$ as the worst-case outcome for player A.

2 Minimax and Expectimax

In this problem, you will investigate the relationship between expectimax trees and minimax trees for zero-sum two player games. Imagine you have a game which alternates between player 1 (max) and player 2. The game begins in state s_0, with player 1 to move. Player 1 can either choose a move using minimax search, or expectimax search, where player 2's nodes are chance rather than min nodes.

1. Draw a (small) game tree in which the root node has a larger value if expectimax search is used than if minimax is used, or argue why it is not possible.



We can see here that the above game tree has a root value of 1 for the minimax strategy. If we instead switch to expectimax and replace the min nodes with chance nodes, the root of the tree takes on a value of 50 and the optimal action changes for MAX.

2. Draw a (small) game tree in which the root node has a larger value if minimax search is used than if expectimax is used, or argue why it is not possible.

Optimal play for MIN, by definition, means the best moves for MIN to obtain the lowest value possible. Random play includes moves that are not optimal. Assuming there are no ties (no two leaves have the same value), expectimax will always average in suboptimal moves. Averaging a suboptimal move (for MIN) against an optimal move (for MIN) will always increase the expected outcome.

With this in mind, we can see how there is no game tree where the value of the root for expectimax is lower than the value of the root for minimax. One is optimal play – the other is suboptimal play averaged with optimal play, which by definiton leads to a higher value for MIN.

3. Under what assumptions about player 2 should player 1 use minimax search rather than expectimax search to select a move?

   Player 1 should use minimax search if he/she expects player 2 to move optimally.

4. Under what assumptions about player 2 should player 1 use expectimax search rather than minimax search?

   If player 1 expects player 2 to move randomly, he/she should use expectimax search. This will optimize for the maximum expected value.

5. Imagine that player 1 wishes to act optimally (rationally), and player 1 knows that player 2 also intends to act optimally. However, player 1 also knows that player 2 (mistakenly) believes that player 1 is moving uniformly at random rather than optimally. Explain how player 1 should use this knowledge to select a move. Your answer should be a precise algorithm involving a game tree search, and should include a sketch of an appropriate game tree with player 1's move at the root. Be clear what type of nodes are at each ply and whose turn each ply represents.

   Use two games trees:

   Game tree 1: max is replaced by a chance node. Solve this tree to find the policy of MIN.

   Game tree 2: the original tree, but MIN doesn't have any choices now, instead is constrained to follow the policy found from Game Tree 1.

CS188 Spring 2014 Section 4: MDPs

1 MDPs: Micro-Blackjack

In micro-blackjack, you repeatedly draw a card (with replacement) that is equally likely to be a 2, 3, or 4. You can either Draw or Stop if the total score of the cards you have drawn is less than 6. Otherwise, you must Stop. When you Stop, your utility is equal to your total score (up to 5), or zero if you get a total of 6 or higher. When you Draw, you receive no utility. There is no discount ($\gamma = 1$).

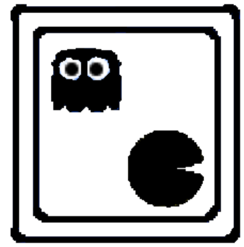
1. What are the states and the actions for this MDP?

2. What is the transition function and the reward function for this MDP?

3. Give the optimal policy for this MDP.

4. What is the smallest number of rounds (k) of value iteration for which this MDP will have its exact values (if value iteration will never converge exactly, state so).

2 Pursuit Evasion



Pacman is trapped in the following 2 by 2 maze with a hungry ghost (the horror)! When it is his turn to move, Pacman must move one step horizontally or vertically to a neighboring square. When it is the ghost's turn, he must also move one step horizontally or vertically. The ghost and Pacman alternate moves. After every move (by either the ghost or Pacman) if Pacman and the ghost occupy the same square, Pacman is eaten and receives utility -100. Otherwise, he receives a utility of 1. The ghost attempts to minimize the utility that Pacman receives. **Assume the ghost makes the first move**.

For example, with a discount factor of $\gamma = 1.0$, if the ghost moves down, then Pacman moves left, Pacman earns a reward of 1 after the ghost's move and -100 after his move for a total utility of -99.

Note that this game is not guaranteed to terminate.

1. Assume a discount factor $\gamma = 0.5$, where the discount factor is applied once every time either Pacman or the ghost moves. What is the minimax value of the truncated game after 2 ghost moves and 2 Pacman moves? (Hint: you should not need to build the minimax tree)

2. Assume a discount factor $\gamma = 0.5$. What is the minimax value of the complete (infinite) game? (Hint: you should not need to build the minimax tree)

3. Why is value iteration superior to minimax for solving this game?

4. This game is similar to an MDP because rewards are earned at every timestep. However, it is also an adversarial game involving decisions by two agents.

   Let s be the state (e.g. the position of Pacman and the ghost), and let $A_P(s)$ be the space of actions available to Pacman in state s (and similarly let $A_G(s)$ be the space of actions available to the ghost). Let $N(s, a) = s'$ denote the successor function (given a starting state s, this function returns the state s' which results after taking action a). Finally, let $R(s)$ denote the utility received after moving to state s.

   Write down an expression for $P^*(s)$, the value of the game to Pacman as a function of the current state s (analogous to the Bellman equations). Use a discount factor of $\gamma = 1.0$. Hint: your answer should include $P^*(s)$ on the right hand side.

   $P^*(s) =$

CS188 Spring 2014 Section 4: MDPs

1 MDPs: Micro-Blackjack

In micro-blackjack, you repeatedly draw a card (with replacement) that is equally likely to be a 2, 3, or 4. You can either Draw or Stop if the total score of the cards you have drawn is less than 6. Otherwise, you must Stop. When you Stop, your utility is equal to your total score (up to 5), or zero if you get a total of 6 or higher. When you Draw, you receive no utility. There is no discount ($\gamma = 1$).

1. What are the states and the actions for this MDP?

   The state is the current sum of your cards, plus a terminal state:

   $$0, 2, 3, 4, 5, Done$$

   (answers which include 1,6,7,8,9, a *Bust* state, or just "the current sum of your cards plus a terminal state" are acceptable.)

   The actions are $\{Draw, Stop\}$.

2. What is the transition function and the reward function for this MDP?

   The transition function is

   $$T(s, Stop, Done) = 1$$
   $$T(s, Draw, s') = \begin{cases} 1/3 \text{ if } s' - s \in \{2,3,4\} \\ 1/3 \text{ if } s = 2 \text{ and } s' = Done \\ 2/3 \text{ if } s = 3 \text{ and } s' = Done \\ 1 \text{ if } s \in \{4,5\} \text{ and } s' = Done \\ 0 \text{ otherwise} \end{cases}$$

   The reward function is

   $$R(s, Stop, s') = s, s \leq 5$$
   $$R(s, a, s') = 0 \text{ otherwise}$$

3. Give the optimal policy for this MDP.

   In general, for finding the optimal policy for an MDP, we would use some method like value iteration followed by policy extraction. However, in this particular case, it is simple to work out that the opimal policy would be **Draw if $s \leq 2$, Stop otherwise.**

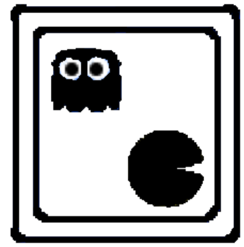
   For completeness, we give below the value iteration steps based on the states and transition functions described above. The optimal policy is given by taking the *argmax* instead of *max*, in the final iteration of value iteration.

V	0	2	3	4	5	Done
V_0	0	0	0	0	0	0
V_1	0	2	3	4	5	0
V_2	3	3	3	4	5	0
V_3	10/3	3	3	4	5	0
Policy Extraction	$10/3_{Draw}$	3_{Draw}	3_{Stop}	4_{Stop}	5_{Stop}	0_{Stop}

4. What is the smallest number of rounds (k) of value iteration for which this MDP will have its exact values (if value iteration will never converge exactly, state so).

   3

2 Pursuit Evasion



Pacman is trapped in the following 2 by 2 maze with a hungry ghost (the horror)! When it is his turn to move, Pacman must move one step horizontally or vertically to a neighboring square. When it is the ghost's turn, he must also move one step horizontally or vertically. The ghost and Pacman alternate moves. After every move (by either the ghost or Pacman) if Pacman and the ghost occupy the same square, Pacman is eaten and receives utility -100. Otherwise, he receives a utility of 1. The ghost attempts to minimize the utility that Pacman receives. **Assume the ghost makes the first move**.

For example, with a discount factor of $\gamma = 1.0$, if the ghost moves down, then Pacman moves left, Pacman earns a reward of 1 after the ghost's move and -100 after his move for a total utility of -99.

Note that this game is not guaranteed to terminate.

1. Assume a discount factor $\gamma = 0.5$, where the discount factor is applied once every time either Pacman or the ghost moves. What is the minimax value of the truncated game after 2 ghost moves and 2 Pacman moves? (Hint: you should not need to build the minimax tree)

$$1 + 0.5 + 0.25 + 0.125 = 1.875$$

2. Assume a discount factor $\gamma = 0.5$. What is the minimax value of the complete (infinite) game? (Hint: you should not need to build the minimax tree)

   2

3. Why is value iteration superior to minimax for solving this game? Value iteration takes advantage of repeated states to efficiently solve for the optimal policy. Even a truncated minimax tree increases in size exponentially as you increase the search depth.

4. This game is similar to an MDP because rewards are earned at every timestep. However, it is also an adversarial game involving decisions by two agents.

   Let s be the state (e.g. the position of Pacman and the ghost), and let $A_P(s)$ be the space of actions available to Pacman in state s (and similarly let $A_G(s)$ be the space of actions available to the ghost). Let $N(s, a) = s'$ denote the successor function (given a starting state s, this function returns the state s' which results after taking action a). Finally, let $R(s)$ denote the utility received after moving to state s.

   Write down an expression for $P^*(s)$, the value of the game to Pacman as a function of the current state s (analogous to the Bellman equations). Use a discount factor of $\gamma = 1.0$. Hint: your answer should include $P^*(s)$ on the right hand side.

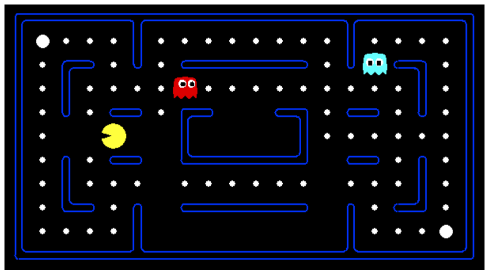
   $P^*(s) =$

$$P^*(s) = \max_{a \in A_P(s)} R(N(s,a)) + \min_{a' \in A_G(N(s,a))} R(N(N(s,a),a')) + P^*(N(N(s,a),a'))$$

CS188 Spring 2014 Section 5: Reinforcement Learning

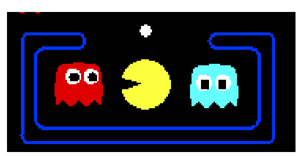
1 Learning with Feature-based Representations

We would like to use a Q-learning agent for Pacman, but the state size for a large grid is too massive to hold in memory (just like at the end of Project 3). To solve this, we will switch to feature-based representation of Pacman's state. Here's a Pacman board to refresh your memory:



1. What features would you extract from a Pacman board to judge the expected outcome of the game?

2. Say our two minimal features are the number of ghosts within 1 step of Pacman (F_g) and the number of food pellets within 1 step of Pacman (F_p). For this pacman board:



Extract the two features (calculate their values).

3. With Q Learning, we train off of a few episodes, so our weights begin to take on values. Right now $w_g = 100$ and $w_p = -10$. Calculate the Q value for the state above.

4. We receive an episode, so now we need to update our values. An episode consists of a start state s, an action a, an end state s', and a reward $R(s, a, s')$. The start state of the episode is the state above (where you already calculated the feature values and the expected Q value). The next state has feature values $F_g = 0$ and $F_p = 2$ and the reward is 50. Assuming a discount of 0.5, calculate the new estimate of the Q value for s based on this episode.

5. With this new estimate and a learning rate (α) of 0.5, update the weights for each feature.

6. Good job on updating the weights. Now let's think about this entire process one step back. What values do we learn in this process (assuming features are defined)? When we have completed learning, how do we tell if Pacman does a good job?

7. In some sense, we can think about this entire process, on a meta level, as an input we control that produces an output that we would like to maximize. If you have a magical function ($F(input)$) that maps an input to an output you would like to maximize, what techniques (from math, CS, etc) can we use to search for the best inputs? Keep in mind that the magical function is a black box.

8. Now say we can calculate the derivative of the magical function, $F'(input)$, giving us a gradient or slope. What techniques can we use now?

2 Odds and Ends

1. When using features to represent the Q-function is it guaranteed that the feature-based Q-learning finds the same optimal $Q*$ as would be found when using a tabular representation for the Q-function?

2. Why is temporal difference (TD) learning of Q-values (Q-learning) superior to TD learning of values?

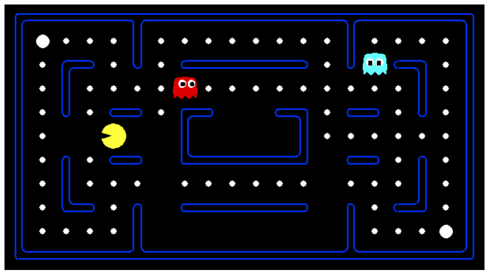
3. Can all MDPs be solved using expectimax search? Justify your answer.

4. When learning with ϵ-greedy action selection, is it a good idea to decrease ϵ to 0 with time? Why or why not?

CS188 Spring 2014 Section 5: Reinforcement Learning

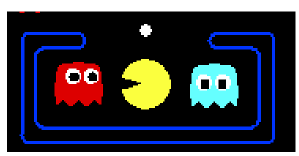
1 Learning with Feature-based Representations

We would like to use a Q-learning agent for Pacman, but the state size for a large grid is too massive to hold in memory (just like at the end of Project 3). To solve this, we will switch to feature-based representation of Pacman's state. Here's a Pacman board to refresh your memory:



1. What features would you extract from a Pacman board to judge the expected outcome of the game?
   The usual ones, for example as in Project 2.

2. Say our two minimal features are the number of ghosts within 1 step of Pacman (F_g) and the number of food pellets within 1 step of Pacman (F_p). For this pacman board:

   Extract the two features (calculate their values).
   $f_g = 2, f_p = 1$

3. With Q Learning, we train off of a few episodes, so our weights begin to take on values. Right now $w_g = 100$ and $w_p = -10$. Calculate the Q value for the state above.
   First of all, the Q value will not depend on what action is taken, because the features we extract do not depend on the action, only the state.

$$Q(s, a) = w_g * f_g + w_p * f_p = 100 * 2 + -10 * 1 = 190$$

4. We receive an episode, so now we need to update our values. An episode consists of a start state s, an action a, an end state s', and a reward $R(s, a, s')$. The start state of the episode is the state above (where you already calculated the feature values and the expected Q value). The next state has feature values $F_g = 0$ and $F_p = 2$ and the reward is 50. Assuming a discount of 0.5, calculate the new estimate of the Q value for s based on this episode.

$$
\begin{aligned}
Q_{new}(s, a) &= R(s, a, s') + \gamma * \max_{a'} Q(s', a') \\
&= 50 + 0.5 * (100 * 0 + -10 * 2) \\
&= 40
\end{aligned}
$$

5. With this new estimate and a learning rate (α) of 0.5, update the weights for each feature.

$$
\begin{aligned}
w_g = w_g + \alpha * (Q_{new}(s, a) - Q(s, a)) * f_g(s, a) = 100 + 0.5 * (40 - 190) * 2 = -50 \\
w_p = w_p + \alpha * (Q_{new}(s, a) - Q(s, a)) * f_p(s, a) = -10 + 0.5 * (40 - 190) * 1 = -85
\end{aligned}
$$

Note that now the weight on ghosts is negative, which makes sense (ghosts should indeed be avoided). Although the weight on food pellets is now also negative, the difference between the two weights is now much lower.

6. Good job on updating the weights. Now let's think about this entire process one step back. What values do we learn in this process (assuming features are defined)? When we have completed learning, how do we tell if Pacman does a good job?
The values we learn are the feature weights. Once Pacman completes its learning, we will evaluate its performance by running the game and looking at the win/lose outcomes and the reward accrued from the start state.

7. In some sense, we can think about this entire process, on a meta level, as an input we control that produces an output that we would like to maximize. If you have a magical function ($F(input)$) that maps an input to an output you would like to maximize, what techniques (from math, CS, etc) can we use to search for the best inputs? Keep in mind that the magical function is a black box.
Of course, you can use random search (changing values indiscriminately), or try some evenly distributed values in the possible range (which is sometimes called beam search).

   Also, remember that earlier in the course we talked about local search, which includes techniques such as hill climbing (going in the direction of maximum positive change), simulated annealing (where the rate of random exploration is lowered as time goes on), and genetic algorithms. These are all possible solutions.

8. Now say we can calculate the derivative of the magical function, $F'(input)$, giving us a gradient or slope. What techniques can we use now?
Gradient descent, or many techniques from calculus and optimization developed for such problems. Although these techniques will be very useful to you as an artificial intelligence researcher, don't worry: we will not expect you to know how to use any of them on the exams!

2 Odds and Ends

1. When using features to represent the Q-function is it guaranteed that the feature-based Q-learning finds the same optimal $Q*$ as would be found when using a tabular representation for the Q-function?
   No, if the optimal Q-function $Q*$ cannot be represented as a weighted combination of features, then the feature-based representation would not have the expressive power to find it.

2. Why is temporal difference (TD) learning of Q-values (Q-learning) superior to TD learning of values?
   Because if you use temporal difference learning on the values, it is hard to extract a policy from the learned values. Specifically, you would need to know the transition model T. For TD learning of Q-values, the policy can be extracted directly by taking $\pi(s) = \arg\max_a Q(s, a)$.

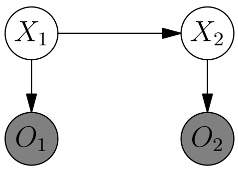
3. Can all MDPs be solved using expectimax search? Justify your answer.
   No, MDPs with self loops lead to infinite expectimax trees. Unlike search problems, this issue cannot be addressed with a graph-search variant.

4. When learning with ϵ-greedy action selection, is it a good idea to decrease ϵ to 0 with time? Why or why not?
   Yes, especially when using on-policy learning methods. The reason is that as the agent learns the actual optimal policy for the world, it should switch from a mix of exploration and exploitation to mostly exploitation (unless the world is changing, in which case it should always keep exploring).

CS188 Spring 2014 Section 6: Hidden Markov Models

1 Basic computations

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.

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.

Use the Viterbi algorithm to compute the maximum probability sequence X_1, X_2. Show your work.

True or false: Variable elimination is generally more accurate than the Forward algorithm. Explain your answer.

2 Tracking a Jabberwock

You have been put in charge of a Jabberwock for your friend Lewis. The Jabberwock is kept in a large tugley wood which is conveniently divided into an $N \times N$ grid. It wanders freely around the N^2 possible cells. At each time step $t = 1, 2, 3, \ldots$, the Jabberwock is in some cell $X_t \in \{1, \ldots, N\}^2$, and it moves to cell X_{t+1} randomly as follows: with probability $1 - \epsilon$, it chooses one of the (up to 4) valid neighboring cells uniformly at random; with probability ϵ, it uses its magical powers to teleport to a random cell uniformly at random among the N^2 possibilities (it might teleport to the same cell). Suppose $\epsilon = \frac{1}{2}$, $N = 10$ and that the Jabberwock always starts in $X_1 = (1, 1)$.

1. Compute the probability that the Jabberwock will be in $X_2 = (2, 1)$ at time step 2. What about $\Pr(X_2 = (4, 4))$?

   $\Pr(X_2 = (2, 1)) =$

   $\Pr(X_2 = (4, 4)) =$

   At each time step t, you dont see X_t but see E_t, which is the row that the Jabberwock is in; that is, if $X_t = (r, c)$, then $E_t = r$. You still know that $X_1 = (1, 1)$.

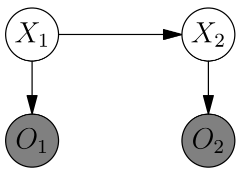
2. Suppose we see that $E_1 = 1$, $E_2 = 2$, $E_3 = 10$. Fill in the following table with the distribution over X_t after each time step, taking into consideration the evidence. Your answer should be concise. *Hint*: you should not need to do any heavy calculations.

t	$\Pr(X_t, e_{1:t-1})$	$\Pr(X_t, e_{1:t})$
1		
2		

CS188 Spring 2014 Section 6: Hidden Markov Models

1 Basic computations

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.

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.

X_1	$\Pr(X_1, O_1 = A)$
0	$0.3 \cdot 0.9$
1	$0.7 \cdot 0.5$

X_2	$\Pr(X_2, O_1 = A, O_2 = B)$
0	$0.1 \cdot [0.4 \cdot (0.3 \cdot 0.9) + 0.8 \cdot (0.7 \cdot 0.5)] = 0.0388$
1	$0.5 \cdot [0.6 \cdot (0.3 \cdot 0.9) + 0.2 \cdot (0.7 \cdot 0.5)] = 0.1160$

Use the Viterbi algorithm to compute the maximum probability sequence X_1, X_2. Show your work.

X_1	$\Pr(X_1, O_1 = A)$
0	$0.3 \cdot 0.9$
1	$0.7 \cdot 0.5$

X_2	$\max_{x_1} \Pr(X_1 = x_1, X_2, O_1 = A, O_2 = B)$	arg max
0	$0.1 \cdot \max(0.4 \cdot (0.3 \cdot 0.9), 0.8 \cdot (0.7 \cdot 0.5)) = 0.1 \cdot \max(0.108, 0.28) = 0.028$	$X_1 = 1$
1	$0.5 \cdot \max(0.6 \cdot (0.3 \cdot 0.9), 0.2 \cdot (0.7 \cdot 0.5)) = 0.5 \cdot \max(0.162, 0.07) = 0.081$	$X_1 = 0$

Thus, in the maximum probability sequence, $X_2 = 1$ and $X_1 = 0$.

True or false: Variable elimination is generally more accurate than the Forward algorithm. Explain your answer.

They both perform exact inference.

2 Tracking a Jabberwock

You have been put in charge of a Jabberwock for your friend Lewis. The Jabberwock is kept in a large tugley wood which is conveniently divided into an $N \times N$ grid. It wanders freely around the N^2 possible cells. At each time step $t = 1, 2, 3, \ldots$, the Jabberwock is in some cell $X_t \in \{1, \ldots, N\}^2$, and it moves to cell X_{t+1} randomly as follows: with probability $1 - \epsilon$, it chooses one of the (up to 4) valid neighboring cells uniformly at random; with probability ϵ, it uses its magical powers to teleport to a random cell uniformly at random among the N^2 possibilities (it might teleport to the same cell). Suppose $\epsilon = \frac{1}{2}$, $N = 10$ and that the Jabberwock always starts in $X_1 = (1, 1)$.

1. Compute the probability that the Jabberwock will be in $X_2 = (2, 1)$ at time step 2. What about $\Pr(X_2 = (4, 4))$?

   $\Pr(X_2 = (2, 1)) = 1/2 \cdot 1/2 + 1/2 \cdot 1/100 = 0.255$

   $\Pr(X_2 = (4, 4)) = 1/2 \cdot 1/100 = 0.005$

   At each time step t, you dont see X_t but see E_t, which is the row that the Jabberwock is in; that is, if $X_t = (r, c)$, then $E_t = r$. You still know that $X_1 = (1, 1)$.

2. Suppose we see that $E_1 = 1$, $E_2 = 2$, $E_3 = 10$. Fill in the following table with the distribution over X_t after each time step, taking into consideration the evidence. Your answer should be concise. *Hint*: you should not need to do any heavy calculations.

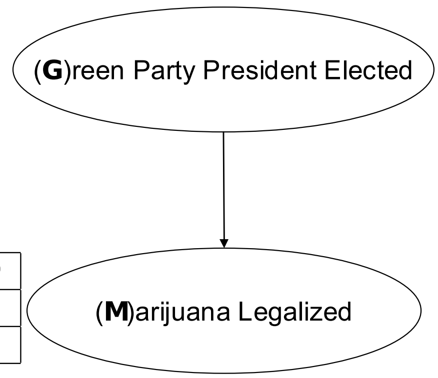
t	$\Pr(X_t, e_{1:t-1})$	$\Pr(X_t, e_{1:t})$
1	(1,1) : 1.0, (others) : 0.0	(1,1): 1.0, (others): 0.0
2	(1,2), (2,1): 51/200, (others): 1/200	(2,1): 51/200, (2,2+): 1/200

CS188 Spring 2014 Section 7: Probability and Bayes Nets

1 Green Party President

It's election year again! In a parallel universe the Green Party is running for presidency. Pundits believe that Green Party presidents are more likely to legalize marijuana than candidates from other parties, but legalization could occur under any administration. Armed with the power of probability, the analysts model the situation with the Bayes Net below.

	+g	-g
P(G)	0.1	0.9



	P(+m | G)	P(-m | G)
+g	0.667	0.333
-g	0.25	0.75

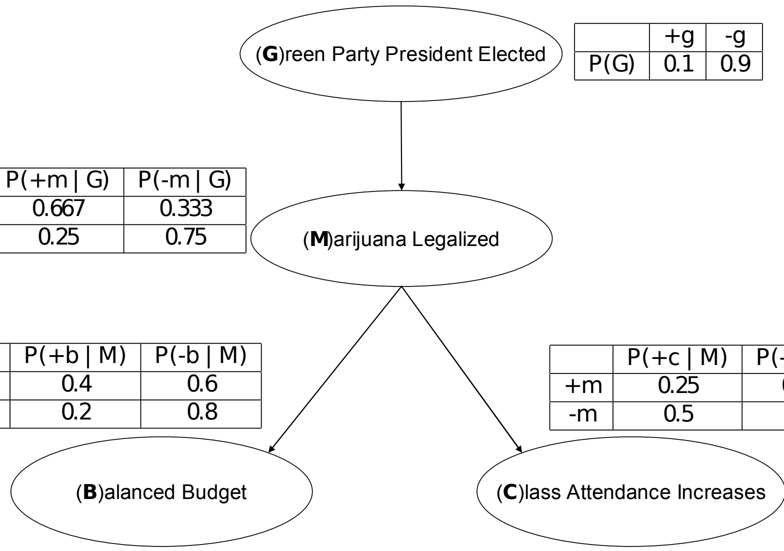
1. Fill in the joint probability table over G and M.

G	M	P(G, M)
+g	+m	
+g	-m	
-g	+m	
-g	-m	

2. What is P(+m), the marginal probability that marijuana is legalized?

3. News agencies air 24/7 coverage of the recent legalization of marijuana (+m), but you can't seem to find out who won the election. What is the conditional probability P(+g | +m) that a Green Party president was elected?

We can make better inferences if we observe more evidence. On the next page, we will expand on the model (Bayes net) by introducing two new random variables: whether the budget is balanced (B), and whether class attendance increases (C). The expanded Bayes net and conditional distributions are shown below.



	+g	-g
P(G)	0.1	0.9

	P(+m | G)	P(-m | G)
+g	0.667	0.333
-g	0.25	0.75

	P(+b | M)	P(-b | M)
+m	0.4	0.6
-m	0.2	0.8

	P(+c | M)	P(-c | M)
+m	0.25	0.75
-m	0.5	0.5

4. The full joint distribution is given below. Fill in the missing values.

G	M	B	C	$P(G, M, B, C)$	G	M	B	C	$P(G, M, B, C)$
+	+	+	+	1/150	-	+	+	+	9/400
+	+	+	-		-	+	+	-	27/400
+	+	-	+	1/100	-	+	-	+	27/800
+	+	-	-	3/100	-	+	-	-	81/800
+	-	+	+	1/300	-	-	+	+	27/400
+	-	+	-	1/300	-	-	+	-	27/400
+	-	-	+		-	-	-	+	
+	-	-	-	1/75	-	-	-	-	27/100

5. Compute the following quantities. You may use either the full joint distribution or the conditional tables, whichever is more convenient.

(a) P(+b | +m) =

(b) P(+b | +m, +g) =

(c) P(+b) =

(d) P(+c | +b) =

6. Now, add a node S to the Bayes net that reflects the possibility that a new scientific study could influence the probability that marijuana is legalized. Assume that the study does not directly influence B or C. Draw the new Bayes net below. Which CPT or CPT's need to be modified?

7. Consider your augmented model. Just based on the structure, which of the following are guaranteed to be true, and which are guaranteed to be false?

   (i) $B \perp\!\!\!\perp G$

   (ii) $C \perp\!\!\!\perp G|M$

   (iii) $G \perp\!\!\!\perp S$

   (iv) $G \perp\!\!\!\perp S|M$

   (v) $G \perp\!\!\!\perp S|B$

   (vi) $B \perp\!\!\!\perp C$

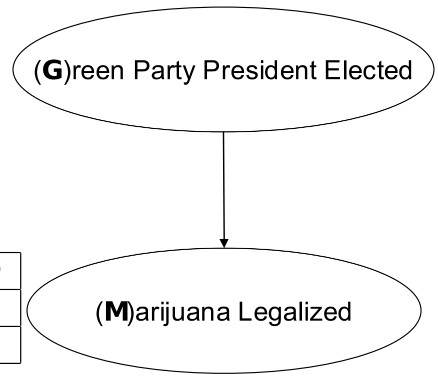
   (vii) $B \perp\!\!\!\perp C|G$

CS188 Spring 2014 Section 7: Probability and Bayes Nets

1 Green Party President

It's election season, and the chosen president may or may not be the Green Party candidate. Pundits believe that Green Party presidents are more likely to legalize marijuana than candidates from other parties, but legalization could occur under any administration. Armed with the power of probability, the analysts model the situation with the Bayes Net below.

	+g	-g
P(G)	0.1	0.9



	P(+m | G)	P(-m | G)
+g	0.667	0.333
-g	0.25	0.75

1. What is P(+m), the marginal probability that marijuana is legalized?

   $\mathrm{P}(+\mathrm{m}) = \mathrm{P}(+\mathrm{m}, +\mathrm{g}) + \mathrm{P}(+\mathrm{m}, -\mathrm{g}) = \mathrm{P}(+\mathrm{m} \mid +\mathrm{g})\mathrm{P}(+\mathrm{g}) + \mathrm{P}(+\mathrm{m} \mid -\mathrm{g})\mathrm{P}(-\mathrm{g}) = \frac{2}{3} \cdot \frac{1}{10} + \frac{1}{4} \cdot \frac{9}{10} = \frac{7}{24}$

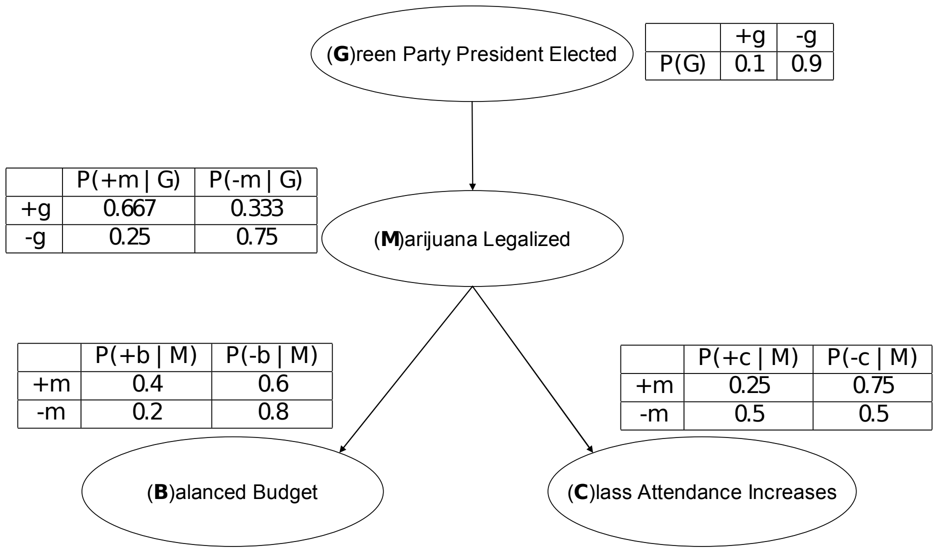
2. News agencies air 24/7 coverage of the recent legalization of marijuana (+m), but you can't seem to find out who won the election. What is the conditional probability P(+g | +m) that a Green Party president was elected?

$$\mathrm{P}(+\mathrm{g} \mid +\mathrm{m}) = \frac{\mathrm{P}(+\mathrm{g}, +\mathrm{m})}{\mathrm{P}(+\mathrm{m})} = \frac{\mathrm{P}(+\mathrm{m} \mid +\mathrm{g})\mathrm{P}(+\mathrm{g})}{\mathrm{P}(+\mathrm{m})} = \frac{\frac{2}{3} \cdot \frac{1}{10}}{\frac{7}{24}} = \frac{8}{35}$$

3. Fill in the joint probability table over G and M.

G	M	P(G, M)
+g	+m	1/15
+g	-m	1/30
-g	+m	9/40
-g	-m	27/40

We can make better inferences if we observe more evidence. On the next page, we will expand on the model (Bayes net) by introducing two new random variables: whether the budget is balanced (B), and whether class attendance increases (C). The expanded Bayes net and conditional distributions are shown below.



	+g	-g
P(G)	0.1	0.9

	P(+m | G)	P(-m | G)
+g	0.667	0.333
-g	0.25	0.75

	P(+b | M)	P(-b | M)
+m	0.4	0.6
-m	0.2	0.8

	P(+c | M)	P(-c | M)
+m	0.25	0.75
-m	0.5	0.5

4. The full joint distribution is given below. Fill in the missing values.

G	M	B	C	$P(G, M, B, C)$	G	M	B	C	$P(G, M, B, C)$
+	+	+	+	1/150	-	+	+	+	9/400
+	+	+	-	1/50	-	+	+	-	27/400
+	+	-	+	1/100	-	+	-	+	27/800
+	+	-	-	3/100	-	+	-	-	81/800
+	-	+	+	1/300	-	-	+	+	27/400
+	-	+	-	1/300	-	-	+	-	27/400
+	-	-	+	1/75	-	-	-	+	27/100
+	-	-	-	1/75	-	-	-	-	27/100

5. Compute the following quantities. You may use either the full joint distribution or the conditional tables, whichever is more convenient.

(a) P(+b | +m) = $\frac{4}{10}$ (directly from conditional)

(b) P(+b | +m, +g) = $\frac{4}{10}$ (also directly from conditional, since $B \perp\!\!\!\perp G|M$)

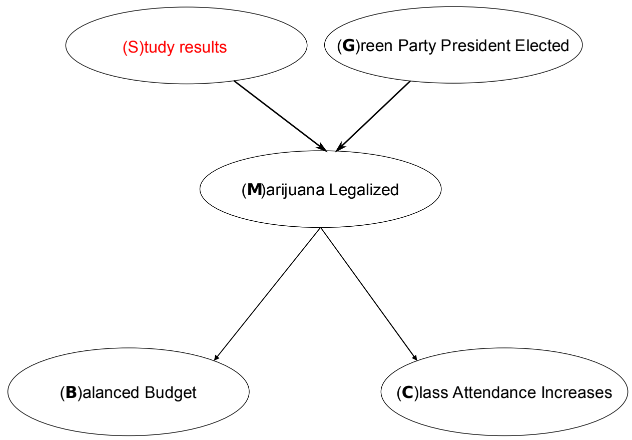
(c) P(+b) = $\sum_{G,M,C}$ P(G,M,+b,C) = $\frac{8}{1200} + \frac{24}{1200} + \frac{4}{1200} + \frac{4}{1200} + \frac{27}{1200} + \frac{81}{1200} + \frac{81}{1200} + \frac{81}{1200} = \frac{31}{120}$

(summed from joint)

(d) P(+c | +b) = $\frac{\text{P(+b, +c)}}{\text{P(+b)}} = \frac{\sum_{G,M} \text{P(G,M,+b,+c)}}{31/120} = \left(\frac{8}{1200} + \frac{4}{1200} + \frac{27}{1200} + \frac{81}{1200}\right) \cdot \frac{120}{31} = \frac{12}{31}$

(summed from joint)

6. Now, add a node S to the Bayes net that reflects the possibility that a new scientific study could influence the probability that marijuana is legalized. Assume that the study does not directly influence B or C. Draw the new Bayes net below. Which CPT or CPT's need to be modified?



$P(M|G)$ will become $P(M|G,S)$, and will contain 8 entries instead of 4.

7. Consider your augmented model. Just based on the structure, which of the following are guaranteed to be true, and which are guaranteed to be false?

(i) $B \perp\!\!\!\perp G$ Not indicated by structure

(ii) $C \perp\!\!\!\perp G|M$ True

(iii) $G \perp\!\!\!\perp S$ True

(iv) $G \perp\!\!\!\perp S|M$ Not indicated by structure

(v) $G \perp\!\!\!\perp S|B$ Not indicated by structure

(vi) $B \perp\!\!\!\perp C$ Not indicated by structure

(vii) $B \perp\!\!\!\perp C|G$ Not indicated by structure

Note that just based on the structure (but without knowing the CPT values), we can assert independence but we cannot assert dependence.

CS188 Spring 2014 Section 8: Inference and Sampling

1 Sampling and Dynamic Bayes Nets

Many people would prefer to eat ice cream on a sunny day than on a rainy day. We can model this situation with a Bayesian network. Suppose we consider the weather, along with a person's ice-cream eating, over the span of two days. We'll have four random variables: W_1 and W_2 stand for the weather on days 1 and 2, which can either be rainy `R` or sunny `S`, and the variables I_1 and I_2 represent whether or not the person ate ice cream on days 1 and 2, and take values `T` (for truly eating ice cream) or `F`.

(a) What conditional independence relationships might be reasonable to assume among these variables?

(b) Suppose you decide to assume for modeling purposes that the weather on each day might depend on the previous day's weather, but that ice cream consumption on day i is independent of ice cream consumption on day $i-1$, conditioned on the weather on day i. Draw a Bayes net encoding these independence assumptions.

Suppose you are given the following conditional probability distributions:

$W_1 = \mathtt{S}$	$W_1 = \mathtt{R}$
.6	.4

(a) $P(W_1)$

	$W_2 = \mathtt{S}$	$W_2 = \mathtt{R}$
$W_1 = \mathtt{S}$	.7	.3
$W_1 = \mathtt{R}$	.5	.5

(b) $P(W_2 \mid W_1)$

	$I = \mathtt{T}$	$I = \mathtt{F}$
$W = \mathtt{S}$	.9	.1
$W = \mathtt{R}$	.2	.8

(c) $P(I \mid W)$

Together with the graph you drew above, these distributions fully specify a joint probability distribution for the four variables. Now we want to do inference, and in particular we'll try approximate inference through sampling. Suppose we sample from the prior to produce the following samples of (W_1, I_1, W_2, I_2) from the ice-cream model:

R, F, R, F	R, F, R, F	S, F, S, T	S, T, S, T	S, T, R, F
R, F, R, T	S, T, S, T	S, T, S, T	S, T, R, F	R, F, S, T

(c) What is $\widehat{P}(W_2 = \mathtt{R})$, the probability that sampling assigns to the event $W_2 = \mathtt{R}$?

(d) Cross off samples rejected by rejection sampling if we're computing $\mathbb{P}(W_2 \mid I_1 = \mathtt{T}, I_2 = \mathtt{F})$

Rejection sampling seems to be wasting a lot of effort, so we decide to switch to likelihood weighting. Assume we generate the following six samples given the evidence $I_1 = \mathtt{T}$ and $I_2 = \mathtt{F}$:

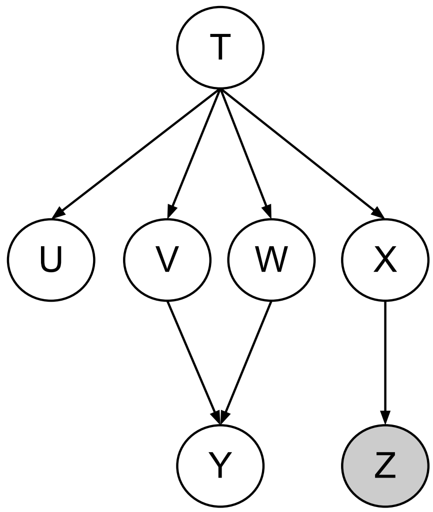
$$(W_1, I_1, W_2, I_2) = \Big\{ \langle \mathtt{S}, \mathtt{T}, \mathtt{R}, \mathtt{F} \rangle, \langle \mathtt{R}, \mathtt{T}, \mathtt{R}, \mathtt{F} \rangle, \langle \mathtt{S}, \mathtt{T}, \mathtt{R}, \mathtt{F} \rangle, \langle \mathtt{S}, \mathtt{T}, \mathtt{S}, \mathtt{F} \rangle, \langle \mathtt{S}, \mathtt{T}, \mathtt{S}, \mathtt{F} \rangle, \langle \mathtt{R}, \mathtt{T}, \mathtt{S}, \mathtt{F} \rangle \Big\}$$

(e) What is the weight of the first sample (S, T, R, F) above?

(f) Use likelihood weighting to estimate $\widehat{P}(W_2 \mid I_1 = \mathtt{T}, I_2 = \mathtt{F})$

2 Variable Elimination

For the Bayes' net below, we are given the query $P(Y \mid +z)$. All variables have binary domains. Assume we run variable elimination to compute the answer to this query, with the following variable elimination ordering: X, T, U, V, W.



Complete the following description of the factors generated in this process:

After inserting evidence, we have the following factors to start out with:

$$P(T), P(U|T), P(V|T), P(W|T), P(X|T), P(Y|V,W), P(+z|X)$$

(a) When eliminating X we generate a new factor f_1 as follows:

$$f_1(T,+z) = \sum_x P(x|T)P(+z|x)$$

(b) This leaves us with the factors:

$$P(T), P(U|T), P(V|T), P(W|T), P(Y|V,W), f_1(T,+z)$$

(c) When eliminating T we generate a new factor f_2 as follows:

(d) This leaves us with the factors:

(e) When eliminating U we generate a new factor f_3 as follows:

(f) This leaves us with the factors:

(g) When eliminating V we generate a new factor f_4 as follows:

(h) This leaves us with the factors:

(i) When eliminating W we generate a new factor f_5 as follows:

(j) This leaves us with the factors:

(k) How would you obtain $P(Y \mid +z)$ from the factors left above:

(l) What is the size of the largest factor that gets generated during the above process?

(m) Does there exist a better elimination ordering (one which generates smaller largest factors)?

CS188 Spring 2014 Section 8: Inference and Sampling

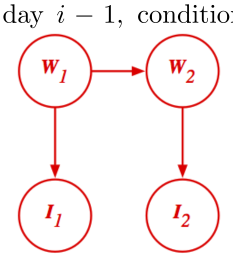
1 Sampling and Dynamic Bayes Nets

Many people would prefer to eat ice cream on a sunny day than on a rainy day. We can model this situation with a Bayesian network. Suppose we consider the weather, along with a person's ice-cream eating, over the span of two days. We'll have four random variables: W_1 and W_2 stand for the weather on days 1 and 2, which can either be rainy `R` or sunny `S`, and the variables I_1 and I_2 represent whether or not the person ate ice cream on days 1 and 2, and take values `T` (for truly eating ice cream) or `F`.

(a) What conditional independence relationships might be reasonable to assume among these variables?

This is very much subjective, and intended as a prompt for discussion. Question (b) gives one example of reasonable assumptions.

(b) Suppose you decide to assume for modeling purposes that the weather on each day might depend on the previous day's weather, but that ice cream consumption on day i is independent of ice cream consumption on day $i-1$, conditioned on the weather on day i. Draw a Bayes net encoding these independence assumptions.



Suppose you are given the following conditional probability distributions:

$W_1 = \texttt{S}$	$W_1 = \texttt{R}$
.6	.4

(a) $P(W_1)$

	$W_2 = \texttt{S}$	$W_2 = \texttt{R}$
$W_1 = \texttt{S}$	.7	.3
$W_1 = \texttt{R}$	.5	.5

(b) $P(W_2|W_1)$

	$I = \texttt{T}$	$I = \texttt{F}$
$W = \texttt{S}$	.9	.1
$W = \texttt{R}$	.2	.8

(c) $P(I|W)$

Together with the graph you drew above, these distributions fully specify a joint probability distribution for the four variables. Now we want to do inference, and in particular we'll try approximate inference through sampling. Suppose we sample from the prior to produce the following samples of (W_1, I_1, W_2, I_2) from the ice-cream model:

~~R, F, R, F~~ ~~R, F, R, F~~ ~~S, F, S, T~~ ~~S, T, S, T~~ S, T, R, F

~~R, F, R, T~~ ~~S, T, S, T~~ ~~S, T, S, T~~ S, T, R, F ~~R, F, S, T~~

(c) What is $\widehat{P}(W_2 = \texttt{R})$, the probability that sampling assigns to the event $W_2 = \texttt{R}$?
Number of samples in which $W_2 = \texttt{R}$: 5. Total number of samples: 10. Answer $5/10 = 0.5$.

(d) Cross off samples rejected by rejection sampling if we're computing $\mathbb{P}(W_2|I_1 = \texttt{T}, I_2 = \texttt{F})$

Rejection sampling seems to be wasting a lot of effort, so we decide to switch to likelihood weighting. Assume we generate the following six samples given the evidence $I_1 = \texttt{T}$ and $I_2 = \texttt{F}$:

$$(W_1, I_1, W_2, I_2) = \left\{ \langle \texttt{S}, \texttt{T}, \texttt{R}, \texttt{F} \rangle, \langle \texttt{R}, \texttt{T}, \texttt{R}, \texttt{F} \rangle, \langle \texttt{S}, \texttt{T}, \texttt{R}, \texttt{F} \rangle, \langle \texttt{S}, \texttt{T}, \texttt{S}, \texttt{F} \rangle, \langle \texttt{S}, \texttt{T}, \texttt{S}, \texttt{F} \rangle, \langle \texttt{R}, \texttt{T}, \texttt{S}, \texttt{F} \rangle \right\}$$

(e) What is the weight of the first sample (S, T, R, F) above?
The weight given to a sample in likelihood weighting is

$$\prod_{\text{Evidence variables } e} \Pr(e|\text{Parents}(e)).$$

In this case, the evidence is $I_1 = \texttt{T}, I_2 = \texttt{F}$. The weight of the first sample is therefore

$$w = \Pr(I_1 = \texttt{T}|W_1 = \texttt{S}) \cdot \Pr(I_2 = \texttt{F}|W_2 = \texttt{R}) = 0.9 \cdot 0.8 = 0.72$$

(f) Use likelihood weighting to estimate $\widehat{P}(W_2|I_1 = \texttt{T}, I_2 = \texttt{F})$
The sample weights are given by

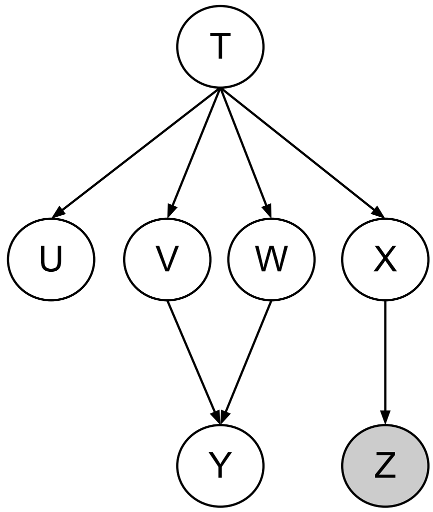
(W_1, I_1, W_2, I_2)	w	(W_1, I_1, W_2, I_2)	w
S, T, R, F	0.72	S, T, S, F	0.09
R, T, R, F	0.16	S, T, S, F	0.09
S, T, R, F	0.72	R, T, S, F	0.02

To compute the probabilities, we thus normalize the weights and find

$$\widehat{P}(W_2 = \mathtt{R} | I_1 = \mathtt{T}, I_2 = \mathtt{F}) = \frac{0.72 + 0.16 + 0.72}{0.72 + 0.16 + 0.72 + 0.09 + 0.09 + 0.02} = 0.889$$
$$\widehat{P}(W_2 = \mathtt{S} | I_1 = \mathtt{T}, I_2 = \mathtt{F}) = 1 - 0.889 = 0.111.$$

2 Variable Elimination

For the Bayes' net below, we are given the query $P(Y \mid +z)$. All variables have binary domains. Assume we run variable elimination to compute the answer to this query, with the following variable elimination ordering: X, T, U, V, W.



Complete the following description of the factors generated in this process:

After inserting evidence, we have the following factors to start out with:

$$P(T), P(U|T), P(V|T), P(W|T), P(X|T), P(Y|V,W), P(+z|X)$$

(a) When eliminating X we generate a new factor f_1 as follows:

$$f_1(T, +z) = \sum_x P(x|T)P(+z|x)$$

(b) This leaves us with the factors:

$$P(T), P(U|T), P(V|T), P(W|T), P(Y|V,W), f_1(T, +z)$$

(c) When eliminating T we generate a new factor f_2 as follows:

$$f_2(U, V, W, +z) = \sum_t P(t)P(U|t)P(V|t)P(W|t)f_1(t, +z).$$

(d) This leaves us with the factors:

$$P(Y|V,W), f_2(U, V, W, +z)$$

(e) When eliminating U we generate a new factor f_3 as follows:

$$f_3(V, W, +z) = \sum_u f_2(u, V, W, +z)$$

(f) This leaves us with the factors:

$$P(Y|V, W), f_3(V, W, +z)$$

(g) When eliminating V we generate a new factor f_4 as follows:

$$f_4(W, Y, +z) = \sum_v f_3(v, W, +z) P(Y|v, W)$$

(h) This leaves us with the factors:

$$f_4(W, Y, +z)$$

(i) When eliminating W we generate a new factor f_5 as follows:

$$f_5(Y, +z) = \sum_w f_4(w, Y, +z)$$

(j) This leaves us with the factors:

$$f_5(Y, +z)$$

(k) How would you obtain $P(Y \mid +z)$ from the factors left above:
Simply renormalize $f_5(Y, +z)$ to obtain $P(Y \mid +z)$. Concretely,

$$P(y \mid +z) = \frac{f_5(y, +z)}{\sum_{y'} f_5(y', +z)}$$

(l) What is the size of the largest factor that gets generated during the above process?
$f_2(U, V, W, +z)$, of size 3.

(m) Does there exist a better elimination ordering (one which generates smaller largest factors)?
Yes, elimination ordering of X, U, T, V, W generates only factors of size 2.

Used Car Purchase

[Adapted from problem 16.11 in Russell & Norvig]

A used car buyer can decide to carry out various tests with various costs (e.g., kick the tires, take the car to a qualified mechanic) and then, depending on the outcome of the tests, decide which car to buy. We will assume that the buyer is deciding whether to buy the car and that there is time to carry out at most one test which costs \$50 and which can help to figure out the quality of the car. A car can be in good shape (of good quality $Q = q^+$) or in bad shape (of bad quality Q=q^-), and the test might help to indicate what shape the car is in. There are only two outcomes for the test T: pass (T=pass) or fail (T=fail). The car costs \$1,500, and its market value is \$2,000 if it is in good shape; if not, \$700 in repairs will be needed to make it in good shape. The buyers estimate is that the car has 70% chance of being in good shape.

1. Draw the decision network that represents this problem.

2. Calculate the expected net gain from buying the car, given no test.

3. Tests can be described by the probability that the car will pass or fail the test given that the car is in good or bad shape. We have the following information:

$$P(T = \text{pass}|Q = q^+) = 0.9$$
$$P(T = \text{pass}|Q = q^-) = 0.2$$

Calculate the probability that the car will pass (or fail) its test, and then the probability that it is in good (or bad) shape given each possible test outcome.

4. Calculate the optimal decisions given either a pass or a fail, and their expected utilities.

5. Calculate the value of (perfect) information of the test. Should the buyer pay for a test?

6. The value of the information in this problem depends greatly on the prior probability $P(Q = q^+)$. What do you think happens to the VPI as you vary $P(Q = q^+)$? What happens when $P(Q = q^+)$ approaches 1? Approaches 0? Approaches 0.5?

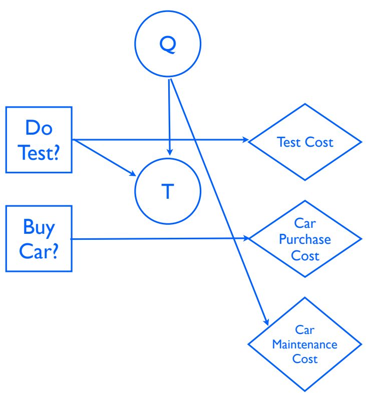
7. If you still have time, try calculating some VPI's for different values of $P(Q = q^+)$ from the previous part. (Using Python or Excel is a good idea here!) Where is/are the break-even point(s)? Where is the maximum?

Used Car Purchase

[Adapted from problem 16.11 in Russell & Norvig]

A used car buyer can decide to carry out various tests with various costs (e.g., kick the tires, take the car to a qualified mechanic) and then, depending on the outcome of the tests, decide which car to buy. We will assume that the buyer is deciding whether to buy the car and that there is time to carry out at most one test which costs $50 and which can help to figure out the quality of the car. A car can be in good shape (of good quality $Q = q^+$) or in bad shape (of bad quality Q=q^-), and the test might help to indicate what shape the car is in. There are only two outcomes for the test T: pass (T=pass) or fail (T=fail). The car costs $1,500, and its market value is $2,000 if it is in good shape; if not, $700 in repairs will be needed to make it in good shape. The buyers estimate is that the car has 70% chance of being in good shape.

1. Draw the decision network that represents this problem.



2. Calculate the expected net gain from buying the car, given no test.

$$
\begin{aligned}
EU(\text{buy}) &= P(Q = q^+) \cdot U(q^+, \text{buy}) + P(Q = q^-) \cdot U(q^-, buy) \\
&= .7 \cdot 500 + 0.3 \cdot -200 = 290
\end{aligned}
$$

3. Tests can be described by the probability that the car will pass or fail the test given that the car is in good or bad shape. We have the following information:

$$P(T = \text{pass}|Q = q^+) = 0.9$$
$$P(T = \text{pass}|Q = q^-) = 0.2$$

Calculate the probability that the car will pass (or fail) its test, and then the probability that it is in good (or bad) shape given each possible test outcome.

$$\begin{aligned}
P(T = \text{pass}) &= \sum_q P(T = \text{pass}, Q = q) \\
&= P(T = \text{pass}|Q = q^+)P(Q = q^+) + P(T = \text{pass}|Q = q^-)P(Q = q^-) \\
&= 0.69 \\
P(T = fail) &= 0.31 \\
P(Q = q^+|T = \text{pass}) &= \frac{P(T = \text{pass}|Q = q^+)P(Q = q^+)}{P(T = \text{pass})} \\
&= \frac{0.9 \cdot 0.7}{0.69} = \frac{21}{23} \approx 0.91 \\
P(Q = q^+|T = \text{fail}) &= \frac{P(T = \text{fail}|Q = q^+)P(Q = q^+)}{P(T = \text{fail})} \\
&= \frac{0.1 \cdot 0.7}{0.31} = \frac{7}{31} \approx 0.22
\end{aligned}$$

4. Calculate the optimal decisions given either a pass or a fail, and their expected utilities.

$$\begin{aligned}
EU(\text{buy}|T = pass) &= P(Q = q^+|T = \text{pass})U(q^+, \text{buy}) + P(Q = q^-|T = \text{pass})U(q^-, \text{buy}) \\
&\approx 0.91 \cdot 500 + 0.09 \cdot (-200) \approx 437 \\
EU(\text{buy}|T = \text{fail}) &= P(Q = q^+|T = \text{fail})U(q^+, \text{buy}) + P(Q = q^-|T = \text{fail})U(q^-, \text{buy}) \\
&\approx 0.22 \cdot 500 + 0.78 \cdot (-200) = -46 \\
EU(\neg\text{buy}|T = pass) &= 0 \\
EU(\neg\text{buy}|T = fail) &= 0
\end{aligned}$$

Therefore: $MEU(T = \text{pass}) = 437$ (with buy) and $MEU(T = \text{fail}) = 0$ (using $\neg$buy)

5. Calculate the value of (perfect) information of the test. Should the buyer pay for a test?

$$\begin{aligned}
VPI(T) &= (\sum_t P(T = t)MEU(T = t)) - MEU(\phi) \\
&= 0.69 \cdot 437 + 0.31 \cdot 0 - 290 \approx 11.53
\end{aligned}$$

You shouldn't pay for it, since the cost is \$50.

6. The value of the information in this problem depends greatly on the prior probability $P(Q = q^+)$. What do you think happens to the VPI as you vary $P(Q = q^+)$? What happens when $P(Q = q^+)$ approaches 1? Approaches 0? Approaches 0.5?

Intuitively, the more unsure you are of something, the more value you stand to gain by getting "perfect information" related to it. Thus, we expect that near 1 and near 0, there is little value in the information, but somewhere in the middle there is higher value. Utilities mess with this intuition: if you have an action that is low risk (not buying) and one that is high risk (buying), the midpoint might shift. Thus, as you might see in the next part, the critical range where getting the test is useful lies somewhere between 0.18 and 0.52.

7. If you still have time, try calculating some VPI's for different values of $P(Q = q^+)$ from the previous part. (Using Python or Excel is a good idea here!) Where is/are the break-even point(s)? Where is the maximum?

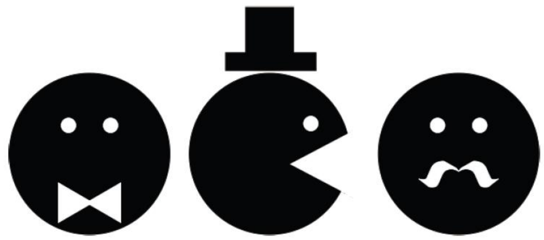
The break-even points are somewhere around 0.52 and 0.18. The maximum is around 0.3.

CS188 Spring 2014 Section 10: ML

Pacman and Mrs. Pacman have been searching for each other in the Maze. Mrs. Pacman has been pregnant with a baby, and just this morning she has given birth to Pacbaby (Congratulations, Pacmans!).

Because Pacbaby was born before Pacman and Mrs. Pacman reunited in the maze, he has never met his father. Naturally, Mrs. Pacman wants to teach Pacbaby to recognize his father, using a set of Polaroids of Pacman. She also has several pictures of ghosts to use as negative examples.

Because the polaroids are black and white, and were taken from strange angles, Mrs. Pacman has decided to teach Pacbaby to identify Pacman based on more salient features: the presence of a bowtie (b), hat (h), or mustache (m).



The following table summarizes the content of the Polaroids. Each binary feature is represented as 1 (meaning the feature is present) or 0 (meaning it is absent). The subject y of the photo is encoded as $+1$ for Pacman or -1 for ghost.

(m)	(b)	(h)	Subject (y)
0	0	0	$+1$
1	0	0	$+1$
1	1	0	$+1$
0	1	1	$+1$
1	0	1	-1
1	1	1	-1

1 Naive Pacbaby

Suppose Pacbaby has a Naive Bayes based brain.

a) Write the Naive Bayes classification rule for this problem (i.e. write a formula which given a data point $x = (m, b, h)$ returns the most likely subject y). Write the formula in terms of conditional and prior probabilities. Be explicit about which parameters are involved, but you do not need to estimate them yet.

b) Assuming no smoothing, give estimates for the parameters of the classification rule based on the Polaroids.

	$y = +1$	$y = -1$
$P(y)$		

P	$y = +1$	$y = -1$
$P(m = 1\|y)$		
$P(b = 1\|y)$		
$P(h = 1\|y)$		

c) Suppose a character comes by wearing a hat but without a mustache or bowtie. What would happen if Pacbaby had to guess the identity of the character?

d) Suppose now that Pacbaby performs Laplace smoothing with strength $k = 1$ (on both the prior and class-conditional parameters). Re-estimate the parameters. Now how will Pacbaby classify this new character with the hat and without a mustache or bowtie?

	$y = +1$	$y = -1$
$P(y)$		

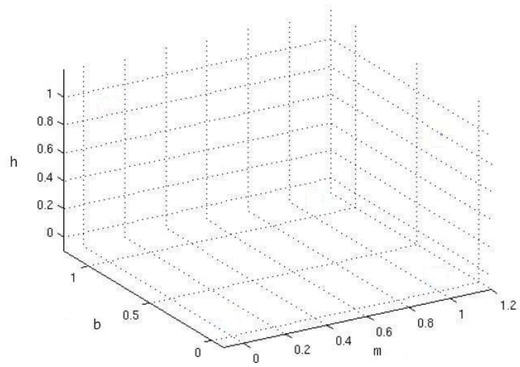
P	$y = +1$	$y = -1$
$P(m = 1\|y)$		
$P(b = 1\|y)$		
$P(h = 1\|y)$		

2 Perceptron Pacbaby

Suppose Pacbaby has a perceptron based brain, meaning his "internal classifier" learns through perceptron updates and is limited to learning linear classification rules. Further, suppose Pacbaby "augments" each example it sees with a bias feature that is always equal to 1 (this allows Pacbaby to learn decision rules with boundaries that do not pass through the origin).

(m)	(b)	(h)	Subject (y)
0	0	0	$+1$
1	0	0	$+1$
1	1	0	$+1$
0	1	1	$+1$
1	0	1	-1
1	1	1	-1

a) Will Pacbaby be able to learn a rule that makes no mistakes on the set of Polaroids? In other words, is the training set linearly separable? (Plot the training data)



b) Suppose there was another Polaroid of a character without a mustache or a hat, but who was wearing a bowtie. If this Polaroid was of Pacman, would the data be linearly separable? What if it contained a ghost?

c) Suppose we start with the training weights $[-1, 1, -1, -1]$, and wish to train a perceptron on the above data. Perform two updates of the Perceptron algorithm, processing the training data in the order they appear. The first 3 weights correspond to the features (m),(b),(h), respectively. The last weight corresponds to the bias feature. If a training example has weight exactly 0, classify it as +1.

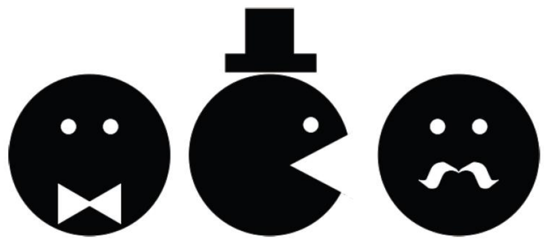
	w_1	w_2	w_3	w_4
Initial weights	-1	1	-1	-1
Training: (0,0,0,1)$\rightarrow$ +1				
Training: (1,0,0,1)$\rightarrow$ +1				

CS188 Spring 2014 Section 10: ML

Pacman and Mrs. Pacman have been searching for each other in the Maze. Mrs. Pacman has been pregnant with a baby, and just this morning she has given birth to Pacbaby (Congratulations, Pacmans!).

Because Pacbaby was born before Pacman and Mrs. Pacman reunited in the maze, he has never met his father. Naturally, Mrs. Pacman wants to teach Pacbaby to recognize his father, using a set of Polaroids of Pacman. She also has several pictures of ghosts to use as negative examples.

Because the polaroids are black and white, and were taken from strange angles, Mrs. Pacman has decided to teach Pacbaby to identify Pacman based on more salient features: the presence of a bowtie (b), hat (h), or mustache (m).



The following table summarizes the content of the Polaroids. Each binary feature is represented as 1 (meaning the feature is present) or 0 (meaning it is absent). The subject y of the photo is encoded as $+1$ for Pacman or -1 for ghost.

(m)	(b)	(h)	Subject (y)
0	0	0	$+1$
1	0	0	$+1$
1	1	0	$+1$
0	1	1	$+1$
1	0	1	-1
1	1	1	-1

1 Naive Pacbaby

Suppose Pacbaby has a Naive Bayes based brain.

a) Write the Naive Bayes classification rule for this problem (i.e. write a formula which given a data point $x = (m, b, h)$ returns the most likely subject y). Write the formula in terms of conditional and prior probabilities. Be explicit about which parameters are involved, but you do not need to estimate them yet.

$$y^* = \arg\max_y \{p(y)p(m|y)p(b|y)p(h|y)\}$$

The parameters are the prior / conditional probabilities of seeing the feature given the class.

b) Assuming no smoothing, give estimates for the parameters of the classification rule based on the Polaroids.

	$y = +1$	$y = -1$
$P(y)$	$\frac{4}{6}$	$\frac{2}{6}$

P	$y = +1$	$y = -1$
$P(m = 1\|y)$	$\frac{2}{4}$	$\frac{2}{2}$
$P(b = 1\|y)$	$\frac{2}{4}$	$\frac{1}{2}$
$P(h = 1\|y)$	$\frac{1}{4}$	$\frac{2}{2}$

c) Suppose a character comes by wearing a hat but without a mustache or bowtie. What would happen if Pacbaby had to guess the identity of the character?

If $y = 1$, the probability of seeing this data point $x = (0, 0, 1)$ is

$$P(y = 1)P(m = 0|y = 1)P(b = 0|y = 1)P(h = 1|y = 1) = (4/6)(2/4)(2/4)(1/4) = 1/24$$

If $y = -1$, the probability $P(m = 0|y = -1) = 0$, so the probability of seeing this is 0.

Pacbaby would classify this as Pacman.

d) Suppose now that Pacbaby performs Laplace smoothing with strength $k = 1$ (on both the prior and class-conditional parameters). Re-estimate the parameters. Now how will Pacbaby classify this new character with the hat and without a mustache or bowtie?

	$y = +1$	$y = -1$
$P(y)$	$\frac{5}{8}$	$\frac{3}{8}$

P	$y = +1$	$y = -1$
$P(m = 1\|y)$	$\frac{3}{6}$	$\frac{3}{4}$
$P(b = 1\|y)$	$\frac{3}{6}$	$\frac{2}{4}$
$P(h = 1\|y)$	$\frac{2}{6}$	$\frac{3}{4}$

If $y = 1$, the probability of seeing this data point $x = (0, 0, 1)$ is

$$P(y = 1)P(m = 0|y = 1)P(b = 0|y = 1)P(h = 1|y = 1) = (5/8)(3/6)(3/6)(2/6) = 5/96$$

If $y = -1$, the probability is

$$P(y = -1)P(m = 0|y = -1)P(b = 0|y = -1)P(h = 1|y = -1) = (3/8)(1/4)(2/4)(3/4) = 9/256$$

Pacbaby would still classify this as Pacman.

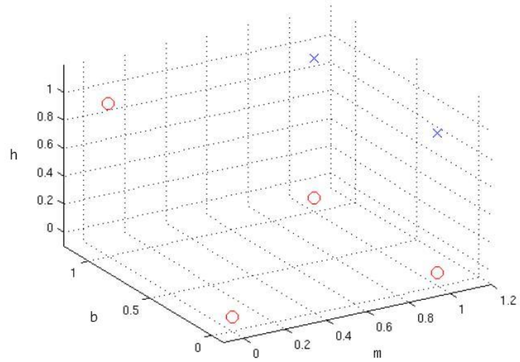
2 Perceptron Pacbaby

Suppose Pacbaby has a perceptron based brain, meaning his "internal classifier" learns through perceptron updates and is limited to learning linear classification rules. Further, suppose Pacbaby "augments" each example it sees with a bias feature that is always equal to 1 (this allows Pacbaby to learn decision rules with boundaries that do not pass through the origin).

(m)	(b)	(h)	Subject (y)
0	0	0	$+1$
1	0	0	$+1$
1	1	0	$+1$
0	1	1	$+1$
1	0	1	-1
1	1	1	-1

a) Will Pacbaby be able to learn a rule that makes no mistakes on the set of Polaroids? In other words, is the training set linearly separable? (Plot the training data)

Yes, as clear in the above graph, the training set is linearly separable. Above, the blue Xs represent the ghost (-1) examples, and the red circles represent the Pacman $(+1)$ examples.



b) Suppose there was another Polaroid of a character without a mustache or a hat, but who was wearing a bowtie. If this Polaroid was of Pacman, would the data be linearly separable? What if it contained a ghost?

The data would be linearly separable if the Polaroid was of Pacman, but NOT if the Polaroid was of a ghost. If the Polaroid was of a ghost, consider the 2d plane formed by the vertices (0,1,0), (0,1,1), (1,0,1), (1,0,0). In this plane, two opposite corners of the rectangle are +1 and the other opposite corners are −1; it is not possible to separate these 2 classes in this 2d plane.

c) Suppose we start with the training weights $[-1, 1, -1, -1]$, and wish to train a perceptron on the above data. Perform two updates of the Perceptron algorithm, processing the training data in the order they appear. The first 3 weights correspond to the features (m),(b),(h), respectively. The last weight corresponds to the bias feature. If a training example has weight exactly 0, classify it as +1.

	w_1	w_2	w_3	w_4
Initial weights	-1	1	-1	-1
Training: (0,0,0,1)$\to$ +1	-1	1	-1	0
Training: (1,0,0,1)$\to$ +1	0	1	-1	1

The perceptron makes a mistake on the first training example: $w^\top x = -1 < 0 \to -1$, so we add the first training example to get $[-1, 1, -1, 0]$. On the second training example it makes another mistake $w^\top x = -1 < 0 \to -1$, so we add the second training example to get weights $[0, 1, -1, 1]$.

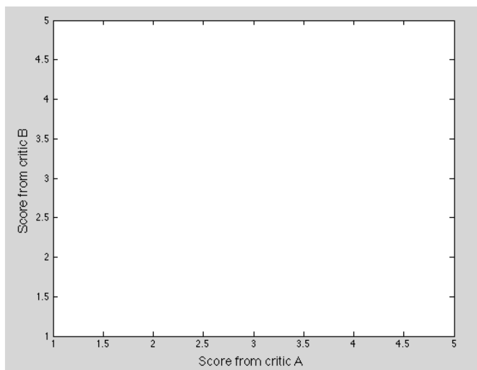
CS188 Spring 2014 Section 11: Machine Learning

You want to predict if movies will be profitable based on their screenplays. You hire two critics A and B to read a script you have and rate it on a scale of 1 to 5. The critics are not perfect; here are five data points including the critics' scores and the performance of the movie:

Movie Name	A	B	Profit?
Pellet Power	1	1	No
Ghosts!	3	2	Yes
Pac is Bac	4	5	No
Not a Pizza	3	4	Yes
Endless Maze	2	3	Yes

Training Data

First, you would like to examine the linear separability of the data. Plot the data on the 2D plane below; label profitable movies with + and non-profitable movies with − and determine if the data are linearly separable.



Now you first decide to use a perceptron to classify your data. This problem will use the multi-class formulation even though there are only two classes. Suppose you directly use the scores given above as features, together with a bias feature. That is $f_0 = 1$, $f_1 =$ score given by A and $f_2 =$ score given by B.

1. You want to train the perceptron on the training data in Table 1. The initial weights are given below:

Profit	Weights	Weights after 1st update
Yes	[-1, 0, 0]	
No	[1, 0, 0]	

   (i) Which is the first training instance at which you update your weights?

   (ii) In the table above, write the updated weights after the first update.

2. More generally, irrespective of the training data, you want to know if your features are powerful enough to allow you to handle a range of scenarios. Some scenarios are given on the next page. Circle those scenarios for which a perceptron using the features above can indeed perfectly classify the data.

(i) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be a success and otherwise it will fail.

(ii) Your reviewers are art critics. Your movie will succeed if and only if each reviewer gives either a score of 2 or a score of 3.

(iii) Your reviewers have weird but different tastes. Your movie will succeed if and only if both reviewers agree.

You decide to use a different set of features. Consider the following feature space:

$$\begin{aligned}
f_0 &= 1 \text{ (The bias feature)} \\
f_{1A} &= 1 \text{ if score given by A is 1, 0 otherwise} \\
f_{1B} &= 1 \text{ if score given by B is 1, 0 otherwise} \\
f_{2A} &= 1 \text{ if score given by A is 2, 0 otherwise} \\
f_{2B} &= 1 \text{ if score given by B is 2, 0 otherwise} \\
&\dots \\
f_{5B} &= 1 \text{ if score given by B is 5, 0 otherwise}
\end{aligned}$$

3. Consider again the three scenarios in part 2. Using a perceptron with the new features, which of the three scenarios can be perfectly classified? Circle your answer(s) below:

   (i) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be a success, and otherwise it will fail.

   (ii) Your reviewers are art critics. Your movie will succeed if and only if each reviewer gives either a score of 2 or a score of 3.

   (iii) Your reviewers have weird but different tastes. Your movie will succeed if and only if both reviewers agree.

You have just heard of naive Bayes and you want to use a naive Bayes classifier. You use the scores given by the reviewers as the features of the naive Bayes classifier, i.e., the random variables in your naive Bayes model are A and B, each with a domain of $\{1, 2, \dots, 5\}$, and *Profit* with a domain of *Yes* and *No*.

4. Draw the Bayes net corresponding to the naive Bayes model on the back of this page.

5. List the types of the conditional probability tables you need to estimate along with their sizes (e.g., $P(X \mid Y)$ has 24 entries).

Probability	Size

6. Your nephew is taking the CS188 class at Berkeley. He claims that the naive Bayes classifier you just built is actually a linear classifier in the feature space used for part 3. In other words, the decision boundary of the naive Bayes classifier is a hyperplane in this feature space. For the positive class, what is the weight of the feature f_{3B} in terms of the parameters of the naive Bayes model? You can answer in symbols, but be precise. (Hint: Consider the log of the probability.)

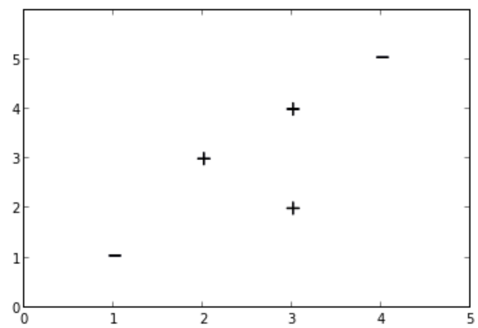
CS188 Spring 2014 Section 11: Machine Learning

You want to predict if movies will be profitable based on their screenplays. You hire two critics A and B to read a script you have and rate it on a scale of 1 to 5. The critics are not perfect; here are five data points including the critics' scores and the performance of the movie:

Movie Name	A	B	Profit?
Pellet Power	1	1	No
Ghosts!	3	2	Yes
Pac is Bac	4	5	No
Not a Pizza	3	4	Yes
Endless Maze	2	3	Yes

Training Data

First, you would like to examine the linear separability of the data. Plot the data on the 2D plane below; label profitable movies with + and non-profitable movies with − and determine if the data are linearly separable.



The data are not linearly separable!

Now you first decide to use a perceptron to classify your data. This problem will use the multi-class formulation even though there are only two classes. Suppose you directly use the scores given above as features, together with a bias feature. That is $f_0 = 1$, $f_1 =$ score given by A and $f_2 =$ score given by B.

1. You want to train the perceptron on the training data in Table ??. The initial weights are given below:

Profit	Weights	Weights after 1st update
Yes	[-1, 0, 0]	[0, 3, 2]
No	[1, 0, 0]	[0, -3, -2]

   (i) Which is the first training instance at which you update your weights? "Ghosts!", because the perceptron predicts "No" for profit, whereas the label is "Yes."

   (ii) In the table above, write the updated weights after the first update. Note that in the binary case, the weights for two classes are exactly opposites, which is why we usually simplify and just work with a single weight vector.

2. More generally, irrespective of the training data, you want to know if your features are powerful enough to allow you to handle a range of scenarios. Some scenarios are given on the next page. Circle those scenarios for which a perceptron using the features above can indeed perfectly classify the data.

   (i) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be a success and otherwise it will fail. Can classify (consider weights $[-8, 1, 1]$)

   (ii) Your reviewers are art critics. Your movie will succeed if and only if each reviewer gives either a score of 2 or a score of 3. Cannot classify

   (iii) Your reviewers have weird but different tastes. Your movie will succeed if and only if both reviewers agree. Cannot classify

You decide to use a different set of features. Consider the following feature space:

$$
\begin{aligned}
f_0 &= 1 \text{ (The bias feature)} \\
f_{1A} &= 1 \text{ if score given by A is 1, 0 otherwise} \\
f_{1B} &= 1 \text{ if score given by B is 1, 0 otherwise} \\
f_{2A} &= 1 \text{ if score given by A is 2, 0 otherwise} \\
f_{2B} &= 1 \text{ if score given by B is 2, 0 otherwise} \\
&\dots \\
f_{5B} &= 1 \text{ if score given by B is 5, 0 otherwise}
\end{aligned}
$$

3. Consider again the three scenarios in part 2. Using a perceptron with the new features, which of the three scenarios can be perfectly classified? Circle your answer(s) below:

   (i) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be a success, and otherwise it will fail. Can classify (consider weights $[-8, 1, 1, 2, 2, \dots, 5, 5]$)

   (ii) Your reviewers are art critics. Your movie will succeed if and only if each reviewer gives either a score of 2 or a score of 3. Can classify (consider weights $[1, -\infty, -\infty, 0, 0, 0, 0, -\infty, -\infty, -\infty, \infty]$, or $[-1.5, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0]$)

   (iii) Your reviewers have weird but different tastes. Your movie will succeed if and only if both reviewers agree. Cannot classify

You have just heard of naive Bayes and you want to use a naive Bayes classifier. You use the scores given by the reviewers as the features of the naive Bayes classifier, i.e., the random variables in your naive Bayes model are A and B, each with a domain of $\{1, 2, \dots, 5\}$, and *Profit* with a domain of *Yes* and *No*.

4. Draw the Bayes net corresponding to the naive Bayes model on the back of this page.

5. List the types of the conditional probability tables you need to estimate along with their sizes (e.g., $P(X \mid Y)$ has 24 entries).

Probability	Size
$P(Profit)$	2
$P(A\|Profit)$	10
$P(B\|Profit)$	10

6. Your nephew is taking the CS188 class at Berkeley. He claims that the naive Bayes classifier you just built is actually a linear classifier in the feature space used for part 3. In other words, the decision boundary of the naive Bayes classifier is a hyperplane in this feature space. For the positive class, what is the weight of the feature f_{3B} in terms of the parameters of the naive Bayes model? You can answer in symbols, but be precise. (Hint: Consider the log of the probability.)

   The weight is $\log P(B = 3|Profit = Yes)$.

   Consider the following weight vectors:

$$w_{Yes} = [\log P(Profit = Yes), \log P(A = 1|Profit = Yes), \ldots, \log P(B = 5|Profit = Yes)]$$

$$w_{No} = [\log P(Profit = No), \log P(A = 1|Profit = No), \ldots, \log P(B = 5|Profit = No)].$$

   Using these weight vectors, the linear classification rule

$$y = \arg\max_y w_y \cdot f(x)$$

   is equivalent to the Naive Bayes rule

$$y = \arg\max_y P(Profit = y)P(A = f_A(x)|Profit = y)P(B = f_B(x)|Profit = y).$$

   This is because the weights are log probabilities, so summing the weights is equivalent to multiplying probabilities, and the 0/1 features pick out which entries of the conditional probability table to include in the product.