Joel Kerfoot
V00855134

1 Agents(1 point)

In the first assignment we implemented two blind dog agents: one that followed a reflex architectue and one that had a model-based architecture. Provide a your own examples of a reflex agent and a model-based agent. The two agents should share the same environment and be different than any of the agent examples in the textbook (and of course not be blind dogs). Describe each of your two agents using PEAS (Performance, Environment, Actuators and Sensors). In addition characterize their shared task environment in terms of observability, number of agents, determinism, whether it is episodic or not, static or dynamic and discrete or continuous (see Figure 2.6 of the 3rd Edition of the textbook). Finally, select one of these task environment characteristics and describe how you could change the task environment so that it changes value. As an example of what I mean by that for the blind dog we could change the task environment from discrete to continuous but having the location of the agent be represented by two floating point numbers *x* and *y*. **(1 point - minimum)**

The "smart" sprinkler agents will be responsible for watering a garden.

Agent Type	Performance	Environment	Actuators	Sensors
Reflex	Plant Health, Water Waste	Outdoor Gardens, Fields	Valves/Pumps	Clock/Timer
Model	Plant Health, Water Waste	Outdoor Gardens, Fields	Valves/Pumps	Clock/Timer, Rain gauge(s), Soil moisture sensor

In both cases there is just one agent the "smart" sprinkler. The environment for reflex agent can only partially observe the environment becuase the only information it has is where or not it is time to water the garden (turn the sprinklers on). The model based agent can fully observe the environment. It knows if it's the right time to water the garden and whether or not the garden should be watered. If it rained the agent can determine if the garden needs to be watered, partially watered, or not watered at all becuase it knows if the garden got enough water that day. The environment is deterministic, if the agent turns on the sprinklers the garden gets watered. The task environment is episodic becuase the agent's actions aren't affected by the previous actions. The environment is static for the reflex agent because the environment doesn't change while the agent is thinking. The reflex agent environment is discrete, it only opperates at a set intervale while the model based agent has a continuous environment as the rain, moisture data is continuous.

2 Uninformed Search (2 points)

Consider a combination lock with three rotary gears. Each gear has settings for four letter A, B, C, D so that any 3 letter string of these letters such as AAA, ABD, BBC etc can be represented. Given an initial random configuration of letters in order to open the lock you need to perform a number of rotary moves so that the gears display a specific unique lock combination. You can only turn one gear at a time and you can only rotate it by one letter. For example if the second gear is set to B there are two valid moves for that gear (B → C) and (B → A) or if the first gear is set to D there are two valid moves for that rear (D → A) and (D → C).

1. Formulate breaking the lock as a search problem. Be PRECISE and use the terminology of the textbook. What is the branching factor of the resulting search tree ? **(1 point - minimum)**

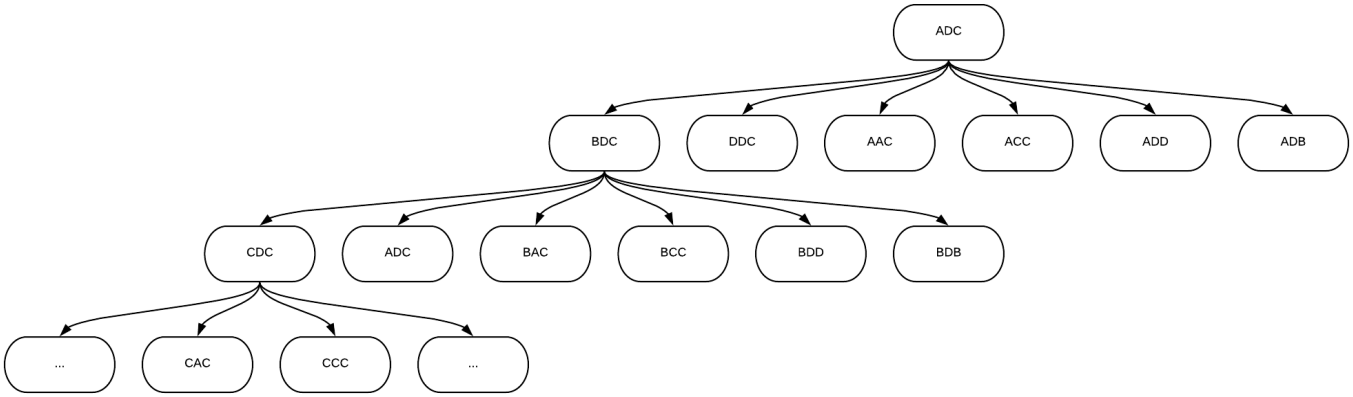
Answer

- *initial state* - the "initial random configuration of letters" eg AAA
- *actions* - rotating one of the three gears either forward (F) or backward (B), the possible actions in any state are {Rotate(1,F), Rotate(1,B), Rotate(2,F), ... , Rotate(3,B)}
- *transition model* - the result of rotating a gear in a given direction eg RESULT(In(AAA), Rotate(1,F)) = In(BAA)
- *goal test* - where or not the combination is the correct combination that opens the lock
- *path cost* - the distance from one state to another is the same regardless of the action taken so in this case the path cost of taking an action in any state can simply be 1 for any single rotation of a gear

For any node in the tree there are 6 possible children, the result of rotating 1 of the 3 gears in 1 of the 2 possible directions, so the branching factor b = 6

1. Suppose that your start combination is A, D, C and the lock combination is B, D, D. Draw a diagram of the corresponding search tree (you don't have to draw all the nodes but draw enough to convey how the tree is constructed). If you don't want to scan this answer or use some kind of drawing program you can write your answer using text with some kind of convention of representing the tree that is clear. Write down the nodes (digits) in order of expansion for Bread-First Search (BFS), Depth-First Search (DFS) and Iterative Deepening (IDS). **(1 point - expected)**

Answer



Breadth-First Search

ADC -> BDC -> DDC -> AAC -> ACC -> ADD -> ADB -> CDC -> ADC -> BAC -> BCC -> BDD

Depth-First Search

If the search tree doesn't terminate because the nodes can be generated infinately (AAA -> BAA -> CAA -> DAA -> AAA -> BAA down the left side of the tree) then DFS will never find the solution in this case. Only in specific cases where the answer is on the right side of the tree will DFS find a solution. A tree of depth 4 should contain all the possible combinations since each gear has 4 possible values. However given the partial tree of depth 3 above the DFS path would be:

ADC -> BDC -> CDC -> CDC's children -> ADC -> ADC's children -> BAC -> BAC's children -> BCC -> BCC's children -> BDD

Iterative Deepening Search

Limit = 0 and 1 will not find the solution and when limit = 2 the path will be:

ADC -> BDC -> CDC -> ADC -> BAC -> BCC -> BDD

3 Informed Search and Heuristics (2 points)

Consider a similar combination lock setup as question 1 but with the following difference: the user needs to enter a specific sequence of 3-digit unlocking configurations before entering the final configuration. A button is used to indicate that an intermediate configuration has been reached. For example the user rotates the gears to reach the first 3-letter combination, then presses the button, then rotate the gears to enter the second 3-letter combination, etc until the final. A solution consists of a sequence of 3-letter combinations where the first item is the initial configuration (S0), then there are some intermediate configurations (S1, S2, ..., SN−1), and the final configuration (SN) is the last item. Fortunately for each intermediate step there is a set of 4 candidate configurations that can be tried (provided by a spy). This candidate set is different for each step of ther process. You can think of the initial configuration as the initial state, and the final configuration as the goal state, and the intermediate configurations as the search states. In analogy to the Romania city problem, the initial configuration corresponds to the start city, the candidate intermediate configurations are the neighbors of a particular step, and the final configuration corresponds to the destination city.

The cost of a solution is the total number of gear moves required to unlock (going through the all intermediate configurations). Note that the gears can move in both directions. For example the cost of going from A, B, D to B, D, C is 1 + 2 + 1 = 4 using the following gear moves: (A, B, D) → (B, B, D), B, B, D → (B, D, D), B, D, D → (B, D, C). Note that the cost of going directly from the inital state to the final configuration is NOT the same as the cost of going through the intermediate configurations.

1. Define an admissable heuristic for this version of the combination lock and explain why it is admissable. **(1 point - expected)**

Answer

An admissable heuristic would be the number of gear moves needed to unlock the lock from the current configuration without considering further intermediate configurations. For example if the curent cofiguration is AAA and the goal configuration is BCD the huristic would be 1 + 2 + 1 = 4. It is admissable because it will never over estimate the true cost of opening the lock since possible intermediate steps aren't considered. The intermediate sets will add gear rotations and therefore increase the true cost of opening the lock. In an extreme case the heurisitc could be the same as the true cost if all the intermediate steps are the same as the final solution. This is still not an overestimate and therefore an admissable heuristic.

1. Suppose that in your informed search you have reached the first step configuration (B, C, A) after the inital configuration (A, D, C). The step 2 candidate set is (A, A, B),(B, C, C),(D, D, D),(D, B, A) and the final state is (B, D, D). Which will be the next configuration after one gear move if you are doing greedy search ? What is the corresponding value of the heuristic function f(n) for greedy search for that next configuration ? Which will be the next configuration after one move if you are doing A *search ? What will be the corresponding value of the heuristic function f(n) for that next configuation for A* search ? Be precise in your answers. **(1 point - advanced)**

Answer

AAB -> BDD = 1 + 1 + 2 = 4 BCC -> BDD = 0 + 1 + 1 = 2 DDD -> BDD = 2 + 0 + 0 = 2 DBA -> BDD = 2 + 2 + 1 = 5

Greedy Search

BCC and DDD both have the same heuristic of 2. Greedy search selects the node closest to the goal f(n) = h(n) so we'll select BCC. To get to BCC from BCA the next move is BCB (or BCD) and the heuristic from BCB to BCC is 1.

A* Search

f(n) = h(n) + g(n)

BCA -> AAB = 1 + 2 + 1 = 4 -> 4 + h(n) = 4 + 4 = 8
BCA -> BCC = 0 + 0 + 2 = 2 -> 2 + h(n) = 2 + 2 = 4
BCA -> DDD = 2 + 1 + 1 = 4 -> 4 + h(n) = 4 + 2 = 6
BCA -> DBA = 2 + 1 + 0 = 3 -> 3 + h(n) = 3 + 5 = 8

A* will select BCC. To get to BCC from BCA the next move is BCB (or BCD) and the heuristic from BCB to BCC is 1.

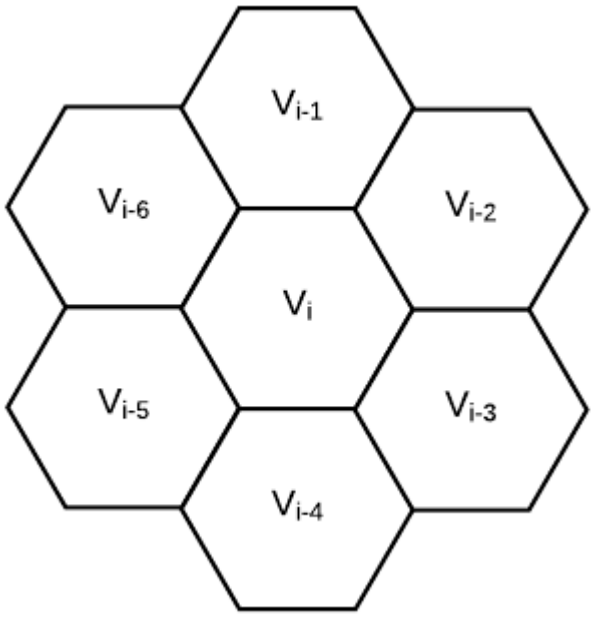
4 CSP (1 point)

Consider a hexagonal grid (such as the one used in the popular Civilization series of compuer games or the Settlers of Catan board game). The grid can be represented as a graph with vertices Vi and edges between neighboring nodes. Most vertices Vi that are not at the edge of the world (or board) have 6 neighbors. Each grid location is assigned a land type: forest, ocean, desert, mountain, or grass. A map is created by setting a number of vertices to specific land types and then assigning land types to the remaining vertices given a number of specific constraints such as that all neighbors of a particular vertex must have different land types.

1. Using the terminology of the book formulate this problem as a CSP problem. Be precise about what are the variables, domains and roughly sketch how constraints would be expressed. Describe what the solution of the CSP would look like with a specific example. **(1 point - expected)**

X = { V_1, V_2, ... , V_n }
The domain of each variable is the set D_i = { forest, ocean, desert, mountain, or grass }

Assume the following naming convention for addressing adject nodes to a node V_i



Given the example constraint: "all neighbors of a particular vertex must have different land types" we can create the following constraint:
C = { $V_i \neq V_{i-1}, V_i \neq V_{i-2}, V_i \neq V_{i-3}, V_i \neq V_{i-4}, V_i \neq V_{i-5}, V_i \neq V_{i-6}$ }

5 Propositional Logic (1 point)

Assume that we have the following propositions: EasyFinal, LearnedAI, NeverReadBook, MissedLectures. (you can use the abbreviations E,L,N,M in your answer). **(1 point - minimum)**

1. What is the number of possible models ?

4 propositions means there are 2^4 = 16 possible models

1. How many models are there in which the following sentence is false ? EasyFinal∧LearnedAI → (¬NeverReadBook∧¬MissedLectures)

An implication is false when T -> F so

EasyFinal∧LearnedAI == T which is only true when EasyFinal and LearnedAI are both true and

¬NeverReadBook∧¬MissedLectures == F is false when NeverReadBook, MissedLectures, or both are true

There are **3** models in which the sentence is false: (E,L,N,M), (E,L,¬N,M), and (E,L,N,¬M)

1. Prove using model checking that the above sentence is NOT entailed by the sentence Easymidterm → ¬NeverReadBook.

lets take (E,L,¬N,M) for example

EasyFinal∧LearnedAI → (¬NeverReadBook∧¬MissedLectures) becomes:
T∧T → (¬F∧¬T) == T -> F == F

Easymidterm → ¬NeverReadBook becomes:
T -> ¬F == T

Clearly the sentance Easymidterm → ¬NeverReadBook is not true in every case EasyFinal∧LearnedAI → (¬NeverReadBook∧¬MissedLectures) is true. Therefore it is not entailed.

6 First order logic (1 point)

Translate each of the following English sentences into the language of standard first order logic stating the intended interpretation for any predicate function or constant you use. Using a finite world assumption i.e variables can only take a finite number of values show how these sentences can be translated to propositional logic. Use the following tools for values: philips screw driver, square screw driver, wrench, adjustable wrench. **(1 point - advanced)**

1. All screwdrivers are tools

∀ x Screwdriver(x) => Tool(x)

1. If someone owns a screwdriver, then there is some tool they own.

7 Probabilities (2 points)

Consider the following two random variables: X taking values AI and Data Mining and Y taking values Computer Science and Software Engineering. There are 80 computer science students and 40 software engineering students in AI. There are 60 software engineering students and 30 computer science students in Data Mining.

1. Write the full joint probability distribution for these two random variables with probabilities estimated from the data provided. **(1 point - minimum)**

	X=AI	**X=DM**
Y=SENG	**0.19**	**0.29**
Y=CSC	**0.38**	**0.14**

2. Calculate the probabilities P(X = AI), P(X = AI, Y = SENG), and P(X = AI|Y = SENG). Show using specific probabilities from this example that the Bayes theorem holds for X = AI and Y = SENG. **(1 point - minimum)**

$P(X = AI) = 0.38 + 0.19 = 0.57$

$P(X = AI, Y = SENG) = 0.19$

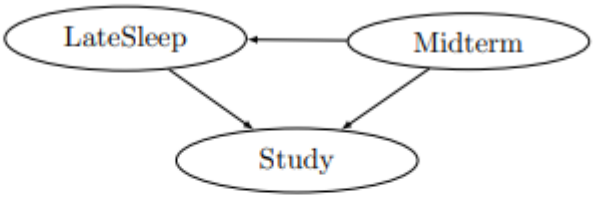
$P(X = AI|Y = SENG) = \frac{P(X=AI,Y=SENG)}{P(Y=SENG)} = \frac{0.19}{0.19+0.29} = 0.396$

$P(X = AI|Y = SENG) = \frac{P(Y=SENG|X=AI)*P(X=AI)}{P(Y=SENG)} = \frac{\frac{1}{3}*0.57}{0.19+0.29} = 0.396$

8 Bayesian Networks (3 points)

Consider the following scenario. The probability that you sleep late and that you study is influenced by whether you have a midterm or not the next day. Frequently the nights that you sleep late you do it so that you can study (for example for courses in addition to midterms). The following Bayesian network can be used to represent this scenario.

Midterm	LateSleep T	LateSleep F
F	0.4	0.6
T	0.1	0.9



Midterm T	Midterm F
0.2	0.8

LateSleep	Midterm	Study T	Study F
F	F	0.4	0.6
F	T	0.1	0.9
T	F	0.2	0.8
T	T	0.3	0.7

Use the convention that s means Study=True (similarly for m and l). The answers to the quesions below are somewhat tedious to write but straightforward.

1. Show how P(s|m) can be expressed using the CPT above using exact inference by enumeration. **(1 point - minimum)**

$P(s|m) = \alpha \sum_l P(s, m, l) = \alpha(P(s, m, l) + P(s, m, \neg l)) =$

$\alpha * (0.2 * 0.1 * 0.3 + 0.2 * 0.9 * 0.1) = \alpha * 0.024$

$P(\neg s|m) = \alpha \sum_l P(\neg s, m, l) = \alpha(P(\neg s, m, l) + P(\neg s, m, \neg l)) =$

$\alpha * (0.2 * 0.1 * 0.7 + 0.2 * 0.9 * 0.9) = \alpha * 0.176$

$P(s|m) = \frac{0.024}{0.024+0.176} = 0.12$

$P(\neg s|m) = \frac{0.176}{0.024+0.176} = 0.88$

1. Calculate the full joint probability expressed by this Bayesian network and calculate P(s|m) using the full joint **(1 point - expected)**

m	l	s	P
T	T	T	0.006
T	T	F	0.014
T	F	T	0.018
F	T	T	0.064

m	l	s	P
T	F	F	0.162
F	T	F	0.256
F	F	T	0.192
F	F	F	0.288
			1.000

1. Do two round of approximate inference using direct sampling. Use a random sequence of numbers between 0.0 and 1.0 - you can calculate it using Python to perform the sampling. You need to show these numbers and how you use them with the CPTs. Show the two generated events and explain in your own words how rejection sampling could be used to calculate P(s|m). **(1 point - advanced)**

```
from random import random
r1 = [random() for i in range(3)]
# [0.6403168746817319, 0.8563299554037306, 0.5067558777972552]
```

$0.64 > 0.2- > \neg m$

$0.85 > P(l|\neg m) == 0.85 > 0.4- > \neg l$

$0.50 > P(s|\neg l, \neg m) == 0.5 > 0.4- > \neg s$

result: $(\neg m, \neg l, \neg s)$

```
r2 = [random() for i in range(3)]
# [0.5206704094713512, 0.36543413837216177, 0.9302752683866587]
```

$0.52 > 0.2- > \neg m$

$0.37 < P(l|\neg m) == 0.37 < 0.4- > l$

$0.93 > P(s|\neg l, \neg m) == 0.93 > 0.2- > \neg s$

result: $(\neg m, l, \neg s)$

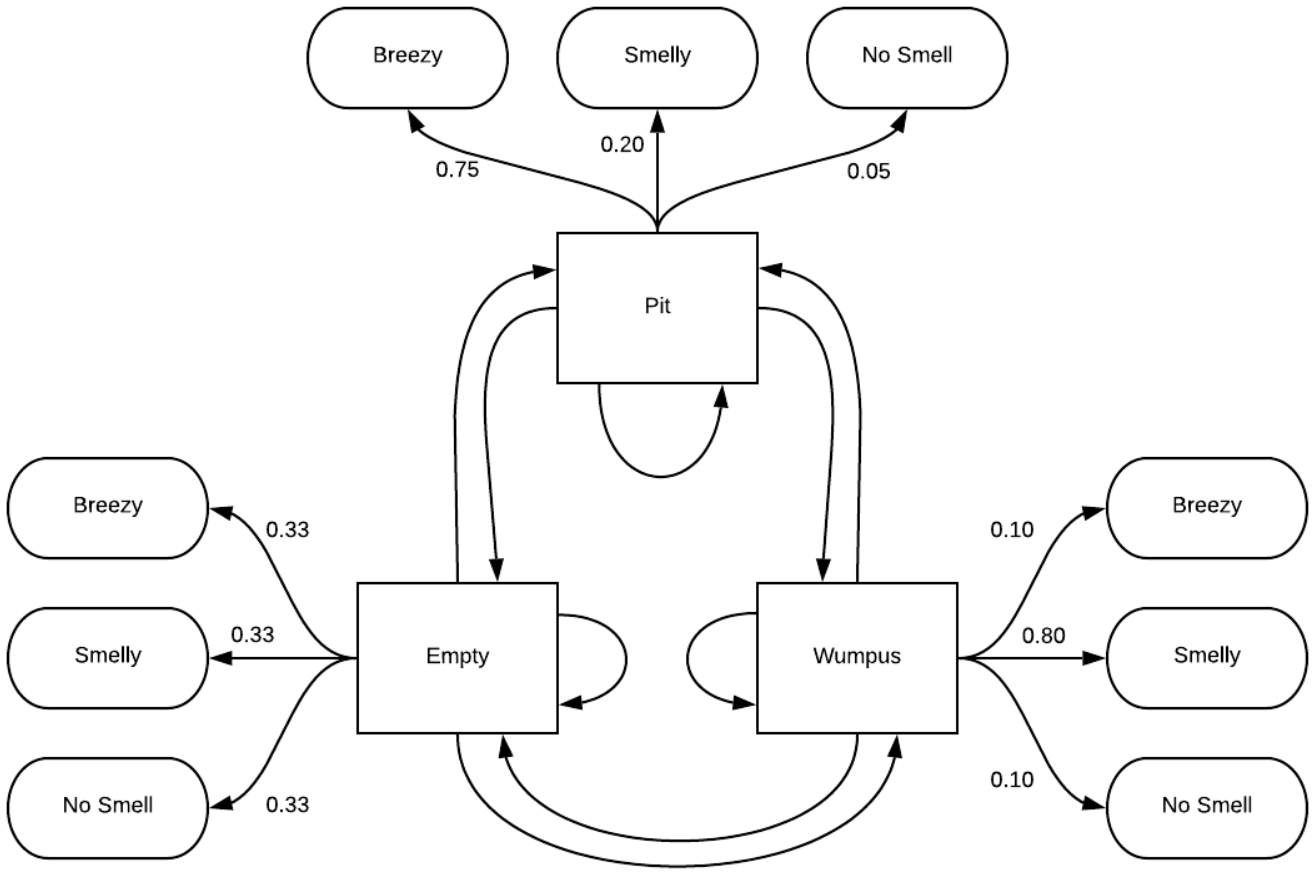
9 Hidden Markov Models (2 points)

Consider the following variation of the Wumpus world. The world is one infinite corridor of squares and your character can only move in one direction (left-to-right). Each square can either be Empty, be a Pit, or contain a Wumpus. You are wearing a special levitation device and a shield so you can go over Pits and not be killed by Wumpuses. Because of your levitation device your nose is not as reliable as it used to be and you can only "sense" the square you are in.

1. If the square is Empty you can sense Breezy, Smelly, NoSmell with equal probability i.e P(Breezy|Empty) = 0.33, P(Smelly|Empty) = 0.33, P(NoSmell|Empty) = 0.33
2. If the square is a Pit then P(Breezy|P it) = 0.75, P(Smelly|P it) = 0.20, P(NoSmell|P it) = 0.05.
3. If the square has a Wumpus then P(Breezy|W umpus) = 0.10, P(Smelly|W umpus) = 0.80), P(NoSmell|W umpus) = 0.10.

Your task will be, given a sequence of sensory observations, to map the corridor (make informed decision about the "state" of each square).

1. Specify this problem formally as a Hidden Markov Model. How many numbers do you need to specify the transition model (assuming a firstorder process)? Write down the sensor model. **(1 point - minimum)**



```
sensor_model = {
   'Empty' : {'Breezy': 0.33, 'Smelly': 0.33, 'NoSmell': 0.33 },
   'Pit' : {'Breezy': 0.75, 'Smelly': 0.20, 'NoSmell': 0.05 },
   'Wumpus' : {'Breezy':  0.10, 'Smelly': 0.80, 'NoSmell': 0.10 }
   }
```

9 numbers would be needed to specify the tranisition model. There are 3 states each could transition to one of the other 2 states or stay the same - 3 states * 3 possibilities per state = 9.

1. Explain what would the task of most likely explanation correspond for this particular problem. Now, assume that all the transition probabilities are equal. Assume that you have observed the following sequence for the first two squares you have visited (S1 = Breezy, S2 = Breezy). Compute the probability of square three (S3) being a Pit, given these observations and the Hidden Markov Model. What is this task called? **(1 point - advanced)**

The task of most likely explanation would be finding the most probable layout of pits, empty squares, or wumpus's in the world.

Pit = 0.75

Not Pit = 0.33 + 0.10 = 0.43

$$\frac{0.75}{0.75+0.43} = 0.64$$

$$\frac{0.43}{0.75+0.43} = 0.36$$

10 Cross-validation (2 points)

Suppose that you are given a binary executable with a method for training a classifier given a labeled set of samples that returns a trained model and a method for predicting the labels of a test set of samples given the trained model. Basically a standard black-box classification API. Your task is to implement cross-validation and to provide some predictions about how long it will take to compute. The computation time for training this particular classifier is $3 * N^2 + N$ where N is the number of samples used for training. The computation time for predicting is $5 * N$ where N is the number of samples being tested.

1. How many times will the train method and predict method be called when doing a 5-fold cross-validation? Explain your answer **(1 point - minimum)**

When doing k-fold cros-validation the train method will be called k times using k-1 folds of the data with the last 1 fold being reserved for testing. Therefore when doing 5-fold cross-validation traing will be called 5 times and predict will be called 5 times (once per fold).

1. Express the total computation cost of k-fold cross-validation as a functio of K and N for this particular classifier. Explain your answer. **(1 point - expected)**

Training computation time: $K * [3(\frac{K-1}{K}N)^2 + \frac{K-1}{K}N]$

Have to train the classifier K times using $\frac{K-1}{K}N$ each time as training data. For example if K=5 then we'd train with 80% of the data 5 times.

Testing computation time: $K(5 * \frac{1}{K}N)$

Again have to test K times using $\frac{1}{K}N$ each time as test data. For example if K=5 then we'd test with 20% of the data 5 times.

Total Computation Time:

$K * [3(\frac{K-1}{K}N)^2 + \frac{K-1}{K}N] + K(5 * \frac{1}{K}N) =$

$K * [3(\frac{K-1}{K}N)^2 + \frac{K-1}{K}N + 5 * \frac{1}{K}N]$

11 Maximum Likelihood Learning and Naive Bayes Classification (3 points)

Suppose you are given a sequence of observations t1, t2, ..., tn. You are also told that these observations were obtained by sampling from an exponential distribution with an unknown parameter λ, that is,

$p(t) = \lambda e^{-\lambda t_i}$

1. Express the likelihood as a function of λ assuming that the observations t1, t2, ..., tn are i.i.d (independently and identically distributed). Express and simplify the log-likelihood function. **(1 point - expected)**

$$L = \sum_{i=1}^{n} \ln(\gamma e^{-\gamma t_i}) = \sum_{i=1}^{n} \ln(\gamma) - \gamma t_i$$

1. Show how the parameter λ can be learned using Maximum-Likelihood parameter learning analytically by setting the derivative of the loglikelihood to zero. Estimate the value of λML for the following set of observations (0.2, 0.3, 0.1, 0.1, 0.2, 0.5). **(1 point - expected)**

$$\frac{\partial}{\partial \gamma} \sum_{i=1}^{n} \ln(\gamma) - \gamma t_i = \sum_{i=1}^{n} \frac{1}{\gamma} - t_i = \frac{n}{\gamma} - \sum_{i=1}^{n} t_i$$

$$\frac{n}{\gamma} - \sum_{i=1}^{n} t_i = 0$$

$$\gamma = \frac{n}{\sum_{i=1}^{n} t_i}$$

Given the following set of observations (0.2, 0.3, 0.1, 0.1, 0.2, 0.5)

$$\gamma_{ML} = \frac{6}{0.2 + 0.3 + 0.1 + 0.1 + 0.2 + 0.5} = \frac{6}{1.4} = 4.29$$

1. Suppose that you are given the following observations for positive samples (0.2, 0.3, 0.1, 0.1, 0.2) and the following for negative samples (0.6, 0.2, 0.1, 0.1, 0.1). Classify a new sample with value 0.5 into positive or negative using statistical learning with Maximum-Likelihood parameter estimation and assuming that the observations were generated by an exponential probability density function. **(1 point - expected)**