COS 302 Answers to the Practice Final

Michael Littman, Corey Campbell, Jamie Feinberg

1 A) In order to solve this problem one must understand that the chart gives the minimax of the state in which the player (who's turn it is) has i numbers left and the other player has j numbers left.

Also the two opponent's minimax value must sum to 1 (b/c one of the two players is going to win). minimax of i state + minimax of j state = 1

On a 6-4 split

60%: opponent's turn, opponent has 9 we have 6
minimax of i + minimax of j = 1
minimax of i = 1 – minimax of j
1-[9,6] = 1-.481 = .519

40%: opponent's turn, opponent has 9 we have 4
minimax of i + minimax of j = 1
minimax of i = 1 – minimax of j
1-[9,4] = 1-.417 = .583

On a 5-5 split

100%: opponent's turn, opponent has 9 we have 5
minimax of i + minimax of j = 1
minimax of i = 1 – minimax of j
1-[9,5] = 1-.467 = .533

Sum the these two scenarios with their probability

On a 6-4 split
.6*.519+.4*.583 = .5446

On a 5-5 split
0.533

B) The value of the [i, j] given the base case in which player i has only one number left and it's his turn player i in this case will win this game every time so the minimax value for p[1,j] where j = {0-9} is 1. The [i,j] entries are based on the idea that one should choose a strategy from guessing randomly to picking the optimal split given the number of choices you have left and the number of choices your opponent.

p[1,j] = 1   The base case

p[i,j] = the max value(randomly guessing, choosing a maximum value split out of all possible splits)

randomly guessing = 1/i

determining the value of the split is the same procedure as problem 1A

choose maximum split = max(prob[i-k section having opponent number] * minimax value of the state it puts you in + prob[k section having opponent number] * minimax value of the state it puts you in

p[i,j] = max(1/i, max_k=1^{i-1} (((i-k)/i)*(1-p[j,i-k]) + (k/i)(1-p[j,k])))

2

A) gimmel = # of gimmel(s) / # of runs = 1/10 = 0.1
   hey = # of hey(s) / # of runs = 0/10 = 0.0
   nun = # of nun(s) / # of runs = 7/10 = 0.7
   shin = # of shin(s) / # of runs = 2/10 = 0.2

B) The likelihood of the validation data using the answer from part A is 0. The reason its likelihood is 0 is because the probability of getting a hey is 0 and so any validation data with a hey, likelihood will also be 0.

C) Laplace smoothing means that we add one in the training to each one of the categories.

Val	Name	Train	Validate
1	gimmel	2	0
2	hey	1	2
3	nun	8	5
4	skin	3	3

gimmel = # of gimmel(s) / (# of runs + smoothing) = 2/14 = 0.1429
hey = # of hey(s) / (# of runs + smoothing) = 1/14 = 0.0714
nun = # of nun(s) / (# of runs + smoothing) = 8/14 = 0.5714
shin = # of shin(s) / (# of runs + smoothing) = 3/14 = 0.2143

D) Likelihood of Validation = $\text{Product}(\text{probability}(\text{name})^{\text{\# of appearances in validation}})$
   Likelihood of Validation = $0.1429^0 * 0.0714^2 * 0.5714^5 * 0.2143^3$
   Likelihood of Validation = 3.05×10^{-6}

Note: This formula computes the probability of the data occurring in precisely the order given. The correct formula is more complex and isn't really needed for this example

3 A)

rft = Probability of right answer on first try
wft = Probability of wrong answer on first try
rs = Probability of right answer after switch
ws = Probability of wrong answer after switch
Pred = Predicted value of square

	Don't switch	Switch
Don't Show	rft * 2 + wft * 0 = Pred 0.3333 * 2 = 0.66667 0.6667	rft * ws * -1 + wft * rs * 2 = Pred 0.333 * 1 * -1 + 0.667 * 0.5 *2 = 0.333
Show	rft * 2 + wft * 0 = Pred 0.3333 * 2 = 0.66667 0.6667	rft * ws * -1 + wft * rs * 2 = Pred 0.333 * 1 * -1 + 0.667 * 1 *2 = 1

	Don't switch	Switch
No show	0.6667	0.333
Show	0.6667	1

B) Monte should choose not to show because if Monte doesn't show he will minimize the contestant winnings because with no show the contestant can only get expected winning of 0.667 or 0.333 with showing the contestant can get expect winning of 0.667 or 1.
Assuming the contestant plays optimally the resulting expected winning for the contestant is 0.6667

There is an alternate matrix in which Monte and the contestant have four choices instead of two. The analysis of that problem is given below.

M = Monte C = contestant

M :
I: always do nothing
II: always show
III: show if C right, otherwise nothing
IV: show if C wrong, otherwise nothing
C:
I: always do nothing
II: always change
III: change if M show, otherwise nothing
IV: change if M nothing, otherwise nothing

Expected values:

M\C	I	II	III	IV
I	2/3	1/3	2/3	1/3
II	2/3	3/3	3/3	2/3
III	2/3	1/3	-1/3	4/3
IV	2/3	3/3	6/3	-1/3

Now, note that M is trying to minimize and that M's II dominates its I strategy. So, we can eliminate II. (This isn't strictly necessary, but it can simply things.)

M\C	I	II	III	IV
I	2/3	1/3	2/3	1/3
III	2/3	1/3	-1/3	4/3
IV	2/3	3/3	6/3	-1/3

For this game, we are lucky and the minimax analysis is pretty simple. If M chooses I, the best C can do is 2/3. So, the value of the game is no larger than 2/3. On the other hand, if C chooses I, C cannot do any worse than 2/3. So, the value of the game is no smaller than 2/3. Put these two together and we find that the value of the game is exactly 2/3 (where both players do nothing).

For a 2x2 game, if the method above doesn't work, write an expression for the payoff for player 1 playing strategy I with probability x and strategy II with 1-x against player 2's strategy I. Do the same for player 2's strategy II. Set these two expressions equal and solve for x. If x lies between 0 and 1, this is the probabilistic strategy that maximizes player 1's payoffs.

#4 Part of Speech Tagging

***NOTE: roman numerals and letters in the table correspond to explanations below

Note: This example assumes that sentences must begin with a DET. If this is not the case, you would need to multiply elements in the first column of boxes by the probability of starting in each state.

	The	**Man**	**Saw**	**Her**	**.**
DET	1	0	0	0	0
NOUN	0	I. = .5035	III. = .001712 IV. = 1.92 x 10^-6	VII = 1.310 x 10^-4 IX = 7.250 x 10^-6	0
POS	0	0	0	XI = 1.712 x 10^-5 XIII = 1.027 x 10^-4	0
VERB	0	II. = .0012	V. = 2.014 x 10^-4 VI. = 3.6 x 10^-7	0	0
EOS	0	0	0	0	A = **1.009 x 10^-4** B = 1.027 x 10^-6

I. = pr(noun | det) * pr(man | noun) = (.95)(.53) = .5035
II. = pr(verb | det) * pr(man | verb) = (.01)(.12) = .0012
III. = pr(saw | noun)*pr(noun | noun)*(I.) = (.02)(.17)(.5035) = .001712
IV. = pr(saw | noun)*pr(noun | verb)*(II.) = (.02)(.08)(.0012) = 1.92 x 10^-6
V. = pr(saw | verb)*pr(verb|noun)*(I.) = (.01)(.04)(.5053) = 2.014 x 10^-4
VI. = pr(saw | verb)*pr(verb | verb)*(II.) = (.01)(.03)(.0012) = 3.6 x 10^-7
VII. = pr(her | noun)*pr(noun | noun)*(III.) = (.45)(.17)(.001712) = 1.310 x 10^-4
VIII.
IX. = pr(her | noun)*pr(noun | verb)*(V.) = (.45)(.08)(2.014 x 10^-4) = 7.250 x 10^-6
X.
XI. = pr(her | pos)*pr(pos | noun)*(III.) = (1)(.01)(.001712) = 1.712 x 10^-5
XII.
XIII. = pr(her | pos)*pr(pos | verb)*(V.) = (1)(.51)(2.014 x 10^-4) = 1.027 x 10^-4
XIV.

A) = pr(. | EOS)*pr(EOS | noun)*(VII.) = (1)(.77)(1.310 x 10^-4) = 1.009 x 10^-4

B) = pr(. | EOS)*pr(EOS | noun)*(XIII.) = (1)(.77)(1.027 x 10^-4) = 1.027 x 10^-6

Line (A.) has the highest probability, so tracing back from A:

A ◊ VII. ◊ III. ◊ I.

This yields the sequence:

DET ◊ NOUN ◊ NOUN ◊ NOUN ◊ EOS

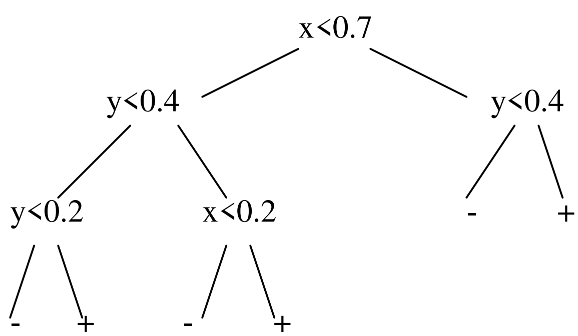
b) You can pick up where you left off and add two more rows.

The most likely sequence of tags is

DET, NOUN, VERB, POS, NOUN, EOS

5 A) This is a sample tree that solves the problem. Any of a number of decision tress will classify this problem correctly as long as they split the graph into regions of all pluses and regions of all minuses.

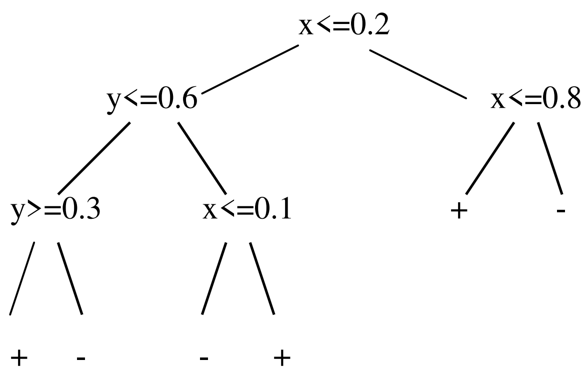
Left = true; Right = false;



B) No, A perceptron cannot correctly classify the points in 5a because a perceptron can only classify points that are linearly separable. The points in 5a are not linearly separable therefore the perceptron cannot classify them.

C) Although solvable, people were having trouble understanding how the original problem worked because y<=0.3 was on the false branch of y<=0.6. This is a solution which has switched y<=0.3 and x<=0.1 leaves of the tree

Left = true; Right = false;



	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9
.0	-	-	-	+	+	+	+	+	+	-
.1	-	-	-	+	+	+	+	+	+	-
.2	-	-	-	+	+	+	+	+	+	-
.3	+	+	+	+	+	+	+	+	+	-
.4	+	+	+	+	+	+	+	+	+	-
.5	+	+	+	+	+	+	+	+	+	-
.6	+	+	+	+	+	+	+	+	+	-
.7	-	-	+	+	+	+	+	+	+	-
.8	-	-	+	+	+	+	+	+	+	-
.9	-	-	+	+	+	+	+	+	+	-

6 A) This is the work of the first row of the chart

Pr(inside | snow sport) = Pr(snow sport | inside) Pr(inside) / Pr(snow sport)
Pr(inside | snow sport) = (0.1 * 0.2 * 0.4) / Pr(snow sport)
Pr(inside | snow sport) = 0.008 / Pr(snow sport)

Pr(outside | snow sport) = Pr(snow sport | outside) Pr(outside) / Pr(snow sport)
Pr(outside | snow sport) = (0.3 * 0.5 * 0.6) / Pr(snow sport)
Pr(outside | snow sport) = 0.090 / Pr(snow sport)

This statement is true because the picture has to be either an outside or inside picture therefore the probability of it being one of the two is 1

Pr(inside | snow sport) + Pr(inside | snow sport) = 1
0.008 / Pr(snow sport) + 0.090 / Pr(snow sport) = 1

We normalize to turn the expression into real numbers

0.098/ Pr(snow sport) = 1
Pr(snow sport) = 0.098

$Pr(inside \mid snow\ sport) = 0.008 / 0.098 = 0.0816$
$Pr(outside \mid snow\ sport) = 0.090 / 0.098 = 0.9184$

	Pr(inside)	Pr(outside)
snow sport	0.0816	0.9184
sport speech	0.5161	0.4839
chair chair	0.8571	0.1429

B)
Pr(inside) = Pr(inside | snow sport) + Pr(inside | sport speech) + Pr(inside | chair chair) / (Pr(inside | snow sport) + Pr(inside | sport speech) + Pr(inside | chair chair) + Pr(outside | snow sport) + Pr(outside | sport speech) + Pr(outside | chair chair))

Pr(inside) = (0.0816+.5161+.8571) / (0.0816+.5161+.8571+.9184+.4839+.1429)
Pr(inside) = 1.4548 / 3 = 0.4849
Pr(outside) = 1-Pr(inside) = .5151

	x	y: snow	sport	chair	speech
Pr(y|x)	inside	0.0280	0.2054	0.5892	0.1774
	outside	0.2972	0.4538	0.0925	0.1566

The number in the chart are calculated as follows

Pr(snow|inside) = (count of snow and inside)/(count of inside words)
Pr(snow|inside) = (0.0816) / (2*(0.0816+.5161+.8571))
Pr(snow|inside) = .0280