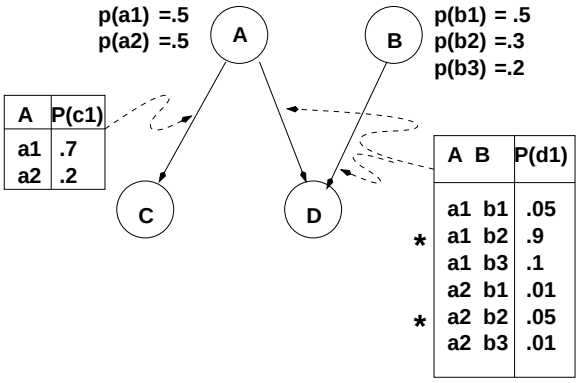
Final with Answers and FFQ (tm)

CSC 242

9 May 2003

*Write your **NAME** legibly on the bluebook. Work all problems. Best strategy is not to spend more than the indicated time on any question (minutes = points). Open book, open notes.*

1. Bayes Nets: 15 Mins



A	P(c1)
a1	.7
a2	.2

	A	B	P(d1)
	a1	b1	.05
*	a1	b2	.9
	a1	b3	.1
	a2	b1	.01
*	a2	b2	.05
	a2	b3	.01

Shown is a Bayes Net relating four random variables. Variable B can have three values, the others are binary (values of "true" (a_1, c_1, d_1) or "false" (a_2, c_2, d_2). Clearly $P(a_2) = (1 - P(a_1))$, etc. Shown are the prior probabilities of A and B and the conditional probability matrices (CPMs) linking A, B, C, D.

A (10 mins) . Suppose you find out that C is true (i.e. C's value is c_1). What is the resulting probability of b_2? I want first a nice neat formula into which we can substitute things we know. No numbers! Then if you'd like to compute a real value you may, but it isn't necessary or expected (or wise!).

B (5 mins). Suppose the entries for $P(d_1)$ in the two starred rows in the CPM are swapped. Does the answer to part 1. go up or down and why? Do you actually have to answer part 1 to figure this out and if not how not?

Answer:

A.

Lots of people thought that the "causal" arrows meant that variables were independent, but Bayes' rule says that isn't the case, right? I expected people just to write down a version of the formula for a similar example from the book but most people who got this did it by brute force, figuring out how to rewrite the probabilities by applying Bayes, working around the tree.

$$P(b_2 \mid c_1) = \alpha \sum_A \sum_B P(b_2)P(A)P(D \mid A, B)P(c_1 \mid A)$$

You can move non-summed-over factors outside various sums to get

$$\alpha P(b_2) \sum_A P(A) \sum_D P(D \mid A, B) P(c_1 \mid A)$$

which looks to take just four terms of 4 factors apiece to calculate.

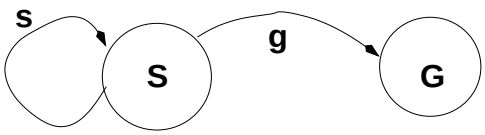
B.

Most people got this right.

You could substitute into your sums if you wanted and show what happens.

In general you don't need the formula though. You could explain about Bayes theorem and even quote it, but the intuition is that C makes A more likely, and A and b2 make D more likely with the original CPM. Swapping the two probabilities means that it is much less likely that d_1 occurs with b_2, so the answer should go down.

2. Reinforcement Learning: 40 Min.



R(S) = r = −0.1 **R(G) = R = 1.0**

The above rather simple world has a start state S and a goal state G, with action s (for "stay" or "start") that is meant to keep the agent in the start state, and g (for "go" or "goal") meant to take agent to goal. The world is completely observable to the agent (it knows what state it's in and what reward it's getting). At goal, agent exits the game; there are no more moves until its next trial. Say there is a future-reward discount factor of γ. Reward of the start state is r and the reward of the goal is R, which in this world are -0.1 and +1.0 respectively. Both rewards and discounts start when the agent is "born", so if the agent wakes up in S and takes action g successfully, its total reward is $-0.1 + \gamma \cdot 1.0$.

The world is uncertain. In fact the transition function $T(FromState, Action, ResultState)$ giving the probabilities of transitions given actions is: $T(S, s, S) = .9, T(S, s, G) = .1, T(S, g, S) = .2, T(S, g, G) = .8$. There aren't any $T(G, ., .)$ since in state G there aren't any moves, the game is over. Call $T(S, g, G) = p$, the probability of successfully "go"ing, and then $T(S, g, S) = q = 1 - p$, the probabilty of an unsuccessful "go" action.

A (1 min). What is the agent's optimal policy? (Don't get this wrong!)

B (10 min). What is the utility of S in the above situation for arbitrary $\gamma < 1$? Ideally, I'd like to see your work leading to an explicit, symbolic, closed-form solution. Less ideally, give a coherent approach to such a solution. A final number is not necessary.

C (9 min). If $\gamma = 1$ (additive rewards), write down your thus-simplified formula (or make a new one) and give your best estimate of the numerical value of the utility of state S using your formula (better) or your intuition.

D (10 min.). Now assume the agent knows neither how the world works nor the utilities of any states. However suppose it is following, by luck or accident, the optimal policy. It makes four trials in the world; in each trial it emits the best action until it finally gets to G. The trials yield the following state (not action!) sequences: SSG, SG, SG, SSSG. What is the agent's resulting estimate of $T(., ., .)$?

E (10 min). Now assume that the agent is a temporal difference learner. Assume that it computes (by direct utility estimation) the utilities of states S and G after the first trial above. Recall that direct utility estimation takes the expected total reward from a state onward to be its estimated utility. Then the agent makes the second trial above. What are its original and updated estimates of the utility of S? Assume that $\gamma = 1$ to keep the math simple. Use an α of 0.5.

Answer:

A. Optimal policy is always to "go". This has got to be obvious by inspection: hanging around costs, there's only one other thing to do...

For some mysterious reason just about EVERYone gave an answer like "$T(S, g, G)$" here, which is NOT an action but a probability.

B. Here I pretty much gave full credit to anyone (and there were precious few) who realized there was a nontrivial, in fact infinite, sum involved.

First step, agent does "g", and with probabilty p gets total reward $r + \gamma R$. If there's a step 2, then with probability pq the reward is $r + \gamma r + \gamma^2 R$. And so on, so I get the sum of these outcomes weighted by their probabilities as:

$$U(S) = \sum_{t=0}^{\infty} pq^t [\gamma^{t+1} R + \sum_{k=0}^{t} \gamma^k r] = 0.875.$$

The second sum simplifies to $r\frac{(\gamma^t - 1)}{(\gamma - 1)}$.

C. With $\gamma = 1$, the above simplifies to

$$U(S) = \sum pq^t [R + rt]$$

This isn't quite as intuitive to me as another formula, which describes your getting R except when you don't, which is the first time (you get 1- .1), and then with probability decreasing by factors of .2 every time, you lose another 0.1:

$$U(S) = 1 - .1 - .2(.1) - (.2)^2(.1)... = 1 - (.1)\sum_{t=0}^{\infty} .2^t = 1 - (0.1)\frac{1}{1 - .2} = 0.875.$$

D. Lots of people got this right.

I make it $T(S, g, S) = 3/7$ and $T(S, g, G) = 4/7$. Clearly no info on $T(S, s, .)$ since agent following best policy would never do that, and none on $T(G, ., .)$ since those are terminal states.

E.

I gave full credit to anyone who even wrote down an approximation to the TD formula.

After first trial, utility estimates are the expected reward to go of the states it experienced. So 1st S was worth .8, the 2nd S was .9, and G was worth 1. Thus the estimated utility of S is average for S, (or .85) after this trial and expected utility of G is 1.

Applying the TD update on the second trial (SG),

$$U(S) = .85 + \alpha(-.1 + 1 - .85) = .85 + .5(.05) = .875.$$

This looks pretty smart, but the fact that we're at the "true" utility is an accident of choice for α, and isn't permanent....

3. Neural Nets 15 Min.

A (5 min.) We've discussed the delta rule for perceptrons and the back-propagation rule for feed-forward multi-layer nets. Give your thoughts about using other search or optimization techniques we've studied to find correct weights in neural nets.

B (10 min). The NAND boolean function is 1 unless both its input are 1. Write a linear threshold device (1 unit, two inputs and a threshold) to compute NAND. Draw or carefully describe its decision surface in (x, y) space.

Answer

A. Delta and backprop are forms of gradient ascent, which is why you need a differentiable activation function to use them for multilayer perceptrons. Also the rules derive from particular choices of error metric (squared error) at the outputs. One could imagine other error choices and the whole range of nonlinear optimization techniques, such as one-dimension at a time search (change a random weight and if there's an improvement, keep it else try something else), simulated annealing (or Cauchy or Boltzmann training) or genetic algorithms. Hybrid approaches are also possible, with hill-climbing (aka backprop) alternating with some random component to get off of local maxima.

B. $W_x = -1, W_y = -1, T = 1.5$ works, giving a -45 degree slope line that slices between the "yes" points (1,0) and (0,1) and the "no" point (1,1).

4. Linear Systems: 25 Min.

A (5 min). The attached sheet has four images of texture and four power spectra. Supply the number of the corresponding image for each power spectrum's letter:

a:

b:

c:

d:

B (10 min). Suppose you are designing a digital camera. You plan to have a FFT chip that furnishes the power spectrum (in real time, of course) for your internal use. Also of course you'll have the image that gets passed to the consumer. How would you implement an *autofocus* capability for the camera given the power spectrum?

C. (10 min) How would you use an image transform like the FFT to do image compression?.

Answer:

A.

a:3

b:1

c:4

d:2

B. Move focus in and out while monitoring the energy in the high-frequency part of the spectrum by integrating the power spectrum for frequencies above some fraction of the band-limit of the particular image you're getting. Stop at the maximum.

C. I gave half credit for the idea of tossing out low coefficients, and half for the idea you had to do this locally, not on the whole image. Unfortunately nobody got the second 5 points!

Tile the image up into small squares (maybe 8x8 pixels), take the FFT (or if you're JPG, the Discrete Cosine Transform) of the patch and keep the largest coefficients of the frequencies. Keeping them all allows perfect recovery but no compression. In the limit of one tile per image, you apply the same attenuation of wavelengths everywhere, which is not likely to be useful (could vary from edge-enhancement to blurring to something like emphasizing the "midrange", done uniformly all over the image).

5. Vision and Robotics: 25 Mins.

A. (3 min). Express the Cartesian vector (x, y, z) in homogeneous coordinates.

B. (3 min). Express the homogeneous vector (6, 8, 4, 2) in 3-D Cartesian coordinates.

C. (3 min). What good are homogeneous coordinates anyway?

D. (3 min). Which is harder, forward or inverse kinematics and why?

E. (8 min.) We've seen how the reflectance map $R(p, q)$ gives one-dimensional loci of surface normals corresponding to iso-brightness contours in an image, under some assumptions. Which of the following assumptions allow and which invalidate the use of a single $R(p, q)$, and (very briefly) explain what's wrong in the bad cases.

1. Uniform surface material
2. Multiple light sources at infinity
3. Light source near object
4. Shiny surface
5. Moving light sources at infinity
6. Cast shadows
7. Object rotation
8. Perspective effects

F. (5 min.) Suppose your object and imaging situation meets all the assumptions for shape from shading and you have two lighting setups (say a light from the left and a light from the right) you can switch between. How can you vastly simplify the search for surface normals that conform to the image brightness data available to you? (This trick is called *photometric stereo*).

Answer

Starting off with a few giveaways to reward people who came to class. However, about one person got the first two parts right...

A. $(x, y, z, 1)$

B. (3,4,2)

C. Express affine transforms (in N dims) as linear (in (N+1) dims). Thus translation, rotation (and even perspective, though we didn't see that) can all be expressed as one matrix mpy.

D. inverse, since there can be several manipulator configurations to achieve the same tip position. In general it's an underdetermined system. Forward kinematics have a unique solution.

E. The "object rotation" is ambiguous, my fault. If the object is rotating, it's a BAD, if it's rotated its an OK.

1. Uniform surface material: OK
2. Multiple light sources at infinity: OK
3. Light source near object: BAD – light direction not uniform
4. Shiny surface – OK

5. Moving light sources at infinity BAD – ditto

6. Cast shadows – BAD – cause change in brightness not attributable to slant and tilt.

7. Object rotation – OK

8. Perspective effects – BAD – distort slant and tilt in image nonuniformly.

F. I think one person got this idea. For any point in the image, get its brightness and thus its locus (the contour at that brightness) in both the $R(p, q)$ reflectance maps. These loci should intersect in at most two places, so you have a choice of only two, not a one-dimensional infinity, of surface orientations for that point. Clearly with three sources you get uniqueness and with more you can get more robust estimations.