1 RL: Open Discussion

Recall that in Q-learning, we continually update the values of each Q-state by learning through a series of episodes, ultimately converging upon the optimal policy.

(a) Assume that we are given our exploration probability $\epsilon = 1$. Are we still guaranteed to converge upon the optimal policy?

Yes! Given enough iterations, the agent would be able to still calculate the Q-values from its actions. However, the learning will take considerably longer than if $\epsilon < 1$, because the agent would not be able to exploit its learned Q-values, thereby making the same mistakes repeatedly until the correct solution is found by random chance.

(b) Consider a variant of the Q-iteration algorithm that is changed such that instead of using the policy extracted from our current Q-values, we use a fixed policy instead, with exploration. If this fixed policy happens to be optimal, how does the performance of this algorithm compare to normal Q-iteration?

Both algorithms will result in finding the optimal Q-values eventually. However, normal Q-iteration makes more mistakes along the way, racking up more *regret* (the difference between actual yielded rewards and the optimal rewards).
In practice, normal Q-iteration may lead to a policy that is "pretty good" but not necessarily optimal, thus making it very unlikely for it to change unless given an extremely high number of iterations to allow for random chance to find a better policy. This result is known as a local optimum.

(c) How would we be able to tell if a given policy is optimal or not?

In order to know whether our policy is optimal, we would still need to run Q-iteration and explore other states in the space. For this reason, it will still take the same amount of iterations to explore all the possible state-action pairs to fully verify that another policy isn't better.
Note that it is not necessarily true that we have the optimal policy if the policy extracted from our learned Q-values stays the same from one iteration to the next (which was our gauge for convergence in policy iteration) - we may not yet have sufficiently explored the entire state space.

(d) **(Bonus)** Let's revisit the CandyGrab code from recitation 1 (https://www.cs.cmu.edu/~15381/recitations/rec1/candygrab.zip). What RL strategies does `AgentRL` employ? Does it evaluate states or Q-states?

Note that `AgentRL` plays out each game (either randomly playing each round with probability ϵ if `explore_mode` is on, or by exploiting its learned policy) and records the lose/win rate for each state, action pair seen along the way.
`AgentRL` uses direct evaluation, with an option to execute ϵ-greedy exploration. It evaluates Q-states.

2 Chain Rule

In class, we discussed the product rule, which states that $P(A, B) = P(A|B)P(B) = P(B|A)P(A)$. Let's now try to extend this to understand the chain rule. By the chain rule, we can write any joint distribution as an incremental product of conditional distributions (in other words, the chain rule can be found by repeated application of the product rule).

(a) Apply the chain rule to express $P(A, B, C)$ in terms of $P(A), P(B|A)$, and $P(C|A, B)$.

$P(A,B,C) = P(C|A,B)P(A,B) = P(C|A,B)P(B|A)P(A)$

(b) There are five other ways to express this probability. Write them out below.

$P(C|A,B)P(A|B)P(B)$
$P(B|A,C)P(A|C)P(C)$
$P(B|A,C)P(C|A)P(A)$
$P(A|B,C)P(B|C)P(C)$
$P(A|B,C)P(C|B)P(B)$

(c) Finally, let's also express $P(A,B,C,D)$ (in any way) using the chain rule.

Possible solution: $P(D|A,B,C)P(C|B,A)P(B|A)P(A)$

3 The Distributive Property

We'll start with something more basic. This example will serve as a good reference in trying to understand more complicated concepts.

(a) Compute $\sum_{x=1}^{3}\left(\sum_{y=4}^{6} xy\right)$

$$\begin{aligned}\sum_{x=1}^{3}\left(\sum_{y=4}^{6} xy\right) &= \sum_{x=1}^{3}(4x+5x+6x)\\ &= (4(1)+5(1)+6(1))+(4(2)+5(2)+6(2))+(4(3)+5(3)+6(3))\\ &= \boxed{90}\end{aligned}$$

Note that we deliberately avoided using the distributive property here (i.e. that $4x+5x+6x=15x$).

(b) Compute $\sum_{x=1}^{3}\left(x\sum_{y=4}^{6} y\right)$

$$\begin{aligned}\sum_{x=1}^{3}\left(x\sum_{y=4}^{6} y\right) &= \sum_{x=1}^{3}(x(4+5+6))\\ &= \sum_{x=1}^{3} 15x\\ &= 15(1)+15(2)+15(3)\\ &= \boxed{90}\end{aligned}$$

(c) What do you notice about the values? Why is this the case?

These two evaluate to the same value. The summation in part (b) can be obtained by factoring x out of the sum over values of y. Notice that the latter way of evaluating the sum requires far fewer arithmetic operations. This means that it is more computationally efficient. This is the motivating idea behind everything that will be covered in this worksheet.

4 Generic Functions

Consider two sets $X = \{x_1, x_2, x_3\}$ and $Y = \{y_1, y_2\}$. Let us also define arbitrary functions $f(x)$, $g(y)$, and $h(x,y)$. The notation $\sum_{x\in X} f(x)$ means that we want to apply the function f to all elements of X and add all the results. For the X defined above, this means $f(x_1)+f(x_2)+f(x_3)$.

(a) Consider the sums $\sum_{x\in X}\sum_{y\in Y} h(x,y)$ and $\sum_{y\in Y}\sum_{x\in X} h(x,y)$. Are they the same? Can they be simplified? In general, is it valid to switch the sums in this manner?

$$
\begin{aligned}
&\sum_{x\in X}\sum_{y\in Y} h(x,y)\\
&= \sum_{x\in X}(h(x,y_1)+h(x,y_2))\\
&= (h(x_1,y_1)+h(x_1,y_2))+(h(x_2,y_1)+h(x_2,y_2))+(h(x_3,y_1)+h(x_3,y_2))\\
&= (h(x_1,y_1)+h(x_2,y_1)+h(x_3,y_1))+(h(x_1,y_2)+h(x_2,y_2)+h(x_3,y_2)))\\
&= \sum_{y\in Y}(h(x_1,y)+h(x_2,y)+h(x_3,y))\\
&= \sum_{y\in Y}\sum_{x\in X} h(x,y)
\end{aligned}
$$

Thus rearranging the order of the sums does not change the value of the sum. This is true in general for sums of this form.

(b) Now consider

$$\sum_{x\in X}\sum_{y\in Y} f(x)g(y)h(x,y).$$

Can we simplify the calculation of this sum?

Using the same reasoning as on the first page, since $f(x)$ is only a function of x, it can be factored out of the sum over values of y, giving

$$\sum_{x\in X} f(x)\sum_{y\in Y} g(y)h(x,y).$$

We can also reverse the order of the sums and factor out $g(y)$ to get

$$\sum_{y\in Y} g(y)\sum_{x\in X} f(x)h(x,y).$$

5 Magnetic Factors

Observe that in a summation over x, any term that does not depend on x can be treated as a constant, and therefore moved out of the sum. This is to say that $\sum_{x \in X} cf(x) = c\left(\sum_{x \in X} f(x)\right)$, where c is any term that is constant *with respect to* x. This includes functions of y or any other variable.

Consider this analogy: pretend each Σ is a *magnet* that attracts only factors containing the variable it is iterating over. Factors can only pass through this magnet if they are independent of that variable. All factors will be pulled as far to the left as possible. Since some factors contain more than one variable, the ordering of the summations affects the result, and choosing an optimal ordering can greatly speed up computation.

With this in mind, simplify the following expressions so that computing them requires as few operations as possible.

(a) $\sum_{s \in S} \sum_{r \in R} \sum_{q \in Q} f(q)g(q,r)h(q,r,s)$

The first thing to notice is that since the inner sum is over q, since all the terms involve q, nothing can be factored out of the inner sum. But as shown in the previous question, we can change the order of the sums to get

$$\begin{aligned}\sum_{s \in S} \sum_{r \in R} \sum_{q \in Q} f(q)g(q,r)h(q,r,s) &= \sum_{q \in Q} \sum_{r \in R} \sum_{s \in S} f(q)g(q,r)h(q,r,s) \\ &= \sum_{q \in Q} f(q) \sum_{r \in R} g(q,r) \sum_{s \in S} h(q,r,s)\end{aligned}$$

Make sure you are convinced that this ordering is optimal, i.e. that no other ordering could result in more functions being factored further.

(b) $\sum_{a \in A} \sum_{b \in B} \sum_{c \in C} f(b)f(c)h(a,b,c)g(a,j)g(a,m)$

Here the key observation is that j and m are constants with respect to all the variables being summed over, so they can be moved about freely. Thus the $g(a,j)$ and $g(a,m)$ factors are only restricted by a.

$$\sum_{a \in A} \sum_{b \in B} \sum_{c \in C} f(b)f(c)h(a,b,c)g(a,j)g(a,m) = \sum_{a \in A} g(a,j)g(a,m) \sum_{b \in B} f(b) \sum_{c \in C} f(c)h(a,b,c)$$

In this case, we can also do just as well by switching the order of the summations over b and c.

(c) $\sum_{d \in D} \sum_{k \in K} \sum_{c \in C} \sum_{x \in X} g(c,k)f(x)h(c,k,x)f(k)l(c,d,k,x)g(k,x)$

$$= \sum_{x \in X} f(x) \sum_{k \in K} f(k)g(k,x) \sum_{c \in C} g(c,k)h(c,k,x) \sum_{d \in D} l(c,d,k,x)$$

Here we can also do just as well by switching the order of the sums over x and k (but then it wouldn't spell XKCD). Make sure you are convinced that no other ordering of the sums could be better.

6 And Finally, Some Probability Again

Recall the example of *marginalization*, which means summing out variables from a joint distribution. Consider three binary random variables A, B, and C with domains $\{+a, -a\}$, $\{+b, -b\}$, and $\{+c, -c\}$, respectively. Remember that $P(A)$ refers to the table of probabilities of all the elements of the domain A.

(a) Express $P(A)$ in terms of the joint distribution $P(A, B, C)$. Your answers should contain summations.

We can find $P(A)$ by summing out the variables B and C from the joint distribution $P(A, B, C)$. Remember that $P(A)$ is a probability table, so the equations below can take on two values: one for $+a$ and one for $-a$.

$$P(A) = \sum_{B \in \{+b,-b\}} \sum_{C \in \{+c,-c\}} P(A, B, C)$$

For convenience, we will abbreviate the above sums as

$$P(A) = \sum_B \sum_C P(A, B, C)$$

(b) Express $P(A)$ in terms of $P(C)$, $P(B \mid C)$ and $P(A \mid B, C)$.

By the chain rule, $P(C)P(B \mid C)P(A \mid B, C) = P(A, B, C)$, so we can substitute this into the expression from (a) and get

$$P(A) = \sum_B \sum_C P(C)P(B \mid C)P(A \mid B, C)$$

We can reorder the sums and factor just like we did in the previous two sections to get

$$P(A) = \sum_C P(C) \sum_B P(B \mid C)P(A \mid B, C)$$

(c) Expand the sums from part (b) to show the two elements of $P(A)$ ($P(+a)$ and $P(-a)$) in terms of the individual probabilities (e.g. $P(+b)$, $P(-c)$ instead of $P(B)$ or $P(C)$).

We evaluate these sums by simply plugging in each possible value for b and c and taking the sum. To get the individual element in the table $P(A)$, we plug in the corresponding value (either $+a$ or $-a$).

$$\begin{aligned} P(+a) = {} & P(+c)\left(P(+b \mid +c)P(+a \mid +b, +c) + P(-b \mid +c)P(+a \mid -b, +c)\right) \\ & + P(-c)\left(P(+b \mid -c)P(+a \mid +b, -c) + P(-b \mid -c)P(+a \mid -b, -c)\right) \end{aligned}$$

$$\begin{aligned} P(-a) = {} & P(+c)\left(P(+b \mid +c)P(-a \mid +b, +c) + P(-b \mid +c)P(-a \mid -b, +c)\right) \\ & + P(-c)\left(P(+b \mid -c)P(-a \mid +b, -c) + P(-b \mid -c)P(-a \mid -b, -c)\right) \end{aligned}$$

Thus we have achieved concrete expressions for $P(+a)$ and $P(-a)$ in terms of the given values (the rows in the conditional probability tables).

This is the process by which we perform inference in a Bayes net. We are given several conditional probabilities which multiply together to give a joint distribution. To answer a query about one of them, we sum out the result of multiplying the tables together. However, in order to make the process more computationally efficient, we also will try to factor the sum as much as possible.