CMPUT 605: Theoretical Foundations of Reinforcement Learning, Winter 2023
Homework #0

Instructions: You need to submit a single PDF file, named `p01_<name>.pdf` where `<name>` is your name. The PDF file should include your typed up solutions (we strongly encourage to use pdfLaTeX). Write your name in the title of your PDF file. We provide a LaTeXtemplate that you are encouraged to use. To submit your PDF file you should send the PDF file via private message to Vlad Tkachuk on Slack before the deadline.

Collaboration and sources Work on your own. You can consult the problems with your classmates, use books or web, papers, etc. Also, the write-up must be your own and you must acknowledge all the sources (names of people you worked with, books, webpages etc., including class notes.) Failure to do so will be considered cheating. Identical or similar write-ups will be considered cheating as well. Students are expected to understand and explain all the steps of their proofs.

Scheduling Start early: It takes time to solve the problems, as well as to write down the solutions. Most problems should have a short solution (and you can refer to results we have learned about to shorten your solution). Don't repeat calculations that we did in the class unnecessarily.

Deadline: January 15 at 11:55 pm

1 In search of ...

Consider the following simple learning problem: Fix $\delta > 0$ and an integer $k > 0$. Here, k will be the number of actions available. Each action $i \in [k] := \{1, \dots, k\}$ is assigned a value, say, μ_i, which is a real number. The learner does not know these values. The problem of the learner is to find an action that is δ close to the best action out of k actions (higher values are better). The learner can try any action in any order and can even randomize when choosing the next action. When choosing an action, the learner receives in response the value of the action (without noise!). When choosing an action, the learner can take into account all the past observations and choices: It has no limit of what it can remember, or on the precision of its memory. At one point, the learner needs to stop and return an action.

A *problem instance* that the learner interacts with is fully characterized by $\mu = (\mu_1, \dots, \mu_k) \in \mathbb{R}^k$, the k values assigned to the actions. The learner is called *δ-sound* for a set of instances $\mathcal{H} \subset \mathbb{R}^k$ if no matter the problem instance taken from $\mathcal{H}$, when the learner interacts with the aforementioned way with the chosen problem instance, the learner always stops and returns an action that is strictly less than δ away from the optimal action on that instance. If the instance was $\mu \in \mathbb{R}^k$, the learner chose $A \in [k]$, with probability one, it has to hold that $\mu_A > \max_{i \in [k]} \mu_i - \delta$ (note the strict inequality). For a learner $\mathcal{A}$, let $q(\mathcal{A}, \mu)$ be the expected value of the number of rounds that the learner spends with interacting with problem instance μ (we need the expected value, because learners may randomize). By slightly abusing notation, let $q(\mathcal{A}, \mathcal{H}) = \max_{\mu \in \mathcal{H}} q(\mathcal{A}, \mu)$ be the expected number of rounds that the learner $\mathcal{A}$ will spend on the instance in $\mathcal{H}$ which makes it the "slowest".

Let $\mathcal{S}(\mathcal{H}, \delta)$ be the set of δ-sound learners for $\mathcal{H}$. Further, let $\mathcal{B} = \{e_1, \dots, e_k\} \subset \mathbb{R}^k$ where e_i is the ith standard basis vector: $e_{ij} = 0$ if $j \neq i$ and $e_{ii} = 1$.

Question 1 Describe a *deterministic* learner $\mathcal{A} \in \mathcal{S}(\mathcal{B}, 1)$ such that the learner stops on any instance in $\mathcal{B}$ after at most $k - 1$ queries: $q(\mathcal{A}, \mathcal{B}) \leq k - 1$.

Total: **5 points**

Solution. *Loop through the actions, starting from action* 1 *and finishing with action* $k - 1$*: In the* i*th step of this loop, ask for the feedback (response) for action* i*. If the feedback is* 1*, return* i*. If the loop finishes, return* k*.* □

Question 2 Formally prove that for the learner $\mathcal{A}$ you described $\mathcal{A} \in \mathcal{S}(\mathcal{B}, 1)$ holds.

Total: **5 points**

Solution. *It suffices to prove that for any $i \in [k]$, when $\mathcal{A}$ interacts with the environment e_i, it will return index i because this implies that for the returned action $A \in [k]$, $\mu_A = 1 > \max_{j \in [k]} e_{ij} - 1 = 1 - 1 = 0$. Fix $i \in [k]$, arbitrarily. There are two cases: If $i \leq k-1$, the learner in the main loop will get to ask for the feedback for action i. This is because for all earlier queries, it receives the value of 0 ($e_{ij} = 0$ for $j < i$). When asking for the feedback for action i, it receives the value of 1 ($e_{ii} = 1$) and thus quits the loop and returns i, which is the correct output. The second case is if $i = k$. In this case, by the previous argument, the loop finishes without quitting early. The learner then returns $i = k$, which is again the correct output. Since no matter the value of i, the learner's output is correct, $\mathcal{A} \in \mathcal{S}(\mathcal{B}, 1)$ holds.* □

Question 3 Formally prove that for the learner $\mathcal{A}$ you described $q(\mathcal{A}, \mathcal{B}) \leq k - 1$ holds.

Total: **5 points**

Solution. *No matter the instance that the learner interacts with, the learner's loop lasts for at most $k-1$ steps. In each step of the loop one query is submitted. Therefore, the number of queries the learner submits is at most $k-1$.* □

Question 4 Show that any deterministic learner $\mathcal{A} \in \mathcal{S}(\mathcal{B}, 1)$ needs at least $k-1$ queries in the worst case for instances in $\mathcal{B}$.

Total: **10 points**

Solution. *Take a deterministic algorithm $\mathcal{A}$. Let the queries issued by $\mathcal{A}$ on the all-zero input be $s_1, s_2, \cdots \in [k]$: This is a finite sequence if the algorithm stops querying (e.g., returns), otherwise it is an infinite sequence. If the length of this sequence is at least $k-1$ then on e_i with $i \in [k] \setminus \{s_1, \ldots, s_{k-1}\}$ (which exist because $[k] \setminus \{s_1, \ldots, s_{k-1}\}$ is non-empty), $\mathcal{A}$ will issue at least $k-1$ queries because on this instance, the feedbacks corresponding to the first $k-1$ queries of $\mathcal{A}$ are zero, so the first $k-1$ queries of $\mathcal{A}$ will be $s_1, \ldots, s_{k-1}$. Thus, in this case there is nothing to be proven.*

Now assume that the length ℓ of this sequence is strictly smaller than $k-1$. We show that in this case $A \in \mathcal{S}(\mathcal{B}, 1)$ cannot hold and thus from $A \in \mathcal{S}(\mathcal{B}, 1)$ it follows that we cannot be in this case. Thus, the proof will be finished once we show that $A \notin \mathcal{S}(\mathcal{B}, 1)$ proved that $\ell \leq k-2$.

Take $i, j \in [k] \setminus \{s_1, s_2, \ldots, s_\ell\}$ such that $i \neq j$. Such numbers exist because $|\{s_1, s_2, \ldots, s_\ell\}| \leq \ell \leq k-2$ and thus $[k] \setminus \{s_1, s_2, \ldots, s_\ell\}$ has at least $k - (k-2) = 2$ elements. Since $\mathcal{A}$ never queries either action i or action j, $\mathcal{A}$ will receive the same feedbacks on both e_i and e_j. Now, $\mathcal{A}$, being deterministic, will return the same answer $A \in [k]$ on both e_i and e_j. However, either $A \neq i$ or $A \neq j$, hence $\mathcal{A}$ returns the index of an action with payoff zero on at least one of e_i and e_j. Hence, $\mathcal{A} \notin \mathcal{S}(\mathcal{B}, 1)$. □

Question 5 Describe a learner $\mathcal{A} \in \mathcal{S}(\mathcal{B}, 1)$ (deterministic or not) such that the learner stops on any instance in $\mathcal{B}$ after at most $(k+1)/2-1/k$ queries on expectation: $q(\mathcal{A}, \mathcal{B}) \leq (k+1)/2-1/k$. Formally prove that the learner is indeed sound and it indeed stops after at most $(k+1)/2-1/k$ queries on expectation, on every problem instance in $\mathcal{B}$.

Total: **15 points**

Solution. *Consider an algorithm that queries each action in a random order, stopping as soon as it receives a nonzero feedback, at which point the algorithm can return the index of the action just queried. Clearly, this algorithm is sound. For calculating the expected number of queries let $S \in \mathrm{Perm}([k])$ be a (uniform) random permutation of $[k]$: This is the order in which the algorithm queries the actions (i.e., $S(1)$ is queried first, followed by $S(2)$, etc.).*

Fix $i \in [k]$ and let $T = \min\{t \geq 1 : S(t) = i\}$ be the number of queries when $\mu = e_i$. Note that $T = S^{-1}(i)$ and as such that $\mathbb{P}(T = t) = \mathbb{P}(S(t) = i) = 1/k$ for any $t \in [k]$, i.e., T is uniformly distributed over $\{1, \dots, k\}$. Now consider the slightly improved algorithm, which, using the principle of elimination, queries at most $k-1$ items. The runtime of this algorithms on instance $\mu = e_i$ is $\tilde{T} = \min(T, k-1)$ and we have $\mathbb{E}[\tilde{T}] = \sum_{t=1}^{k-1} t\,\mathbb{P}(\tilde{T} = t) = \sum_{t=1}^{k-1} t\,\mathbb{P}(T = t) + (k-1)\mathbb{P}(T = k)$. Plugging in $\mathbb{P}(T = t) = 1/k$, we have

$$\mathbb{E}[\tilde{T}] = \frac{1}{k}\left(-1 + \underbrace{\sum_{t=1}^{k} t}_{k(k+1)/2}\right) = \frac{k+1}{2} - \frac{1}{k}\,. \tag{1}$$

□

Question 6 Show that any learner $\mathcal{A} \in \mathcal{S}(\mathcal{B}, 1)$ needs at least $(k+1)/2 - 1/k$ queries on expectation in the worst case for instances in $\mathcal{B}$. That is, prove that for any $\mathcal{A} \in \mathcal{S}(\mathcal{B}, 1)$, $q(\mathcal{A}, \mathcal{B}) \geq (k+1)/2 - 1/k$.

Hint: Yao's lemma states "that the expected cost of a randomized algorithm on the worst-case input is no better than the expected cost for a worst-case probability distribution on the inputs of the deterministic algorithm that performs best against that distribution." (source: wikipedia). Use this lemma.

Total: **15 points**

Solution. *Following the hint, by Yao's lemma, it suffices to show that $q^* \geq (k+1)/2 - 1/k$ where*

$$q^* = \frac{1}{k} \inf_{\mathcal{A} \in \mathcal{S}^*(\mathcal{B}, 1)} \sum_{i=1}^{k} q(\mathcal{A}, e_i)\,,$$

where $\mathcal{S}^*(\mathcal{B}, 1)$ is $\mathcal{S}(\mathcal{B}, 1)$ restricted to deterministic algorithms (we chose the uniform distribution over the input instances $e_1, \dots, e_k$).

Take $\mathcal{A} \in \mathcal{S}^*(\mathcal{B}, 1)$. It suffices to show that

$$\frac{1}{k} \sum_{i=1}^{k} q(\mathcal{A}, e_i) \geq \frac{k+1}{2} - \frac{1}{k}\,. \tag{2}$$

Assume that $\mathcal{A}$ sometimes repeats some queries. Define $\mathcal{A}'$ as follows: in each round of interaction $\mathcal{A}'$ first calls $\mathcal{A}$, checks whether $\mathcal{A}$ issued a query that was issued before and if so feed $\mathcal{A}$ with the answer received earlier, otherwise issues the query to the environment and return the feedback received back to $\mathcal{A}$. This algorithm $\mathcal{A}'$ works identically on all inputs as $\mathcal{A}$ and never issue a query twice. In particular, for all $i \in [k]$,

$q(\mathcal{A}, e_i) \geq q(\mathcal{A}', e_i)$ and so (2) will follow if we show it with $\mathcal{A} := \mathcal{A}'$. Thus, in what follows, without loss of generality, we assume that $\mathcal{A}$ does not repeat any queries. Similarly, we can assume without loss of generality that $\mathcal{A}$ stops as soon as it receives a nonzero response.

Like in the solution to Question 4, let the queries issued by $\mathcal{A}$ on the all-zero input be $s_1, s_2, \cdots \in [k]$: By the answer to Question 4 the length of this sequence is at least $k-1$. By our assumption the sequence $(s_1, \ldots, s_{k-1})$ does not have repeated elements. Define $s_k \in [k]$ to be the unique action index which is left out from $\{s_1, \ldots, s_{k-1}\}$. Since $(s_1, s_2, \ldots, s_k)$ does not have repeated elements, it is a permutation of $[k]$. By assumption, $\mathcal{A}$ must stop as soon as it receives a nonzero answer and because $\mathcal{A} \in \mathcal{S}(\mathcal{B}, 1)$, it cannot stop earlier. Let $t(i) = \min\{t \geq 1 : s_t = i\}$. It follows that the number of queries issued by $\mathcal{A}$ when run on e_i is $\min(k-1, t(i))$. Since $(s_i)_i$ is a permutation of $[k]$, so is $(t(i))_{i \in [k]}$. Let $I \sim P$, i.e., I is uniformly distributed on $\{1, \ldots, k\}$. We have $\mathbb{P}(T = t(I)) = 1/k$ and thus, with the same calculation as before, $\mathbb{E}[\min(k-1, T)] = (k+1)/2 - 1/k$ (cf. Eq. (1)). □

Question 7 (for fun) It appears that as far as the expected number of rounds is considered, randomized learners can do better than deterministic learners. Is this *real*? Would *you* implement the randomized learner? Why or why not?

Total: **0 points**

Solution. *The answer depends on the following factors:*

- *Is k large?*
- *How many times will we run this algorithm?*
- *Do we care about the cost of generating a uniform random permutation?*
- *Do we care about the simplicity of the code?*

In particular, if k is large, I expect the runtime on any single instance e_i to be relatively close to the expected runtime (relative to k) on any single run, with a "good chance". Thus, with high probability, the prediction is that the randomized algorithm will save over the non-randomized one, regardless of the input. Hence, it may be worth the effort to implement it.

On the second item, if the algorithm will be run on many inputs, then over the many inputs, the average runtime is expected to even more "tightly concentrated" around the expected runtime. This is a good case for the randomizing algorithm.

If we care about the cost of generating the queries, we need to pay some cost for generating the queries uniformly at random. The Fisher-Yates algorithm has a linear cost, thus it adds negligibly to the total runtime, but sometimes even this little cost could matter. Probably not too often. But if it matters, randomization will be out.

By making a code more complex, we may introduce subtle errors in the code. Thus, if reliability is important, a simpler solution may be preferred. If the fancy solution is to be supported, one needs to carefully verify the correctness of the code. There are also subtleties related to whether a specific algorithm plays well with a random number generators. In general, one should just use a library function if available (almost all modern libraries will have fast algorithms for random shuffling, which is what is needed here). □

---

2 Basic probability questions

Assume that you are given a randomized algorithm $\mathcal{B}$ that returns the *unique* correct answer on any problem it is fed with (e.g., computing a shortest path) with probability at least $2/3$. Fix $\delta > 0$.

Question 8 Design an algorithm that returns the correct answer with probability at least $1 - \delta$. The algorithm may use $\mathcal{B}$. Describe how your algorithm works (pseudocode preferred).

Total: **5 points**

Solution. *See solution for Question 9* □

Question 9 Bound the expected runtime of your algorithm as a function of the expected runtime of $\mathcal{B}$. In particular, if the expected runtime of $\mathcal{B}$ on input x is $r(\mathcal{B}, x)$, then show that the expected runtime of your algorithm on input x is at most $\lceil 72 \ln(1/\delta) \rceil \, r(\mathcal{B}, x)$. If needed modify your algorithm (in which case modify your answer to the previous question).
Hint: Use the "Chernoff lower tail bound" for independent Bernoulli variables. This states the following: Let S be the sum of n independent Bernoulli random variables. Let $\mu = \mathbb{E}[S]$ be the expected value of S. Then, for any $0 \leq \delta \leq 1$, $\mathbb{P}(S \leq (1-\delta)\mu) \leq \exp(-\mu\delta^2/3)$.

Total: **5 points**

Solution. *First let us state our algorithm then bound its runtime. The mode of a list is the item on it with the highest repeat count.*

Input: $n \in \mathbb{N}$, subroutine $\mathcal{B}$, x

1. *Initialize list $V = ()$*
2. *for $i = 1, 2, ..., n$ do*
3. *$V = V.append(\mathcal{B}(x))$*
4. *end loop*
5. *return mode of V.*

Thus the question is how to choose n such that our algorithm outputs the correct answer with probability at least $1 - \delta$.

Let Y_i be the value returned by $\mathcal{B}(x)$, on its ith call. Let y be the unique correct answer. Let $B_i = \mathbb{I}(Y_i = y)$ be the indicator that the ith answer is correct and let $S = \sum_{i=1}^{n} B_i$. Thus, S counts the number of successes of the calls.

Now, the algorithm is guaranteed to be correct when $S \geq n/2$. This is because the second highest repeat count on V in this case must be strictly less than $n/2$, hence y "wins" (i.e., $Y = y$). Note that $(B_i)_i$ is an i.i.d. sequence: This follows from the (somewhat sneaky, implicit, but sensible) assumption that $\mathcal{B}(x)$ randomizes independently between the calls to it and that it randomizes the same way.

Hence, S is the sum of n independent, identically distributed Bernoulli variables. We are preparing to use the hint to bound the probability of failure from above. We have $\{Y \neq y\} = \{Y = y\}^c$ and since $\{S \geq n/2\} \subset \{Y = y\}$ by the previous argument, $\{Y = y\}^c \subset \{S < n/2\}$. Hence, $\mathbb{P}(Y \neq y) \leq \mathbb{P}(S < n/2)$. Let $\mu = \mathbb{E}[S]$ as in the hint. Since $\mathbb{P}(Y_i = y) \geq 2/3$ by our assumption, $\mu \geq 2/3n$. Thus,

$$\mathbb{P}\left(Y \neq y\right) \leq \mathbb{P}\left(S < n/2\right) = \mathbb{P}\left(S < (1 - 1/4)\frac{2n}{3}\right) \leq \mathbb{P}\left(S < (1-1/4)\mu\right) \leq \exp\left(-\mu\frac{2(1/4)^2}{3}\right)$$
$$\leq \exp\left(-\frac{2}{9}\left(\frac{1}{4}\right)^2 n\right),$$

where in the last inequality we used that $\exp(-x)$ is monotone decreasing and that $\mu \geq 2/3n$. Thus, it suffices to choose n large enough so that $\exp\left(-\frac{2}{9}\left(\frac{1}{4}\right)^2 n\right) \leq \delta$ holds.

By taking the log *of the last two equation on the right we get that we should run our algorithm at most* $n = \lceil 72 \ln(1/\delta) \rceil$. *Multiplying by the runtime if* $\mathcal{B}$ *on input* x, *we can bound the runtime of the algorithm by* $\lceil 72 \ln(1/\delta) \rceil \, r(\mathcal{B}, x)$. □

Question 10 For the same setting as in Question 8, now consider the case when $\mathcal{B}$ itself is correct with probability at least $1 - \delta$. Consider the case when $\mathcal{B}$ is run on s different inputs. Show that for any $s \leq 1/\delta$, the probability that $\mathcal{B}$ is correct on *all these inputs* is at least $1 - s\delta$.

Total: **5 points**

Solution. *Call a call to* $\mathcal{B}$ *a failure when the call fails to produce the correct output. Call a run a failure when* $\mathcal{B}$ *fails on any of its calls. Let* F_i *be the event that* $\mathcal{B}$ *fails on the* i*th call. Let* F *be the event that a run fails. Then the run fails if and only if any of the events* F_i *holds. That is,* $F = \cup_i F_i$ *(if* $\omega \in F$, *the run fails on* ω *then it must that* $\omega \in F_i$ *for at least one index* i *and if* $\omega \in \cup_i F_i$ *then at least for one index* i, $\omega \in F_i$, *which means that* $\omega \in F$*). Hence,*

$$\mathbb{P}(F) = \mathbb{P}(\cup_i F_i) \leq \sum_{i=1}^{s} \mathbb{P}(F_i) \leq s\delta \, .$$

Further, $\mathcal{B}$ *is correct on all of its calls, if and only if it does not fail. Hence,*

$$\mathbb{P}(\mathcal{B} \textit{ is correct on all of its calls}) = 1 - \mathbb{P}(F) \geq 1 - s\delta \, .$$

Note*: The argument presented here is very commonly used. Oftentimes the situation is so that we can easily prove that an "algorithm" (calculation, etc.) works correctly unless* some *failure event holds. If we collect these failure events into the list* $\{F_i\}_i$, *we can then argue that outside of* $F = \cup_i F_i$ *the algorithm works as expected. Then one is left with controlling the probability of* F *which is done by just noting that* $\mathbb{P}(F) \leq \sum_i \mathbb{P}(F_i)$. *Since the algorithm may also succeed (by errors canceling, etc.) even on* F, *the event when the algorithm actually fails,* F^*, *could be a subset of* F*:* $F^* \subset F$ *may happen. But this does not matter,* $\mathbb{P}(F^*) \leq \mathbb{P}(F) \leq \sum_i \mathbb{P}(F_i)$ *still holds, so to control failure, it suffices to control the probabilities of* F_i, *the individual failure events. The bound* $\mathbb{P}(F) \leq \sum_i \mathbb{P}(F_i)$ *is called the* ***union bound****.* □

---

3 Basic calculus and such

Question 11 For $x \in \mathbb{R}^d$ and $p \geq 1$ let $\|x\|_p = (\sum_{i=1}^d |x_i|^p)^{1/p}$. For $p = \infty$, let $\|x\|_\infty = \max_i |x_i|$. Show that for any x, $1 \leq p \leq q \leq \infty$, $\|x\|_p \geq \|x\|_q$.

Total: **5 points**

Solution. *See the following thread* □

Question 12 Let $1 \leq p \leq \infty$. Show that for any $c \in \mathbb{R}$, $x, y \in \mathbb{R}^d$, the following hold:

1. $\|x\|_p \geq 0$ and if $\|x\|_p = 0$ then $x = 0$;

5 points

2. $\|cx\|_p = |c| \, \|x\|_p$;

5 points

3. $\|x + y\|_p \leq \|x\|_p + \|y\|_p$.

5 points

(That is, $\|\cdot\|_p$ is a norm on $\mathbb{R}^d$.)

Total: **15 points**

Solution. *1)The definition immediately gives that $\|x\|_p \geq 0$. Assume now that $\|x\|_p = 0$. If $p = \infty$, this means $|x_1| = \cdots = |x_d| = 0$, which implies that $x = 0$. If $p < \infty$, we have $0 = \|x\|_p^p = \sum_i |x_i|^p \geq |x_j|^p$ for $j \in [d]$ arbitrary. Taking the pth root, we find that $x_j = 0$. Since j was arbitrary, $x = 0$.*
2)

$$\|cx\|_p = \left(\sum_i |cx_i|^p \right)^{1/p} = (|c|^p)^{1/p} \left(\sum_i |x_i|^p \right)^{1/p} = |c| \|x\|_p \tag{3}$$

Note if $p = \infty$ then the $\max_i |cx_i| = |c| \max_i |x_i|$
3) See Minkowski Inequality □

Question 13 Let $\|\cdot\|$ be any norm on the vector-space of $d \times d$ matrices (that is, $\|\cdot\| : \mathbb{R}^{d \times d} \to \mathbb{R}$ satisfies the properties of a norm when we treat $\mathbb{R}^{d \times d}$ as a d^2-dimensional vector space over the reals). Further, assume that $\|\cdot\|$ is submultiplicative: for any $A, B \in \mathbb{R}^{d \times d}$, $\|AB\| \leq \|A\|\|B\|$. Let I be the $d \times d$ identity matrix, $A \in \mathbb{R}^{d \times d}$ arbitrary. Show that $I - A$ is nonsingular when $\sum_{k=0}^{\infty} \|A\|^k < \infty$.
Hint: Perhaps the inverse of $I - A$ is equal to $\sum_{k \geq 0} A^k$? But is this infinite sum even well-defined?

Total: **15 points**

Solution. ***Solution 1:*** *Note that if $\sum_k ||A||^k$ is finite then the spectral radius $\rho(A) < 1$. Thus the eigenvalues of $(I - A)$, which are $\{1 - \lambda : \lambda$ is an eigenvalue of $A\}$ have strictly positive real parts, and cannot be zero. Hence, the matrix is invertible.*

Solution 2: *Let's just follow the hint. Let $S_n = \sum_{k=0}^{n} A^k$. We claim that $(S_n)_n$ is convergent. That is, for some $S \in \mathbb{R}^{d \times d}$ for every $(i, j) \in [d] \times [d]$, $S_n(i, j) \to S(i, j)$, where we use $X(i, j)$ to denote the (i, j)th entry of a matrix X.*

To show that S_n is convergent it suffices to show that S_n is convergent in (say) the Frobenius norm for if we find S such that $\|S - S_n\|_F \to 0$ as $n \to \infty$ then

$$|S(i, j) - S_n(i, j)|^2 \leq \|S - S_n\|_F^2 \to 0 as \; n \to \infty$$

and thus $(S_n(i, j))_n$ must also converge to S.

Now note that $\mathbb{R}^{d \times d}$ inherits its completeness from the reals. Thus, to show that S_n converges, it suffices to show that for any (i, j), $\limsup_{n \to \infty} \sup_{m \geq n} |S_m(i, j) - S_n(i, j)|$ (i.e., that it is Cauchy). For this, similarly to the above, $|S_m(i, j) - S_n(i, j)|^2 \leq \|S_m - S_n\|_F^2$ and hence

$$|S_m(i, j) - S_n(i, j)| \leq \|S_m - S_n\|_F \leq c\|S_m - S_n\|,$$

where in the last inequality $c > 0$ is a d-dependent constant so that $\|X\|_F \leq c\|X\|$ holds for all $X \in \mathbb{R}^{d \times d}$. This constant exist because in finite dimensional vector spaces all norms are equivalent. Now, $S_m - S_n =$

$\sum_{k=n+1}^{m} A^k$. *Hence,* $\|S_m - S_n\| \leq \sum_{k=n+1}^{m} \|A^k\|$ *by the triangle inequality. Since* $\|\cdot\|$ *is submulitplicative,* $\|A^k\| \leq \|A\|^k$. *Hence,* $\|S_m - S_n\| \leq \sum_{k=n+1}^{m} \|A\|^k$. *Since, by assumption* $\sum_k \|A\|^k$ *converges,* $\limsup_{n\to\infty} \sum_{m\geq n} \sum_{k=n+1}^{m} \|A\|^k = 0$. *Putting things together, we see that*

$$\begin{aligned}\limsup_{n\to\infty} \sup_{m\geq n} |S_m(i,j) - S_n(i,j)| &\leq \limsup_{n\to\infty} \sup_{m\geq n} \|S_m - S_n\|_F \\ &\leq c \limsup_{n\to\infty} \sup_{m\geq n} \|S_m - S_n\| = 0\,.\end{aligned}$$

Thus, S_n *converges. Let its limit be* S. *We calculate*

$$S_n(I-A) = (I + A + \cdots + A^n - (A + A^2 + \cdots + A^{n+1})) = I - A^{n+1}\,.$$

Taking the limit of $n \to \infty$ *of both sides and noting that* $\lim_{n\to\infty}(S_n(I-A)) = (\lim_{n\to\infty} S_n)(I-A) = S(I-A)$, *we get*

$$S(I-A) = I\,.$$

Hence, $I - A$, *being square, is invertible and it's inverse is* S. □

Question 14 Let now $\|\cdot\|$ be a norm on $\mathbb{R}^d$. Call $s : \mathbb{R}^{d\times d} \to \mathbb{R}$ the "norm induced by $\|\cdot\|$" when for any $A \in \mathbb{R}^{d\times d}$, $s(A) = \sup_{x\in\mathbb{R}^d, \|x\|=1} \|Ax\|$. Show that s is a submultiplicative norm on $\mathbb{R}^{d\times d}$, i.e., it satisfies the premise of the previous question.

Total: **5 points**

Solution. *First note that* $s(A) = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}$. *This is because*

$$\left\{ \frac{\|Ax\|}{\|x\|} \,:\, x \neq 0 \right\} = \left\{ \|Av\| \,:\, v \in \mathbb{R}^d, \|v\| \neq 1 \right\}\,.$$

Now, the statement is trivial if either $A = 0$ *or* $B = 0$ *because in this case* $s(AB) = 0$. *Otherwise,*

$$\begin{aligned} s(AB) &= \sup_{x\neq 0} \frac{\|ABx\|}{\|x\|} = \sup_{x\neq 0, Bx\neq 0} \frac{\|ABx\|}{\|Bx\|}\frac{\|Bx\|}{\|x\|} \\ &\leq \sup_{Bx\neq 0} \sup_{y\neq 0} \frac{\|ABx\|}{\|Bx\|}\frac{\|By\|}{\|y\|} \\ &= \left(\sup_{Bx\neq 0} \frac{\|ABx\|}{\|Bx\|}\right)\left(\sup_{y\neq 0} \frac{\|By\|}{\|y\|}\right) \\ &= \left(\sup_{v\neq 0} \frac{\|Av\|}{\|v\|}\right)\left(\sup_{y\neq 0} \frac{\|By\|}{\|y\|}\right) \\ &= s(A)s(B)\,, \end{aligned}$$

where the second equality (where we add $Bx \neq 0$ *to the condition on* x *in the supremum) follows because*

$$\left\{ \frac{\|ABx\|}{\|x\|} \,:\, x \neq 0 \right\} = \{0\} \cup \left\{ \frac{\|ABx\|}{\|Bx\|}\frac{\|Bx\|}{\|x\|} \,:\, x \neq 0, Bx \neq 0 \right\}\,.$$

and thus the suprema of the set on the left-hand side and the second set on the right-hand side agree must agree (the second set on the right-hand side is not empty because $B \neq 0$ *and in particular it has nonnegative elements).* □

Question 15 Calculate a closed form expression for $s(A)$ with $A \in \mathbb{R}^{d\times d}$ and s the norm induced on $\mathbb{R}^{d\times d}$ by the vector-space norm $\|\cdot\|_\infty$.

Total: **5 points**

Solution. *We show that* $s(A) = \max_{i\in[d]} \sum_{j=1}^{d} |A_{i,j}|$.

Let x_i *be defined by* $x_{ij} = \operatorname{sgn}(a_{ij})$ *(when* $a_{ij} = 0$ *use either* $+1$ *or* -1*). Note that* $\|Ax_k\|_\infty = \max_i |\sum_j a_{ij}x_{kj}| \geq |\sum_j a_{kj}x_{kj}| = |\sum_j |a_{kj}|| = \sum_j |a_{kj}|$. *We thus have*

$$s(A) \geq \max_k \|Ax_k\|_\infty \geq \max_k \sum_j |a_{kj}| .$$

Furthermore, $\|Ax\|_\infty = \max_k |\sum_j a_{kj}x_j| \leq \max_k \sum_j |a_{kj}|\,|x_j| \leq (\max_k \sum_j |a_{kj}|)\|x\|_\infty$. *Hence,*

$$s(A) \leq \sup_{x:\|x\|_\infty=1} \max_k \sum_j |a_{kj}| \max_k \sum_j |a_{kj}| .$$

finishing the proof. □

Question 16 Let $T : U \to V$ be a map from a normed vector space U over the reals to another normed vector space V over the reals. Possibly $U \neq V$, so the norms will also be different, but to reduce clutter, we just use $\|\cdot\|$ to denote the norms on all the vector spaces as from the context it will always be clear which norm we are concerned with. Fix $L \geq 0$. The map T is called L-Lipschitz if $\|T(u) - T(u')\| \leq L\|u - u'\|$.

1. Why can we write $T(u) - T(u')$? What set does the result of this operation belong to? Why?

2 points

2. Let $T : U \to V$ be L-Lipschitz, $S : V \to W$ be M-Lipschitz (W is also a normed vector space and $M \geq 0$). Prove that $S \circ T$ is LM-Lipschitz. Here, $S \circ T$ is the composition of T and S. Thus, $S \circ T : U \to W$ and in particular $(S \circ T)(u) = S(T(u))$.

8 points

Total: **10 points**

Solution. *1) By definition of a vector space since both* $T(u)$ *and* $T(u')$ *both belong to the vector space* V *so too does their difference* $T(u) - T(u')$
2) For an arbitrary $u, u' \in U$,

$$\|(S \circ T)(u) - (S \circ T)(u')\| = \|S(T(u)) - S(T(u'))\| \leq M\|T(u) - T(u')\| \leq ML\|u - u'\|$$

Thus $S \circ T$ *is* LM *or* ML *Lipschitz.* □

Total for all questions: 125