Exercises

Exercises are for fun and edification, please do not submit. But please do try to solve them, we may need ideas from there later in the course (even in this very homework!)

1. Consider the entropy function \( H(x) = -\sum_{i=1}^{N} x_i \ln x_i \) defined over the probability simplex \( \Delta_N = \{ x \mid x_j \geq 0 \ \forall j, \sum_i x_i = 1 \} \).
   
   (a) Compute its gradient \( \nabla H(x) \) at the point \( x \)—recall this is a vector in \( \mathbb{R}^N \) whose \( i \)th coordinate is \( \frac{\partial H(x)}{\partial x_i} \).
   
   (b) Compute the Hessian \( \nabla^2 H(x) \)—this is an \( N \times N \) matrix whose \((i, j)\)th entry is \( \frac{\partial^2 H(x)}{\partial x_i \partial x_j} \). Show that all its eigenvalues are non-positive for \( x \in \Delta_N \), and hence the matrix is negative semi-definite in this region. This shows that the function is concave in \( \Delta_N \).
   
   (c) Show that \( \arg \max_{x \in \Delta_N} H(x) \) is \( x_i = 1/n \). Here we’ll use that for a \emph{convex} function \( f \) is minimized over convex set \( K \) at some point \( x \in K \) if and only if \( \langle \nabla f(x), y - x \rangle \geq 0 \) for all \( y \in K \). (For a \emph{concave} function, recall that \( -f \) is convex.)
   
   (d) Given \( \varepsilon > 0 \) and vector \( L \in \mathbb{R}^N \) you want to \emph{minimize} the \emph{convex} function \( f(x) := \langle L, x \rangle - \frac{1}{\varepsilon} H(x) \) over \( \Delta_N \). Show that this minimum is achieved at
   \[
   x_i = \frac{\exp(-\varepsilon L_i)}{\sum_{j=1}^{N} \exp(-\varepsilon L_j)}.
   \]
   (Hint: use the optimality criterion from the previous part.) Let’s just discuss this a bit more: the point \( z \in \mathbb{R}^N \) having \( z_i = \exp(-1 - \varepsilon L_i) \) achieves \( \nabla f(z) = 0 \) and has non-negative coordinates, but is not in \( \Delta_N \) because \( \sum_i z_i \neq 1 \). Rescaling by \( \|z\|_1 \), i.e., defining \( x = \frac{z}{\|z\|_1} \) ensures takes care of that constraint.

2. Suppose that instead of choosing a single expert at each time, we are required to pick a subset of \( k \) experts at each time. We suffer the losses of all those experts. And we want to compare the total (expected) losses we suffer to the loss suffered by best subset of size \( k \) of the experts.

   Give an algorithm that achieves the following: suppose \( S_t \) is the set of size \( k \) that we pick at timestep \( t \), then for any set \( S^* \) of size \( k \),
   \[
   \mathbb{E} \left[ \sum_{t \leq T} \sum_{i \in S_t} \ell_t^i \right] \leq (1 + \varepsilon) \sum_{t \leq T} \sum_{i \in S^*} \ell_t^i + \frac{O(k \log N)}{\varepsilon}.
   \]

   Hint: your algorithm does not need to be efficient for this problem, and can take time proportional to \( \binom{N}{k} \).
Problems

Solve any two of the problems.

1. (Made A Mistakes Recently? No Worries, We All Do.) Here is a variation on the deterministic Weighted-Majority algorithm, designed to make it more adaptive.
   
   (a) Each expert begins with weight 1 (as before).
   (b) We predict the result of a weighted-majority vote of the experts (as before).
   (c) If an expert makes a mistake, we penalize it by dividing its weight by 2, but only if its weight was at least 1/4 of the average weight of experts.

   Prove that in any contiguous block of trials (e.g., the 51st day through the 77th day), the number of mistakes made by the algorithm is at most $O(m + \log n)$, where $m$ is the number of mistakes made by the best expert in that block, and $n$ is the total number of experts.

2. (Linear Systems and GD.) Given an $n \times n$ symmetric matrix $A$ and an $n \times 1$ vector $b$, our goal is to solve the equation $Ax = b$ to high accuracy. We will use gradient descent to solve this problem quickly given some assumptions about $A$; see the lecture notes for background on gradient descent. (The analysis here is independent of the one from lecture, but you should be comfortable with the ideas there.)

   Recall from linear algebra that every symmetric $n \times n$ matrix $A$ can be written as $V \Lambda V^T$, where $V$ is an $n \times n$ matrix whose columns are the eigenvectors of $A$, and $\Lambda$ is a diagonal matrix whose entries are the eigenvalues of $A$. Recall that for $x \in \mathbb{R}^n$, $||x||^2 = \sum_{i=1}^{n} x_i^2$.

   (a) Consider the function $f(x) = \frac{1}{2}||Ax - b||^2$. Prove that $f$ is convex and that its gradient is $\nabla f(x) = A(x - b)$. (Hint: if $g(y)$ is a convex function, what about $f(x) = g(Ax - b)$?)
   (b) Suppose $x^* = \arg\min_x \frac{1}{2}||Ax - b||^2$. State why $A^2x^* = Ab$.
   (c) Suppose we set $x^{(0)} = 0^n$, and
      
      $x^{(t+1)} \leftarrow x^{(t)} - \nabla f(x^{(t)})$.
      
      Argue for any $i \geq 0$, $A(x^{(i+1)} - x^*) = (I - A^2)(A(x^{(i)} - x^*))$.
   (d) Argue that $||Ax^{(t)} - b||^2 = ||A(x^{(t)} - x^*)||^2 + ||Ax^* - b||^2$. Hint: for $x, y \in \mathbb{R}^n$, if $\langle x, y \rangle = 0$, then $||x + y||^2 = ||x||^2 + ||y||^2$. You may also find part (b) useful.

   The above parts should all be proven for any symmetric matrix $A$, regardless of whether it is invertible or not.

   For the next parts, you may find the following statements helpful: (1) for a symmetric matrix $B$ and a vector $y$, $\|By\| \leq \max(|\lambda_{\max}|, |\lambda_{\min}|) \cdot \|y\|$ where $\lambda_{\max}$ is the maximum eigenvalue of $B$ and $\lambda_{\min}$ is the minimum eigenvalue of $B$, and (2) for a symmetric matrix $B$, the eigenvalues of $I - B$ are in the range $[1 - \lambda_{\max}, 1 - \lambda_{\min}]$. Please try to prove these facts about eigenvalues yourself for practice, though you will not need to prove these to us in the oral presentation.

   For the following parts, assume that all eigenvalues of $A$ are in the range $[0.9, 1.1]$; such an $A$ is called well-conditioned. (Although you need not use this fact: is such a matrix invertible, i.e., does $A^{-1}$ exist?)
   (e) Show that $\|A(x^{(i+1)} - x^*)\| \leq \frac{1}{2}\|A(x^{(i)} - x^*)\|$.
(f) Prove that there exists a constant $c$ such that for any $\epsilon \in (0, 1)$, if $t \geq c \log(1/\epsilon)$, then
\[ \| A(x^{(t)} - x^*) \|^2 \leq \epsilon \| b \|^2. \]

(g) Assuming that $A$ has $m$ non-zero entries, what is the overall running time of the algorithm for outputting an $x^{(t)}$ for which $\| Ax^{(t)} - b \|^2 \leq \| Ax^* - b \|^2 + \epsilon \| b \|^2$? Give an answers in terms of $m, n, \epsilon$. Assume the non-zero entries of $A$ are represented in such a way so that for any vector $z$, $A \cdot z$ can be computed in $O(m)$ time.

3. (Data Driven Algorithms.) Consider a setting of the experts problem, where at each timestep you pick one expert (call it $I_t$) and get loss $\ell_{I_t}$ as usual. The new part is that when you change experts, you suffer a moving cost of $k \geq 1$. So your expected loss is
\[ \sum_{t=1}^{T} \mathbb{E} \left[ \ell_{I_t}^t + k \cdot 1_{(I_t \neq I_{t-1})} \right]. \]

The expectation is over the randomness of your algorithm’s coin tosses.

How to choose $I_t$ now that we have moving costs? We do this in two steps. We show (a) Hedge maintains distributions so that $p^{t-1}$ and $p^t$ that are “close” to each other for all $t$, and (b) we can sample from these distributions in a way that $I_{t-1}$ and $I_t$ are likely to agree.

(a) Prove that Hedge (with learning rate $\varepsilon$) ensures that
\[ \| p^t - p^{t+1} \|_1 := \sum_{i=1}^{N} |p^t_i - p^{t+1}_i| \leq O(\varepsilon)(p^t, \ell^t). \]

You may use $\varepsilon$ is small, say $\leq 1/4$ if you need to.

(b) How to choose $I_t$? Here’s one way to do “correlated” rounding. At time 1, recall that $p^1_i = 1/N$ for all $i$. So pick $I_1$ to be an integer in $\{1, 2, \ldots, N\}$, each with probability $1/N$. Now at time $t > 1$, fix some way to redistribute the probability values in $p^t_i$ to $p^{t+1}_i$ that maximizes the amount staying at the same expert: i.e., fix some “redistribution” values $x_{ij} \geq 0$ such that $x_{ii} = \min(p^t_i, p^{t+1}_i)$ for all $i$, and $\sum_j x_{ij} = p^t_i$ and $\sum_i x_{ij} = p^{t+1}_j$. Having chosen this $x$, if $I_t = i$, then pick $I_{t+1}$ to be $j$ with probability $x_{ij}/p^t_i$.

Choose a random value $\alpha$ uniformly from the interval $[0, 1]$ at the very beginning. Now at each time $t$, look for the smallest index $i \in \{1, 2, \ldots, N\}$ such that $\sum_{j \leq i} p^t_j \geq \alpha$. Define $I_t$ to be this index $i$.

i. Show that for any index $i$ and $t$, we have $\Pr[I_t = i] = p^t_i$. So we’re choosing experts with the correct probability.

ii. Moreover, show that for each $t$,
\[ \Pr[I_t \neq I_{t+1}] \leq \| p^t - p^{t+1} \|_1. \]

Hence infer that the expected number of change-of-experts is at most $2\varepsilon \sum_i (p^t_i, \ell^t)$.

(c) Now fix $\delta \in (0, 1)$, say. Show that running Hedge with parameter $\varepsilon = \delta/(2k)$ ensures that the expected cost in this new model incurred these choices is at most
\[ (1 + O(\delta)) \min_i \left( \sum_i \ell^t_i \right) + O(k \log N/\delta). \]

I.e., almost the same guarantee as for the standard experts, except for the extra loss in the additive term.
You may use the claim from lecture: for any instance of the experts problem, where the losses \( \ell^t \in [0,1]^N \), the Hedge algorithm with parameter \( \varepsilon \) maintains \( p^t \in \Delta_N \) such that

\[
\sum_{t} \langle p^t, \ell^t \rangle \leq (1 + \varepsilon) \min_{i} \left( \sum_{t} \ell^t_i \right) + O(\log N) \varepsilon.
\]

Optional Context for the Problem: (If you are interested in how to use it.) You are building an OS, and have \( N \) different algorithms \( A_1, A_2, \ldots, A_N \) that you can use for paging. Some of these are good in some situations, some in other situations. You want to combine them to get a “meta-algorithm” \( A \) that actually decides which pages to evict at each time. Your goal: the number of evictions done by the meta-algorithm on your actual request sequence is (close to) the number done by the best of these algorithms on that input. And you have to combine these algorithms in an online fashion.

You first do the following: you maintain a simulation of all the algorithms “in your mind”. Let \( C_{t-1}(A_i) \) be the contents of the cache of algorithm \( A_i \) after \( r_1, r_2, \ldots, r_{t-1} \). Now when you get a new request \( r_t \) you feed it to each of the algorithms, and they decide which page of their own cache \( C_{t-1}(A_i) \) to evict (if any) to get their new cache \( C_t(A_i) \). If algorithm \( A_i \) evicts a page in its cache at this step, set \( \ell^t_i = 1 \) else \( \ell^t_i = 0 \). Feeding this loss vector \( \ell^t \) to our Hedge algorithm for the experts problem gives us the next probability distribution \( p^{t+1} \). Note: this simulation is in your mind.

Which pages are actually the cache of your OS at each time? Inductively, your actual cache after time \( t - 1 \) will agree with the simulated cache of one of these algorithms, say algorithm number \( I_{t-1} \)'s cache \( C_{t-1}(A_{I_{t-1}}) \). Now given the probability distribution \( p^t \), you choose an algorithm \( I_t \in \{1,2,\ldots,N\} \) in a way we describe soon. You then ensure that your real cache now agrees with the simulated cache \( C_t(A_{I_t}) \). So if \( I_{t-1} = I_t \), you will have to only change at most a single page, otherwise you may have to change all \( k \) pages in the cache (which is a very expensive operation). So you want to ensure that \( I_{t-1} = I_t \) very often, and yet the number of evictions is close to the best.