I will be posting 2-3 problems every week (so I presume there will be around 20 problems in all). You have to submit answers to 10 problems out of these. There will be 3 dates on Canvas where you will be submitting 3, 4, and 3 answers respectively. You are of course free and indeed encouraged to try all these problems. Please submit those problems which you think you have solved it to your satisfaction, and give clear and concise answers.

Problem 1.

In the experts problem, prove that no deterministic algorithm can do better than a 2-factor in the number of mistakes than the best expert. That is, for any deterministic algorithm $A$, any parameter $\delta$ and any $T$, there is a setting of the experts problem such that the best expert makes $\leq \delta T$ mistakes, while $A$ makes $\geq 2\delta T$ mistakes.

Problem 2.

Suppose we changed the MWU algorithm as follows: after observing the losses $\ell_i(t)$ at time $t$, instead of updating the weights as $w_i(t+1) = w_i(t) \cdot (1 - \eta \ell_i(t))$ suppose you updated as $w_i(t+1) = w_i(t) \cdot e^{-\eta \ell_i(t)}$

Analyze this algorithm and figure out the error term. Which one is better and under which circumstances?

Problem 3.

In class we saw that the average regret of the MWU algorithm after $T$ time steps was $O\left(\sqrt{\ln m/T}\right)$. Show that you can’t get better. That is, show for any algorithm (even randomized), the average regret has to be $\Omega\left(\sqrt{\ln m/T}\right)$. To begin with, give an $\Omega(1/\sqrt{T})$ bound for 2 experts.

Hint: The following fact may be useful: if $X_1, \ldots, X_n$ are $n$ independent, iid random variables taking values in $\{-1, +1\}$ with probability 1/2 each, and if $S = \left|\sum_{i=1}^n X_i\right|$, then $\text{Exp}[S] = \Theta(\sqrt{n})$.

Problem 4.

In class when we wanted the average regret to be $O(\sqrt{\ln m/T})$, we set $\eta$ to be something which depended on $T$. What is you didn’t know $T$? That is, the online decision could stop at any unknown time $T$ and you would still like the average regret to be $O(\sqrt{\ln m/T})$ (for all $T$). How would you modify the algorithm?

Hint: Keep a conservative “guess” of $T$ and set $\eta$ likewise; if the actual $T$ is more than your guess, then bump $\eta$ down.

Problem 5.

In class, we saw how to approximate solve the LP

$$\min \sum_{j=1}^n c_j x_j : Ax \geq b; \quad 0 \leq x_j \leq 1$$

The width parameter was defined to be $\rho := \max_{i=1}^m |a_i^T x - b_i|$ for any $x$ returned by the oracle. We showed that for any $\epsilon > 0$, the MWU algorithm can be used to return an $\bar{x}$ such that for every constraint $i \in [m]$, we get $a_i^T \bar{x} \geq b_i - \epsilon$. The number of iterations was $O\left(\frac{\rho^2 \ln m}{\epsilon^2}\right)$. 

1
In this exercise suppose \( A, b \geq 0 \), that is, all entries of the constraints are non-negative. Then, by dividing the \( i \)th row by \( b_i \) we can assume every \( b_i = 1 \). Prove that the MWU algorithm actually needs to only run \( O \left( \frac{\rho \ln m}{\epsilon^2} \right) \) many iterations.

Hint: Recall the loss function we defined in class. Instead of just using \(|\ell_i(t)| \leq 1\), perhaps use that \(|\ell_i(t)| \leq |\ell_i(t)|\).

Problem 6.

For each of these functions \( f : \mathbb{R}^n \to \mathbb{R} \) below, calculate \( \nabla f(x) \).

\[
a. \quad f(x) = b^T Ax \text{ for some } b \in \mathbb{R}^m \text{ and some } m \times n \text{ matrix } A.
\]

\[
b. \quad f(x) = \frac{1}{2} x^T Q x \text{ for some } n \times n \text{ matrix } Q.
\]

\[
c. \quad f(x) = \|Ax - b\|_2^2 \text{ where } A \text{ is an } m \times n \text{ matrix, and } b \in \mathbb{R}^m.
\]

Problem 7.

Prove the following about convex functions.

\[
a. \quad \text{The function } x \mapsto \ln x \text{ is concave (that is, } -\ln x \text{ is convex). Use the definition of convexity to deduce the AM-GM inequality, that is, for any } x_1, \ldots, x_m > 0,
\]

\[
\left( \prod_{i=1}^{m} x_i \right)^{1/m} \leq \frac{\sum_{i=1}^{m} x_i}{m}
\]

\[
b. \quad \text{Given a collection } (a_t, b_t) \text{ for } t = 1, 2, \ldots, T, \text{ where each } a_t \in \mathbb{R}^n \text{ and } b_t \in \mathbb{R}, \text{ prove that}
\]

\[
f(x) := \max_{t=1}^{T} (a_t x + b_t)
\]

is convex. Use this to deduce that the following function, \( x \mapsto x^{[\ell]} \) where the RHS is the sum of the \( \ell \) largest entries of \( x \) is a convex function.

\[
c. \quad \text{Prove that if } h : \mathbb{R} \to \mathbb{R} \text{ is convex and increasing, and } g : \mathbb{R}^n \to \mathbb{R} \text{ is a convex function, then}
\]

\[
f(x) := h(g(x)) \text{ is a convex function.}
\]

\[
d. (\text{Extra Credit})
\]

Prove that

\[
f(x) := \log \left( \sum_{i=1}^{n} e^{x_i} \right)
\]

is convex.

Hint: Use the Hessian

Problem 8.

a. Consider unconstrained convex optimization \( \min_{x \in \mathbb{R}^n} f(x) \) where \( f : \mathbb{R}^n \to \mathbb{R} \) is a convex, differentiable function. Let \( x_* \) be global minimizer of \( f \). Prove that \( \nabla f(x_*) \) is the 0-vector. In particular, given any \( x \) with \( \nabla f(x) \neq 0 \), demonstrate a vector \( y \) with \( f(y) < f(x) \).

b. Now consider constrained convex optimization \( \min_{x \in S} f(x) \) where \( S \) is a convex set. Again, let \( x_* \in S \) be a global minimizer. Is \( \nabla f(x_*) = 0 \) a necessary condition? What is a necessary condition you can assert for \( \nabla f(x_*) \) and points in \( S \). Prove your statement.
**Problem 9.**

In class, we showed that for any $\varepsilon > 0$, if we set $\eta = \varepsilon^2 / \rho$ and we run for $T = D^2 \rho^2 / \varepsilon^2$ rounds, then we can find a point $x_t$ with $f(x_t) \leq f(x_*) + \varepsilon$. Here $D := \|x_1 - x_*\|_2$ and $\|\nabla f(x)\|_2 \leq \rho$ for all $x$.

In this exercise, you take a “time-dependent-parameter-independent” step size. That is, $\eta_t$ is not a fixed $\eta$ but is defined to be $\eta_t := \frac{1}{\sqrt{t}}$. Formally, we start at point $x_1$ and then at time $t$, we take the step

$$x_{t+1} = x_t - \frac{1}{\sqrt{t}} \nabla f(x_t)$$

Analyze the above algorithm. Your goal is to find out given $\varepsilon$ how long do we need to run to get some $f(x_t) \leq f(x_*) + \varepsilon$. The running time will depend on $D, \rho$; your goal is to find out if it is better or worse than the fixed parameter-dependent step size.

**Problem 10.**

Given a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, prove that $\|\nabla f(x)\|_2 \leq \rho$ for all $x$ if and only if $|f(x) - f(y)| \leq \rho \cdot ||x - y||_2$ for all $x, y$.

**Problem 11.**

Consider $f : \mathbb{R}^n \to \mathbb{R}$ which is $L$-smooth and convex. In class, we proved that vanilla gradient descent with $\eta = 1/L$ if run for $T := LD^2 / \varepsilon$ iterations, where $D := ||x_1 - x_*||_2$, one gets a point $f(x) \leq f(x_*) + \varepsilon$.

In this exercise, you are supposed to extend it to the case of constrained convex optimization with projected gradient descent. In particular, the algorithm is

$$z_{t+1} = x_t - \eta \nabla f(x_t); \quad x_{t+1} = \Pi_S(z_{t+1}) = \arg\min_{v \in S} ||z_{t+1} - v||_2$$

Prove that the same convergence bound holds. Indeed, there is only one extra line in the whole analysis. Hint: Recall the fact about projection: if $u \in S, v \notin S$, and $p := \Pi_S(v)$ is the projection of $v$ onto $S$, then $(v - p)^T (v - u) \leq 0$. That is, the line joining $v$ and $p$, and the line joining $v$ and $u$ is obtuse.

**Problem 12.**

In this exercise, you will analyze gradient descent assuming only strong convexity of $f$. Recall that $f : \mathbb{R}^n \to \mathbb{R}$ is $\ell$-strongly convex iff for all $x, y$ we have

$$f(y) \geq f(x) + (y - x)^T \nabla f(x) + \frac{\ell}{2} \cdot ||y - x||^2_2$$

Assume that $||\nabla f(x)||_2 \leq \rho$ for all $x$ (note that $f$ may not be smooth).

Consider gradient descent with time-dependent step-size $\eta_t := \frac{1}{\sqrt{t}}$. Prove that if we run this for $T$ iterations, and if $\text{err}(t) := f(x_t) - f(x_*)$, then

$$\frac{1}{T} \sum_{t=1}^{T} \text{err}(t) \leq \frac{\rho^2 \ln T}{\ell T}$$

In particular, for any $\varepsilon > 0$ this shows that running for $T = \frac{2\rho^2}{\ell \varepsilon} \cdot \left(\frac{1}{\varepsilon} \ln \frac{4\rho^2}{\ell \varepsilon}\right)$ rounds would give error $\leq \varepsilon$.

Hint: Use the fact that for strongly convex functions we have a better upper bound on $\text{err}(t)$. In particular,

$$\text{err}(t) \leq (x_t - x_*)^T \nabla f(x_t) - \frac{\ell}{2} D^2_t$$

Now use the fact that the $\eta_t$’s are decaying to show that when you sum up the $\text{err}(t)$’s cancellations occur to give you what you need. It may be useful to recall the fact that $1 + 1/2 + 1/3 + 1/4 + 1/5 + \cdot + 1/k \ln k$.

**Extra Credit:** In fact, you can also get a $O(1/\varepsilon)$-convergence rate (assuming $\ell, \rho$ are constant) with some slight modification. If you have had fun solving the problem above, then maybe you can try this too.
Exercise 1. Consider the online decision making problem done in class where \( g_i(t) \) is the gain/profit obtained if we play action \( i \) at time \( t \). Design and analyze the MWU algorithm for maximizing total expected gain as compared to the best fixed action in hindsight. This will mimic the analysis done in class; please try doing this without looking at notes, etc.

Exercise 2. Show an example of a setting where the MWU algorithm actually does better (gets less loss or more gain) than the fixed action in hindsight.

Exercise 3. Implement the approximate LP solver in your favorite language. Generate a “random” linear program by taking entries of \( A, b, c \) at random. Does your implementation work “fast”? Do you see a dependence on the width?

Exercise 4. Run the approximate LP solver done in class on the vertex cover LP:

\[
\min \sum_{v \in V} c(v)x_v : \forall (u, v) \in E, \ x_u + x_v \geq 1, \ 0 \leq x_u \leq 1
\]

What is the oracle? What is the width? What is the final algorithm actually doing?