

1. Alternative Multiplicative Weights

In practice, the exponential function is (relatively) expensive to compute, and any computation is only an approximation of the true value. Suppose we make the following change to the algorithm: instead of updating $w_i^{(t+1)} \leftarrow w_i^{(t)} \exp(\epsilon g_i^{(t)})$ on each step, we update $w_i^{(t+1)} \leftarrow w_i^{(t)} (1 + \epsilon g_i^{(t)})$. The goal of this problem is to show that the multiplicative weights analysis still goes through. For context, please refer to the Lecture 16 notes.

As usual, assume that each gain vector $g^{(t)}$ is bounded by $[-1, 1]^N$. Let $\Phi(t) = \sum_i w_i^{(t)}$ from the standard multiplicative weights analysis.

- (a) Show that $\Phi^{(t+1)} \leq \Phi^{(t)} \exp(\epsilon \langle p^{(t)}, g^{(t)} \rangle)$ for each round t .
- (b) Assume that $\epsilon \leq 1/2$, and let T be the final round. Show that $w_i^{(T+1)} \geq \exp(\epsilon \sum_{t=1}^T g_i^{(t)} - \epsilon^2 T)$ for each expert i . You may use the fact that $1 + x \geq e^{x-x^2}$ for all $x \in [-1/2, 1/2]$.
- (c) Conclude that $\sum_{t=1}^T \langle p^{(t)}, g^{(t)} \rangle \geq \sum_{t=1}^T g_i^{(t)} - \epsilon T - \frac{\ln N}{\epsilon}$ for each expert i .

2. Finding the Smallest Enclosing Ball

You're given $P = \{p_1, p_2, \dots, p_m\}$, a set of m points in \mathbb{R}^n , where $\|p_i\| \leq 1$ for all i . Your goal is to find the *smallest enclosing ball* of P . The smallest enclosing ball can be represented by its center point $c^* \in \mathbb{R}^n$.

Please use the following:

Constrained gradient descent framework Given a constraint area K , the idea is that when the update step take us outside K , we just “project back into K ”. Specifically, start with some point x_0 . At each step $t = 1, 2, \dots, T - 1$, set

$$y_{t+1} \leftarrow x_t - \eta_t \cdot \nabla f(x_t) \tag{1}$$

$$x_{t+1} \leftarrow \text{the point in } K \text{ closest to } y_{t+1} \tag{2}$$

and return $\hat{x} = \frac{1}{T} \sum_{t=0}^{T-1} x_t$, where η_t is the learning rate and $\nabla f(x_t)$ is the gradient at x_t .

Theorem (Basic Gradient Descent). For any (differentiable) convex function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and any starting point \mathbf{x}_0 , if we set $T = \left(\frac{GD}{\epsilon}\right)^2$ and $\eta_t = \eta := \frac{\epsilon}{G^2}$, then

$$f(\bar{\mathbf{x}}) - f(\mathbf{x}^*) \leq \epsilon.$$

Here $D := \|\mathbf{x}_0 - \mathbf{x}^*\|$ is the distance of the starting point x_0 from the optimal point x^* , and G is the bound on the norm of the gradient at any point $x \in K$, i.e., $\|\nabla f(x)\| \leq G, \forall x \in K$. Remember that G, D depend on both f and \mathbf{x}_0 .

Your algorithm will use gradient descent to approximate the center of the smallest enclosing ball with a ϵ . If the smallest enclosing ball for P is centered at a point $c^* \in \mathbb{R}^n$, then your algorithm will find a center point $\hat{c} \in \mathbb{R}^n$ such that the *radius squared* of the ball centered there that contains all the points of P and is at most ϵ bigger than that of the optimum center point c^* .

To make use of this framework, you will:

1. Specify a starting point $c_0 \in \mathbb{R}^n$.
2. Specify a convex function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ that is to be minimized. (Don't worry if your function is not differentiable everywhere. It's okay if it's not differentiable on a set of measure zero.)
3. Specify a convex set K .
4. Compute the bounds D and G .
5. Specify a value of η .
6. Specify a number of iterations T .

3. Solving Linear Systems Using Gradient Descent

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 that for $x \in \mathbb{R}^n$, $\|x\|_2^2 = \sum_{i=1}^n x_i^2$.

Let A be an arbitrary *symmetric* $n \times n$ matrix.

- (a) Consider the function $f(x) = \frac{1}{2}\|Ax - b\|_2^2$. Prove that f is convex and that its gradient is $\nabla f(x) = A(Ax - b)$. (*Hint: if $g(y)$ is a convex function, what about $f(x) = g(Ax - b)$?*)
- (b) Suppose $x^* = \operatorname{argmin}_x \frac{1}{2}\|Ax - b\|_2^2$. State why $A^2x^* = Ab$.
- (c) Suppose we set $x^{(0)} = \mathbf{0}$ to be the zero vector, 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 $\|Ax^{(t)} - b\|_2^2 = \|A(x^{(t)} - x^*)\|_2^2 + \|Ax^* - b\|_2^2$. Hint: for $x, y \in \mathbb{R}^n$, if $\langle x, y \rangle = 0$, then $\|x + y\|_2^2 = \|x\|_2^2 + \|y\|_2^2$.

Recall from linear algebra that A can be written as $V\Lambda V^T$, where V is an $n \times n$ matrix whose columns are the (unit) eigenvectors of A , so that $VV^T = V^T V = I$, and Λ is a diagonal matrix whose entries are the eigenvalues of A .

Moreover, assume that all eigenvalues of A are in the range $[0.9, 1.1]$; such an A is called *well-conditioned*. If you want more intuition for how conditioning can impact the convergence rate of gradient descent, you can read [this excellent post](#).

- (e) Show that $\|A(x^{(i+1)} - x^*)\|_2 \leq \frac{1}{2}\|A(x^{(i)} - x^*)\|_2$. Hint: (1) for a symmetric matrix B and a vector y , $\|By\|_2 \leq \max(|\lambda_{max}|, |\lambda_{min}|) \cdot \|y\|_2$ where λ_{max} is the maximum eigenvalue of B and λ_{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 do not need to prove these in your submission.
- (f) Prove that there exists a constant c such that for any $\epsilon \in (0, 1)$, if $t \geq c \log(1/\epsilon)$, then

$$\|A(x^{(t)} - x^*)\|_2^2 \leq \epsilon \|b\|_2^2.$$

Therefore, gradient descent solves this linear system with a sufficient amount of time.