Closer one step at a time

CS 170 Spring 2024

December 12, 2024

Point Pursuit in Two Dimensions

Consider the following game played between two players, whom we call Alice and Bob.[1] There is a point $p^* = (x^*, y^*)$ on the 2-dimensional plane.

[1] This game is set on the 2-dimensional plane; we will later see that the game and its properties also generalize in any number of dimensions.

Point Pursuit in 2-dimensions

In the t^{th} iteration,

1. Bob proposes a point $p^{(t)} = (x^{(t)}, y^{(t)})$.
2. Alice finds a *separating line* $\ell^{(t)}$ that separates Bob's current point $p^{(t)}$ from the destination p^*. Formally, p^* is on one side of the line $\ell^{(t)}$, but $p^{(t)}$ is at least ϵ-away from the line on the opposite side (see Figure (1)).

Alice loses when she is unable to find a separating line.

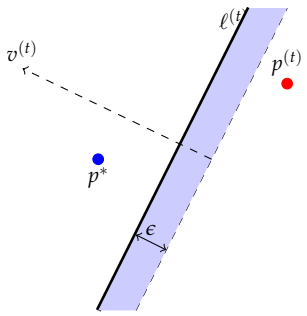

Figure 1: A separating line $\ell^{(t)}$ between Bob's current point $p^{(t)}$ and destination p^*.

Let $v^{(t)}$ denote the unit vector perpendicular to Alice's separating line $\ell^{(t)}$. Consider the following strategy for Bob:

$$p^{(t+1)} = p^{(t)} + \epsilon v^{(t)} \,, \tag{1}$$

in words, Bob moves a distance of ϵ directly towards the separating line $\ell^{(t)}$. With this strategy, we will see that Bob gets closer to the point p^* in each step.

Claim 1. (Bob keeps getting closer) In each iteration, the squared distance of Bob's point $p^{(t)}$ to p^* decreases by ϵ^2

Proof. Consider any iteration t. By shifting and rotating the 2-dimensional plane, we may assume the following (see Figure 2):

- p^* is the origin.
- The separating line is parallel to x-axis, so let's say the line is $y = c$ for some $c \geqslant 0$.

Let us suppose $p^{(t)} = (x, y)$. Moving ϵ distance towards the line, Bob will end up at $p^{(t+1)} = (x, y - \epsilon)$. The squared distances to p^* are

$$\|p^{(t)} - p^*\|^2 = x^2 + y^2 \text{ and } \|p^{(t+1)} - p^*\|^2 = x^2 + (y - \epsilon)^2$$

So the change in the squared distance is,

$$\|p^{(t)} - p^*\|^2 - \|p^{(t+1)} - p^*\|^2 = 2y\epsilon - \epsilon^2 \geqslant \epsilon^2$$

In the last inequality, we used the fact that $y \geqslant \epsilon$, since $p^{(t)}$ is at least ϵ above a horizontal line $\ell^{(t)}$ which itself is above the origin p^*. □

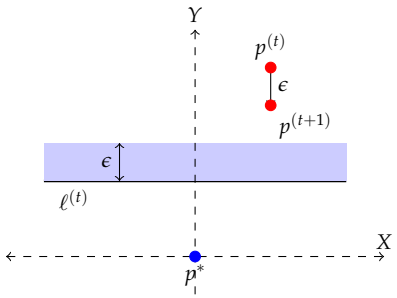

Figure 2: Bob moves ϵ closer to the line. The picture is drawn after rotating and shifting.

Let $p^{(0)}$ denote Bob's location in the first iteration. Since Bob's squared distance keeps dropping by ϵ^2 in every iteration, the game can't go on for too long.

Theorem 2. *If Bob plays strategy given by (1), then Alice fails to find a separating line within* $\left\lceil \frac{\|p^{(0)} - p^*\|^2}{\epsilon^2} \right\rceil$ *iterations.*

Point Pursuit in High Dimensions

The strategy we developed for Bob in two-dimensions is actually successful for the point pursuit game in arbitrary dimensions. Towards seeing this, let us first get comfortable in d-dimensional space, by defining the quantities we need.

Geometry in Any Dimensions

The inner product of two vectors $x, y \in \mathbb{R}^d$ is defined as

$$\langle x, y \rangle = \sum_{i=1}^{d} x_i \cdot y_i.$$

Inner product is bilinear in the arguments, so $\langle x + x', y \rangle = \langle x, y \rangle + \langle x', y \rangle$ and $\langle x, y + y' \rangle = \langle x, y \rangle + \langle x, y' \rangle$. Furthermore, $\langle \lambda x, y \rangle = \lambda \langle x, y \rangle = \langle x, \lambda y \rangle$ for any real number λ. For example, the following calculation makes sense,

$$\begin{aligned} \langle 2a + 3b, 4c + 5d \rangle &= \langle 2a, 4c \rangle + \langle 2a, 5d \rangle + \langle 3b, 4c \rangle + \langle 3b, 5d \rangle \\ &= 8\langle a, c \rangle + 10\langle a, d \rangle + 12\langle b, c \rangle + 15\langle b, d \rangle \end{aligned}$$

The Euclidean length of a vector $x \in \mathbb{R}^d$ is given by,

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_d^2}\,.$$

Observe that

$$\|x\|^2 = \sum_{i=1}^{d} x_i^2 = \langle x, x \rangle$$

A line in 2-dimensions is defined by the set of points $x = (x_1, x_2)$ such that,

$$\langle v, x \rangle - \theta = 0$$

All points on one side of the line satisfy $\langle v, x \rangle - \theta > 0$, while $\langle v, x \rangle - \theta < 0$ on the other side. In 3-dimensions, an equation of the form $\langle v, x \rangle - \theta = 0$ form a plane.Analgously, in higher dimensions, the set of points such that $\langle v, x \rangle - \theta = 0$ form a "*hyperplane*". On two sides of this hyperplane are points wherein $\langle v, x \rangle - \theta > 0$ and $\langle v, x \rangle - \theta < 0$ respectively.

A HYPERPLANE is the set of points $x \in \mathbb{R}^d$ that satisfy an equation of the form,

$$\langle v, x \rangle - \theta = 0$$

Here the vector $v \in \mathbb{R}^d$ is the normal vector to the hyperplane, and can be normalized so that its length is 1.

We now have all the ingredients to talk about the point pursuit game in d dimensions.

Point Pursuit in d-dimensions

In the t^{th} iteration,

1. Bob proposes a point $p^{(t)} \in \mathbb{R}^d$.
2. Alice finds a *separating hyperplane* that separates Bob's current point $p^{(t)}$ from the destination p^*. Formally, Alice finds $v^{(t)}, \theta^{(t)}$ such that v is a unit vector and,

   $$\langle v^{(t)}, p^* \rangle - \theta^{(t)} \geqslant 0$$

   while

   $$\langle v^{(t)}, p^{(t)} \rangle - \theta^{(t)} \leqslant -\epsilon.$$

   Notice that the above pair of inequalities also implies,

   $$\langle v^{(t)}, p^{(t)} - p^{(*)} \rangle \leqslant -\epsilon \tag{2}$$

Alice loses when she is unable to find a separating line.

A SUCCESFUL STRATEGY for Bob is the following:

$$p^{(t+1)} = p^{(t)} + \eta v^{(t)} \ , \tag{3}$$

In particular, just like 2-dimensions, we have,

Theorem 3. *If Bob plays strategy given by* (3) *with* $\eta = \epsilon$*, then Alice fails to find a separating line within* $\left\lceil \frac{\left\| p^{(0)} - p^* \right\|^2}{\epsilon^2} \right\rceil$ *iterations.*

The reasoning is analogous to 2-dimensions as well. We will argue that Bob's squared distance to the destination keeps decreasing by ϵ^2 in each step. Therefore, the game cannot go on for more than $\left\| p^{(0)} - p^* \right\|^2 / \epsilon^2$ steps.

Before we understand why Bob keeps getting closer every step (Claim 1), let us record a small piece of useful calculation here (see Appendix for the calculation).

Fact 4. *(A useful calculation) If $p^{(t+1)} = p^{(t)} + \eta v^{(t)}$ then*

$$\left\|p^{(t+1)} - p^*\right\|^2 - \left\|p^{(t)} - p^*\right\|^2 = 2\eta \cdot \left\langle v^{(t)}, (p^{(t)} - p^*)\right\rangle + \eta^2$$

By the above fact,

$$\begin{aligned}\left\|p^{(t+1)} - p^*\right\|^2 - \left\|p^{(t)} - p^*\right\|^2 &= 2\eta \cdot \underbrace{\left\langle v^{(t)}, (p^{(t)} - p^*)\right\rangle}_{\leqslant -\epsilon \text{ from Eq(2)}} + \epsilon^2 \\ &\leqslant -2\eta\epsilon + \epsilon^2 \\ &\leqslant -\epsilon^2 \quad \text{setting } \eta = \epsilon.\end{aligned}$$

In words, the above inequality says that Bob's squared distance decreases by ϵ^2 each step.

Solving Linear Programs

The feasibility problem for a general linear program is to find a point x satisfying a set of constraints.

Feasibility problem for linear programs:[2]
Find a point $x \in \mathbb{R}^d$ such that

$$\begin{aligned}\langle a_1, x\rangle &\geqslant b_1 \\ \langle a_2, x\rangle &\geqslant b_2 \\ &\cdots \\ \langle a_m, x\rangle &\geqslant b_m\end{aligned}$$

---

[2] Minimization or maximization versions of linear programs can be converted to a feasibility problem, by inserting the objective function as a constraint. For example, to maximize $\langle w, x\rangle$, one can include a constraint of the form $\langle w, x\rangle \geqslant B$, and use a binary search to find the largest B for which the resulting linear program is feasible.

Assume that the constraints of the linear program are normalized so that $\|a_i\| = 1$ for all $i = 1, \ldots, m$. Otherwise, we can always divide each constraint by an appropriate scaling factor to make $\|a_i\| = 1$.

The point-pursuit game can be used to solve this linear program efficiently. In each iteration, Bob proposes a solution $x^{(t)}$ for the above linear program. Alice looks for a constraint violated by Bob's solution, i.e., a constraint for which,

$$\langle a_i, x^{(t)}\rangle \leqslant b_i - \epsilon.$$

and returns the constraint as a separating hyperplane.

The idea is as follows: Let x^* be any solution that satisfies all the constraints of the linear program. Note that Alice does not know such an x^*. By definition, x^* will satisfy $\langle a_i, x^*\rangle \geqslant b_i$. If Bob's solution has $\langle a_i, x^{(t)}\rangle \leqslant b_i - \epsilon$, then $\ell(x) = \langle a_i, x\rangle - b_i$ is a separating hyperplane between x^* and $x^{(t)}$.

Therefore, if Alice and Bob continue to play the game then within $\|x^{(0)} - x^*\|^2/\epsilon^2$ steps, the game will be up. In other words, within these many steps, Bob will arrive at a solution $x^{(t)}$ for which Alice fails, i.e., $\langle a_i, x^{(t)} \rangle \geqslant b_i - \epsilon$ for all $i = 1, \ldots, m$.

Here is a description of the algorithm to solve LPs that simulates both Alice and Bob's play.

Algorithm 1: Solving LPs via Gradient Descent

$x^0 \leftarrow 1$

for $t = 0$ to T **do**

- (Find violated constraint) Find a constraint $\langle a_i, x \rangle \geqslant b_i$ violated by the $x^{(t)}$ with an error at least ϵ, i.e.,

$$\langle a_i, x^{(t)} \rangle \leqslant b_i - \epsilon$$

  If there is no violated constraint then **return** $x^{(t)}$.

- Set

$$x^{(t+1)} \leftarrow x^{(t)} + \eta \cdot a_i$$

end for

return *"No feasible solution within distance* $\epsilon\sqrt{T}$ *of the starting point* $x^{(0)} = 0$*"*

---

[3] A drawback of this algorithm is that it can only find solutions within a fixed distance $\epsilon\sqrt{T}$ of the initial point. Often, the constraints of the linear program immediately imply a bound on the distance $\|x\|$ of a feasible solution, and this may suffice. For example, if the variables are constrained as $0 \leqslant x_i \leqslant 1$, then $\|x\| \leqslant \sqrt{n}$ for all feasible x.

Exponential-sized linear programs

Algorithm 1 for solving linear programs has a curious feature. The number of constraints in the linear program does not directly affect the run-time of the algorithm. Instead, all that matters is whether given a candidate solution $x^{(t)}$, one can find a constraint it violates. We will see how this can be harnessed to solve linear programs with possibly exponentially many constraints!

Min-Cost Arborescence

Arborescence is a natural generalization of spanning trees to directed graphs. Fix a directed graph G and a vertex v within. An *arborescence rooted at* v is a directed tree containing all the vertices of the graph G, with v being the root and all the tree edges being directed away from the root (see Figure 3).

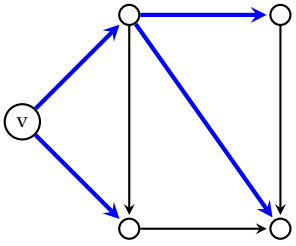

Figure 3: An arborescence rooted at vertex v is shown in blue.

In the *Min-Cost Arborescence* problem, the input is a directed graph G with a specified vertex v, along with non-negative weights $w : E \to \mathbb{R}$ on the edges. The goal is to find the arborescence rooted at vertex v with the minimum cost.

Here is a canonical linear program for the problem. For each directed edge e, associate a variable x_e whose intent is to indicate whether the edge e belongs to the arboresecence.

Consider any partition of the vertices $S \cup \overline{S} = V$ (see Figure 4). If $v \in S$ then there is at least one edge of the arborescence leaving the set S. Otherwise, $\overline{S}$ would not be reachable from v in the arborescence. Therefore, we can write the following constraint for every partition $S \cup \overline{S} = V$,

$$\sum_{e:e=u\to w \text{ where } u\in S, w\in\overline{S}} x_e \geqslant 1 \quad \text{for all } S \cup \overline{S} = V \text{ with } v \in S \qquad (4)$$

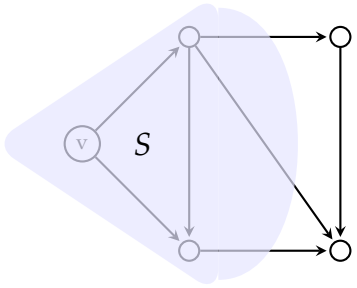

Figure 4: A set S containing the vertex v. Any arborescence rooted at v must pick at least one edge leaving the set S.

The objective function to minimize is naturally $\sum_{e\in E} w_e x_e$.

The above linear program for Min-Cost Arborescence has exponentially-many constraints, one for each subset $S \subset V$. Yet, one can solve this linear program efficiently using Algorithm 1, by finding a violated constraint efficiently (in polynomial time).

Suppose we are given a candidate solution x for the above linear program. Notice that the constraint (4) for each S, is equivalent to the following: If we set the weights of edges e to be x_e, then the minimum cut between v and any other vertex u is at least 1. Therefore, to check if x satisfies all the constraints (4), it is sufficient to assign capacities x_e and compute minimum cuts between v and other vertices u. Therefore, one can find a violating constraint using $n - 1$ Min-Cut computations.

Convex optimization

Convex sets

Definition. *(Halfspaces) A halfspace is the set of points on one side of a hyperplane, formally, for some $w \in \mathbb{R}^d$ and a real number θ,*

$$H = \left\{ x \in \mathbb{R}^d \middle| \langle w, x \rangle - \theta \geqslant 0 \right\}$$

Definition. *(Convex Set) A convex set is an intersection of a (possibly infinite) family of halfspaces.*

See the figure below for a few examples of convex sets expressed as intersections of halfspaces. Notice that a circular disk can be obtained by an intersection of infinitely many halfspaces.

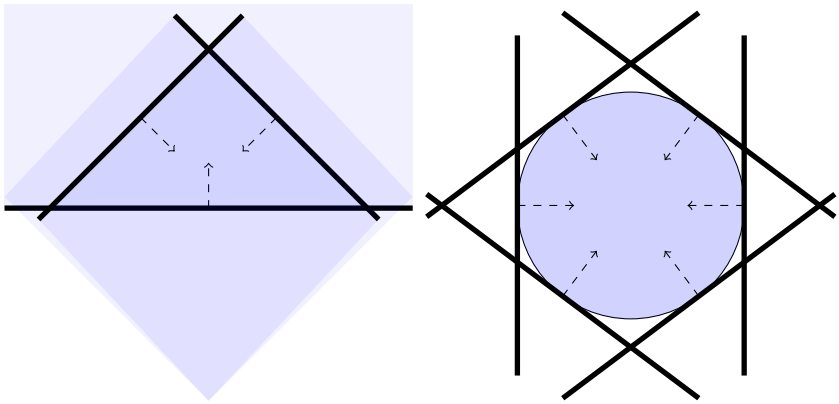

Figure 5: A triangle is an intersection of three halfspaces, while a circular disk D can be obtained via an intersection of infinitely many halfspaces

Consider a convex set $\mathcal{S}$, and let us say that $\mathcal{S}$ is the intersection of a family of halfspaces $\mathcal{H}$, i.e., $\mathcal{S} = \cap_{H \in \mathcal{H}} H$. Consider any point x *not in* the set $\mathcal{S}$. This implies that x is not in some halfspace $H \in \mathcal{H}$. In other words, whenever a point x is outside of a convex set, there is always some halfspace that separates the convex set from x.

Fact. *(Separating halfspaces always exist) If x is a point that is not in a convex set $\mathcal{S}$* [4] *then there exists a separating halfspace H such that,*

$$\mathcal{S} \subseteq H \text{ but } x \notin H$$

Let us understand convex sets a little better by looking at another equivalent definition for them.

Definition. *(Convex sets: equivalent definition) A set $\mathcal{S} \subseteq \mathbb{R}^d$ is a convex set if for every pair of points x, y within S, the line segment joining them is also contained in S* [5] *.*

Intuitively, a convex set $\mathcal{S}$ is a set of points that has no holes within them or no depressions on the boundary (See Figure 6,Figure 7 and Figure 8 for a few examples of convex and non-convex sets.)

[4] The fact only holds for a *closed* convex set. Without going into the definition, we will only consider closed convex sets.

[5] For a pair of points $x, y \in \mathbb{R}^d$, the line segment joining them is given by all points of the form $z = \lambda \cdot x + (1 - \lambda) \cdot y$ for some $\lambda \in [0, 1]$. Hence, a set $\mathcal{S}$ is convex if and only if,

$x, y \in \mathcal{S} \implies \lambda \cdot x + (1 - \lambda) \cdot y \in \mathcal{S}$ for all $\lambda \in [0, 1]$

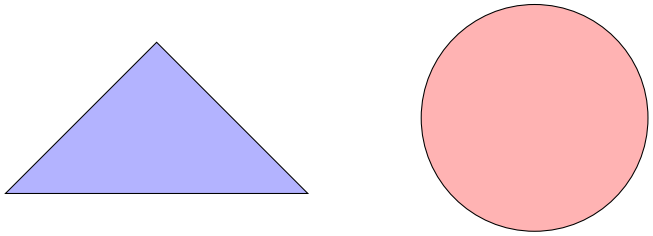

Figure 6: Two-dimensional convex sets

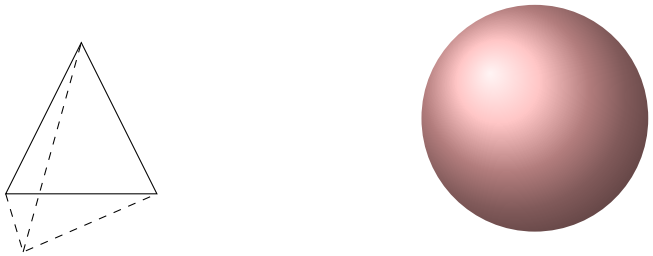

Figure 7: Three-dimensional convex sets: the tetrahedron and the sphere

A *separation oracle* $\mathcal{O}$ for a convex set $\mathcal{S}$ is an algorithm that given a point x not in $\mathcal{S}$, will find a halfspace separating x from the set $\mathcal{S}$.

In the previous section, we saw how Alice and Bob can use the point-pursuit game to find a feasible point for a linear program. In that context, Alice was playing the role of a *separation oracle* for the convex set that is the feasible region of the linear program, i.e., a violating constraint is nothing but a separating halfspace. Notice that the setup of Algorithm 1 makes sense even if we had infinitely many constraints (a.k.a. halfspaces), as long as we can find a violated constraint efficiently. Therefore, one can use Algorithm 1 to find points inside general convex sets for which we have a separation oracle. Let us formalize this intuition here.

Definition. *(Separation Oracle) An ϵ-separation oracle $\mathcal{O}$ for a convex set $\mathcal{S}$ is an algorithm that solves the following problem:*
Input: *A point x that is at least ϵ-away from the convex set $\mathcal{S}$.*
Output: *A halfspace $H = \{y | \langle w, y \rangle - \theta \geqslant 0\}$ such that $S \subseteq H$ but the point x is ϵ-away from H. The point x is ϵ-away from H if $\|w\| = 1$ and*

$$\langle w, x \rangle - \theta \leqslant -\epsilon$$

Theorem 5. *Given an ϵ-separating oracle $\mathcal{O}$ for a* **non-empty** *convex set $\mathcal{S}$, the algorithm Algorithm 2 finds a point x that is at most ϵ-away from the set $\mathcal{S}$ using $T = \Omega(D^2/\epsilon^2)$-calls to the oracle $\mathcal{O}$. Here D is the distance of the initial point $x^{(0)}$ from the set $\mathcal{S}$.*

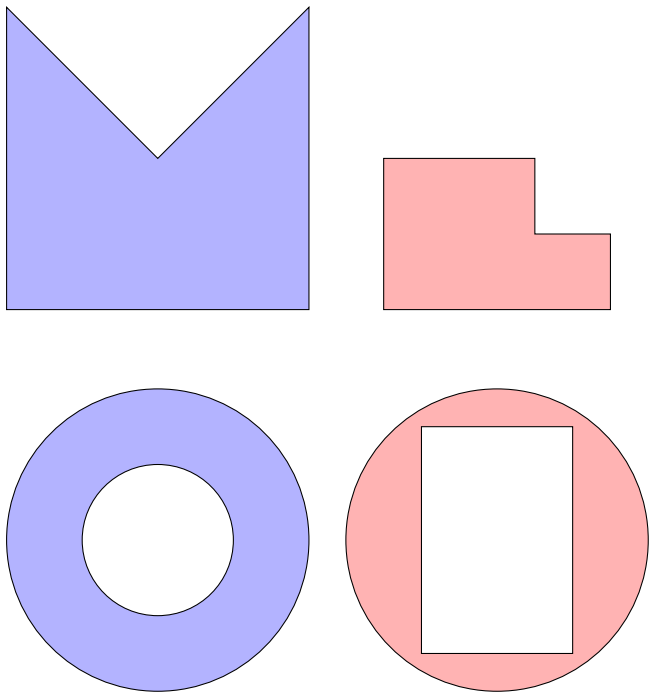

Figure 8: Two-dimensional non-convex sets

Algorithm 2: Finding a point in a convex set given the separation oracle

---

Input: A separation oracle $\mathcal{O}$ for some convex set $\mathcal{S}$

Output: A point x that is at most ϵ-away from the convex set $\mathcal{S}$

for $t = 0$ to T **do**

- (Find a separating halfspace) Use the oracle $\mathcal{O}$ to find a halfspace that separates the current point $x^{(t)}$ from the convex set $\mathcal{S}$, i.e., $\langle w, x^{(t)} \rangle - \theta \leqslant -\epsilon$ while, $\langle w, x \rangle - \theta \geqslant 0$ for all $x \in \mathcal{S}$.
  If there is no separating halfspace then **return** $x^{(t)}$.
- Set
$$x^{(t+1)} \leftarrow x^{(t)} + \eta \cdot w$$

end for

Exercise 6. *Consider the following problem:*

Disk Interior Problem

Input: *n circular disks on the two-dimensional plane specified by $\{(x_i, y_i, R_i) | i = 1, \ldots, n\}$ where (x_i, y_i) is the center and R_i is the radius of the i^{th} disk.*

Output: *A point $p^* = (x^*, y^*)$ that is inside all the disks (approximately). Formally, the point p^* must be within distance ϵ from the interior of every disk.*

---

1. *Consider the convex set $\mathcal{S}$ consisting of a single circular disk centered at (α, β) with radius R. Show how to implement an ϵ-separating oracle for $\mathcal{S}$.*

2. *Prove that the intersection of a family of convex sets is convex. This will imply that the intersection of a family of disks is a convex set.*

3. *Show how to use Algorithm 2 for the disk intersection problem. Can you make the running time be $O(n \cdot r^2/\epsilon^2)$ where r is the radius of the smallest disk?*

4. *What happens in every step of the algorithm?*

Convex functions

Definition 7. A function $f : \mathbb{R}^d \to \mathbb{R}$ is a convex function if for all points x, y,

$$f\left(\frac{x+y}{2}\right) \leqslant \frac{f(x)+f(y)}{2}$$

Intuitively, for every pair of points on the graph of f, the graph of the function dips under the line joining them (see Figure 9).

An equivalent definition of convexity for functions that have first derivatives is,

Fact 8. *A function is convex if and only if the following holds for all x, x^*,*

$$\langle \nabla f(x), x - x^* \rangle \geqslant f(x) - f(x^*)$$

A corollary of the above fact is the following,
"The gradient of a convex function gives a separating half-space between any point x and the global minimum x^*."

Suppose x^* is the global minima of f and let x be a point with $f(x) - f(x^*) \geqslant \epsilon$. From Fact 8, we get that

$$\langle \nabla f(x), x - x^* \rangle \geqslant \epsilon$$

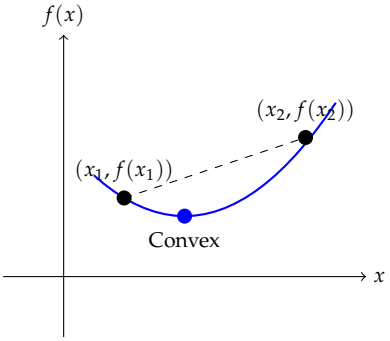

Figure 9: A convex function dips under the line joining any pair of points on the graph.

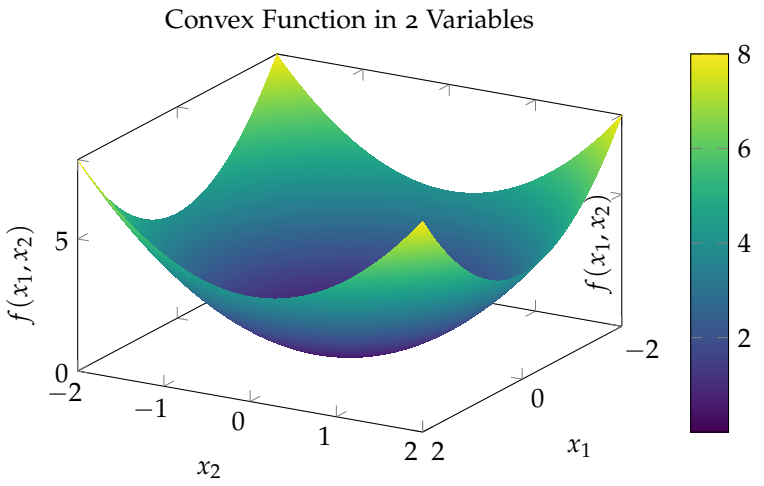

Figure 10: Convex function $f(x_1, x_2) = x_1^2 + x_2^2$

In other words, if we set $v = \frac{\nabla f(x)}{\|\nabla f(x)\|}$ then

$$\langle v, x \rangle - \langle v, x^* \rangle \geqslant \frac{\epsilon}{\|\nabla f(x)\|}$$

Analogous to (2), this is to say that the hyperplane normal to vector v is a separating hyperplane.

Alice and Bob can thus play the point pursuit game where for every point $x^{(t)}$ proposed by Bob, Alice just responds with the hyperplane defined by the gradient $\nabla f(x^{(t)})$. Bob will quickly converge to a point where the value of f is close to the global minimum $f(x^*)$.

More directly, this algorithm just *walks downhill* or moves along the negative gradient a.k.a. it is gradient descent.

Algorithm 3: Gradient descent for convex function

Input: A convex function $f : \mathbb{R}^d \to \mathbb{R}$ such that $\|\nabla f(x)\| \leqslant B$ for all x, and a starting point $x^{(0)} \in \mathbb{R}^d$.

for $t = 0$ to T **do**

Set

$$x^{(t+1)} \leftarrow x^{(t)} - \frac{\epsilon}{B} \cdot \frac{\nabla f(x^{(t)})}{\|\nabla f(x^{(t)})\|}$$

end for

return *" $x^{(t)}$ among $t = 0, \ldots, T$ with the smallest value for $f(x^{(t)})$."*

Theorem 9. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function such that the gradient vector is always bounded by B. Let x^* be the minimum value of f within a distance $\epsilon\sqrt{T}/B$ of the starting point $x^{(0)}$, i.e.,*

$$f(x^*) = \min_{x \text{ such that } \|x - x^0\| \leqslant \frac{\epsilon\sqrt{T}}{B}} f(x)$$

The algorithm returns a point x where,

$$f(x) - f(x^*) \leqslant \epsilon$$

As per the above theorem, the algorithm is guaranteed to find the minimum value of f within a certain distance $\epsilon\sqrt{T}/B$ of the starting point. To find the global minimum, one needs to set T large enough.

For the sake of completeness, we include a proof of the theorem below.

Proof. Let us prove the theorem by contradiction. Suppose for all iterations $x^{(t)}$, $f(x^{(t)}) - f(x^*) \geqslant \epsilon$. Then by Fact 8, for each t,

$$\langle \nabla f(x^{(t)}), x^{(t)} - x^* \rangle \geqslant \epsilon.$$

Since $\|\nabla f(x^{(t)})\| \leqslant B$ we get that,

$$\left\langle \frac{\nabla f(x^{(t)})}{\|\nabla f(x^{(t)})\|}, x^{(t)} - x^* \right\rangle \geqslant \frac{\epsilon}{B}. \tag{5}$$

By our useful little calculation (Fact 4),

$$\begin{aligned} \left\|x^{(t+1)} - x^*\right\|^2 - \left\|x^{(t)} - x^*\right\|^2 &= 2\eta \cdot \left\langle \frac{\nabla f(x^{(t)})}{\|\nabla f(x^{(t)})\|}, x^{(t)} - x^* \right\rangle + \eta^2 \\ &\qquad \text{where } \eta = -\frac{\epsilon}{B} \\ &\leqslant -\frac{\epsilon^2}{B^2} \qquad \text{using inequality (5)} \end{aligned}$$

In words, the squared distance to the minimum x^* drops by ϵ^2/B^2 in every iteration. By definition of x^*, the initial distance $\|x^* - x^{(0)}\|^2 \leqslant \frac{\epsilon^2}{B^2}T$. This is a contradiction, since squared distance cannot go negative. □

Multiplicative updates and KL-divergence

We began our study with the point pursuit game. For the game over $\mathbb{R}^d$, we designed a strategy for Bob that decreases squared distance to destination each step.

As it turns out, this is just one example of a general class of strategies for Bob. In particular, there is a successful strategy for Bob given any space $\mathcal{S}$, and a so-called "Bregman divergence" on it [6]. The space $\mathbb{R}^d$ and the squared distance is just the most natural example.

[6] We won't get into the general setup here. Interested reader may look up "mirror descent", Bregman divergence

Now we will study another very natural example, namely the space of probability distributions. Let us recall the point pursuit game again, this time on probability distributions. Let p^* be a probability distribution over $\{1, \ldots, n\}$.

Point Pursuit on Probability Distributions

In the t^{th} iteration,

1. Bob proposes a probability distribution

$$p^{(t)} = (p_1^{(t)}, \ldots, p_n^{(t)})$$

over $\{1, \ldots, n\}$.

2. Alice finds a *separating hyperplane* $\ell^{(t)}$ that separates Bob's current point $p^{(t)}$ from the destination p^*. Formally, Alice finds a vector $\ell^{(t)}$ so that,

$$\left\langle \ell^{(t)}, p^{(t)} - p^* \right\rangle \geqslant \epsilon \tag{6}$$

wherein each coordinate of $|\ell^{(t)}| \leqslant 1$.

Alice loses when she is unable to find a separating hyperplane.

Bob's strategy from the first section is not appropriate here, since $p^{(t+1)} = p^{(t)} + \eta \ell^{(t)}$ might take on negative values, while probability distributions need to be positive.

A different strategy for Bob is to update his probabilities multiplicatively. So Bob multiplies his i^{th} probability $p_i^{(t)}$ by a factor of $e^{-\eta \ell_i^{(t)}}$, and then scales all the probabilities so that they sum to 1. Formally, Bob's update rule is,

$$p_i^{(t+1)} = p_i^{(t)} \cdot \exp(-\eta \ell_i^{(t)}) \cdot \frac{1}{Z}$$

where Z is the scaling factor to ensure that the probabilities sum to 1. So,

$$Z = \sum_i p_i^{(t)} \exp(-\eta \ell_i^{(t)})$$

To analyze this multiplicative strategy for Bob, we need an appropriate *distance* [7] measure on probability distributions. The Kullback-Leibler distance between probability measures will be useful to this end.

[7] Divergences are not distances in the usual sense of the word. For example, divergences are not symmetric $\mathrm{D}_{KL}(p\|q) \neq \mathrm{D}_{KL}(q\|p)$

Definition 10. Given two probability distributions p, q over $\{1, \ldots, n\}$ the KL-divergence $\mathrm{D}_{KL}(p\|q)$ is given by,

$$\mathrm{D}_{KL}(p\|q) = \sum_{i=1}^{n} p_i \log\left(\frac{q_i}{p_i}\right)$$

As in the first section, the idea would be to argue that the KL-divergence to the destination distribution keeps dropping in every iteration. First, we need a version of Fact 4 for KL-divergence.

Fact 11. *Let p^*, q be probability distributions over $\{1, \ldots, n\}$. For some vector $\ell \in \mathbb{R}^n$, let q' be the distribution re-weighted by $e^{\eta \ell_i}$, i.e.,*

$$q'_i = \frac{q_i \cdot e^{-\eta \ell_i}}{\sum_{i=1}^n q_i e^{-\eta \ell_i}}$$

Then if $|\ell_i| \leqslant 1$ for all i,

$$\mathrm{D}_{KL}(p^* \| q) - \mathrm{D}_{KL}(p^* \| q') \geqslant \eta \langle \ell, q - p^* \rangle - \eta^2$$

We defer the above calculation to the appendix. Let us set the step-size parameter $\eta = \frac{\epsilon}{2}$. By Fact 11 and Equation (6) we get,

$$\mathrm{D}_{KL}\left(p^* \| p^{(t)}\right) - \mathrm{D}_{KL}\left(p^* \| p^{(t+1)}\right) \geqslant \eta \left\langle \ell^{(t)}, p^{(t)} - p^* \right\rangle - \eta^2 \geqslant \frac{\epsilon^2}{4}$$

In words, the KL divergence of Bob's distribution to p^* drops by $\frac{\epsilon^2}{4}$. Moreover, unlike squared distance in $\mathbb{R}^d$, the KL-divergence of distributions on $\{1, \ldots, n\}$ is always bounded by $\log n$.

Theorem 12. *With a step-size of $\eta = \frac{\epsilon}{2}$, Alice fails to find a separating hyperplane after $\frac{4 \log n}{\epsilon^2}$ iterations.*

Online convex optimization

We will now see how the framework of the point pursuit game is useful beyond minimizing fixed convex functions over convex sets. Here is a curious feature of the basic argument of point pursuit. Suppose Alice hasn't chosen on a destination point p^* and is responding with arbitrary halfspaces. Yet, if it so turns out that in the end, by say an accident, there is some p^* that is separated *on average* by many of Alice's halfspaces, then Bob would have arrived close to p^*. We will clarify this intuition with a concrete example now.

Consider a day-trader who is managing a portfolio consisting of n stocks. On each day, the trader has a trading budget of B dollars. By a suitable scaling of the units, let us just assume $B = 1$ henceforth.

On day t, the trader buys $p_i^{(t)}$ dollars worth of stock i, and sells them all at the end of the day. Let us suppose the price of stock i decreases $\ell_i^{(t)}$ percent on day t [8]. The total loss (or profit) incurred by the trader on day t is,

$$\sum_i \ell_i^{(t)} p_i^{(t)} = \langle \ell^{(t)}, p^{(t)} \rangle.$$

[8] The loss $\ell_i^{(t)}$ will be a negative number if stock appreciates on day i

At the end of T days, we will compare the day-trader's losses to the *best fixed portfolio p^* in hindsight*. In other words, suppose we knew all of the losses/profits (the numbers $\ell_i^{(t)}$) of the stocks on all T days apriori on day 1. Armed with this knowledge, we are allowed to buy a fixed portfolio of stocks, say p_i^* dollars of stock i on day 1 and hold the stocks till the end of day T. Then, the total loss incurred by such a *best fixed portfolio in hindsight* would be,

$$\min_{p^*:\sum_i p_i^*=1} \sum_{t=1}^{T} \langle \ell^{(t)}, p^* \rangle$$

The *regret* of the day-trader is the difference between their loss and the best fixed portfolio in hindsight,

$$\text{Regret} = \sum_{t=1}^{T} \langle \ell^{(t)}, p^{(t)} \rangle - \min_{p^*:\sum_i p_i^*=1} \sum_{t=1}^{T} \langle \ell^{(t)}, p^* \rangle$$

The *average regret* is just the regret per day,

$$\text{AverageRegret} = \frac{1}{T} \left(\sum_{t=1}^{T} \langle \ell^{(t)}, p^{(t)} \rangle - \min_{p^*:\sum_i p_i^*=1} \sum_{t=1}^{T} \langle \ell^{(t)}, p^* \rangle \right)$$

Can we have strategies for the day-trader that provably have nearly zero *average regret*? Let's think about this for a moment. We want the day-trader to compete against someone who knows all of the stock movements ahead of time on day 1 but need to pick a fixed portfolio. How could the day-trader do well in comparison? Surprisingly, our techniques can be used to achieve nearly zero regret over a long number of days.

Algorithm 4: Multiplicative weights

$\eta \leftarrow \frac{1}{\sqrt{T}}$

$p^0 \leftarrow (1/n, \ldots, 1/n)$

On day $t+1$,

- Set
$$p_i^{(t+1)} \leftarrow \frac{1}{Z} \cdot p_i^{(t)} \exp(-\eta \ell_i^{(t)})$$
where $Z = \sum_i p_i^{(t)} \exp(-\eta \ell_i^{(t)})$

Theorem 13. *For every sequence of loss vectors $\ell^{(t)}$, with $|\ell_i^{(t)}| \leqslant 1$, the average regret of multiplicative weights algorithm after T days is at most* $\frac{2 \log n}{\sqrt{T}}$

Notice that the average regret per day approaches 0 as $T \to \infty$!

Proof. Let us suppose p^* is the optimal fixed portfolio in hindsight. We will use Fact 11 on $q = p^{(t)}$ and $\ell = \ell^{(t)}$ to conclude,

$$\mathrm{D}_{KL}\left(p^* \| p^{(t)}\right) - \mathrm{D}_{KL}\left(p^* \| p^{(t+1)}\right) \geqslant \eta \left\langle \ell^{(t)}, p^{(t)} - p^* \right\rangle - \eta^2$$

for day t. Let us rewrite this inequality as,

$$\left\langle \ell^{(t)}, p^{(t)} \right\rangle - \left\langle \ell^{(t)}, p^{*} \right\rangle \leqslant \frac{1}{\eta}\left(\mathrm{D}_{KL}\left(p^{*} \| p^{(t)}\right) - \mathrm{D}_{KL}\left(p^{*} \| p^{(t+1)}\right)\right) + \eta$$

In words, the left-hand side of the above inequality is the *regret on day t*, since it is the difference between the loss of day-trader and the loss of the optimal portfolio in hindsight p^{*}.

Let us write down the above inequality for each day $1, \ldots, T$,

$$\begin{aligned}
\text{RegretDay 1} &\leqslant \frac{1}{\eta}\left(\mathrm{D}_{KL}\left(p^{*} \| p^{(1)}\right) - \mathrm{D}_{KL}\left(p^{*} \| p^{(2)}\right)\right) + \eta \\
\text{RegretDay 2} &\leqslant \frac{1}{\eta}\left(\mathrm{D}_{KL}\left(p^{*} \| p^{(2)}\right) - \mathrm{D}_{KL}\left(p^{*} \| p^{(3)}\right)\right) + \eta \\
\text{RegretDay 3} &\leqslant \frac{1}{\eta}\left(\mathrm{D}_{KL}\left(p^{*} \| p^{(3)}\right) - \mathrm{D}_{KL}\left(p^{*} \| p^{(4)}\right)\right) + \eta \\
&\ldots \\
&\ldots \\
\text{RegretDay T} - 1 &\leqslant \frac{1}{\eta}\left(\mathrm{D}_{KL}\left(p^{*} \| p^{(T-1)}\right) - \mathrm{D}_{KL}\left(p^{*} \| p^{(T)}\right)\right) + \eta \\
\text{RegretDay T} &\leqslant \frac{1}{\eta}\left(\mathrm{D}_{KL}\left(p^{*} \| p^{(T)}\right) - \mathrm{D}_{KL}\left(p^{*} \| p^{(T+1)}\right)\right) + \eta
\end{aligned}$$

Observe what happens to the right-hand sides if we add these inequalities together. The consecutive KL-divergences cancel (*telescope*) and we get,

$$\text{TotalRegret} \leqslant \frac{1}{\eta}\left(\mathrm{D}_{KL}\left(p^{*} \| p^{(1)}\right) - \mathrm{D}_{KL}\left(p^{*} \| p^{(T+1)}\right)\right) + \eta T$$

Fix $\eta = 1/\sqrt{T}$ and use the fact that $\mathrm{D}_{KL}\left(p^{*} \| p^{(1)}\right) \leqslant \log n$

$$\text{AverageRegret} \leqslant \frac{1}{\eta T}\left(\mathrm{D}_{KL}\left(p^{*} \| p^{(1)}\right) - \mathrm{D}_{KL}\left(p^{*} \| p^{(T+1)}\right)\right) + \eta \leqslant O\left(\frac{2\log n}{\sqrt{T}}\right)$$

□

Exercises

Exercise 14. *There are N-hours of grading to be done. There are n TAs and the i^{th} TA wishes to grade for at least ℓ_i and at most u_i hours. We would like to allocate the grading to the TAs. An allocation would be unfair if a subset of* 10% *of the TAs do more than* 50% *of the grading work.*

- *Write a linear program for the problem (your program will have exponentially many constraints).*
- *Design an efficient algorithm which solves the following problem: Given a candidate solution x to the linear program, find an LP constraint violated by x violated constraint if there exists one. (a.k.a., design an efficient separation oracle for the LP)*
- *Think through how the separation oracle implies an algorithm to solve the LP, via the point pursuit game.*

Appendix: Some Calculations

Proof of Fact 4

Fact. *(A useful calculation)*

$$\left\|p^{(t+1)} - p^*\right\|^2 - \left\|p^{(t)} - p^*\right\|^2 = 2\eta \cdot \left\langle v^{(t)}, (p^{(t)} - p^*)\right\rangle + \eta^2$$

Proof. To see the above fact, let's begin by writing out the first term,

$$\begin{aligned}\left\|p^{(t+1)} - p^*\right\|^2 &= \left\langle (p^{(t)} - p^*) + \eta v^{(t)}, (p^{(t)} - p^*) + \eta v^{(t)}\right\rangle \\ &= \left\langle (p^{(t)} - p^*), (p^{(t)} - p^*)\right\rangle + 2\left\langle \eta v^{(t)}, (p^{(t)} - p^*)\right\rangle + \left\langle \eta v^{(t)}, \eta v^{(t)}\right\rangle \\ &= \left\|p^{(t)} - p^*\right\|^2 + 2\eta \cdot \left\langle v^{(t)}, (p^{(t)} - p^*)\right\rangle + \eta^2 \cdot \underbrace{\left\|v^{(t)}\right\|^2}_{=1 \text{ since } v^{(t)} \text{ is unit vector}}\end{aligned}$$

Rearranging the terms, Fact 4 follows. □

Proof of Fact 11

Fact. *Let p^*, q be probability distributions over $\{1, \ldots, n\}$. For some vector $\ell \in \mathbb{R}^n$, let q' be the distribution re-weighted by $e^{\eta \ell_i}$, i.e.,*

$$q'_i = \frac{q_i \cdot e^{-\eta \ell_i}}{\sum_{i=1}^n q_i e^{-\eta \ell_i}}$$

Then if $|\ell_i| \leqslant 1$ for all i,

$$\mathrm{D}_{KL}(p^* \| q) - \mathrm{D}_{KL}(p^* \| q') \geqslant \eta \langle \ell, q - p^* \rangle - \eta^2$$

Proof. Let us denote

$$Z = \sum_i q_i e^{-\eta \ell_i}$$

Notice that

$$\begin{aligned} Z &= \sum_i q_i e^{-\eta \ell_i} \\ &\leqslant \sum_i q_i (1 - \eta \ell_i + \eta^2 \ell_i^2) && \text{since } e^x \leqslant 1 + x + x^2 \text{ for } x \in [-1, 1] \\ &\leqslant \sum_i q_i (1 + \eta^2) - \eta \sum_i q_i \ell_i && \text{because } |\ell_i| \leqslant 1 \\ &\leqslant (1 + \eta^2) - \eta \langle q, \ell \rangle && \text{because } \sum_i q_i = 1 \\ &\leqslant e^{\eta^2 - \eta \langle \ell, q \rangle} && \text{using } 1 + x \leqslant e^x \end{aligned}$$

Hence we have that,

$$\log Z \leqslant \eta^2 - \eta \langle \ell, q \rangle \tag{7}$$

$$
\begin{aligned}
\mathrm{D}_{KL}(p^*\|q) - \mathrm{D}_{KL}(p^*\|q') &= \sum_i p_i^* \left(\log\left(\frac{p_i^*}{q_i^*}\right) - \log\left(\frac{p_i^*}{q_i'}\right) \right) \\
&= \sum_i p_i^* \log\left(\frac{q_i'}{q_i}\right) \\
&= \sum_i p_i^* \log\left(\frac{e^{-\eta \ell_i}}{Z}\right) \\
&= -\sum_i p_i^* \log Z + \sum_i p_i^* \log e^{-\eta \ell_i} \\
&= -\log Z - \eta \sum_i p_i^* \ell_i \\
&\geqslant -\eta^2 + \eta \langle \ell, q \rangle - \eta \langle \ell, p^* \rangle \qquad \text{using Equation(7)} \\
&\geqslant \eta \left\langle \ell, q - p^* \right\rangle - \eta^2
\end{aligned}
$$

□