Algorithms for Data Science: Lecture 6

Vasileios Nakos

National Technical University of Athens

April 17, 2021

Basics of Continuous Optimization

Minimize or maximize efficiently a function over a domain.

Basics of Continuous Optimization

Minimize or maximize efficiently a function over a domain.
Most Machine Learning problems under a particular formulation can be solved as optimization problems.

Basics of Continuous Optimization

Minimize or maximize efficiently a function over a domain.
Most Machine Learning problems under a particular formulation can be solved as optimization problems.

- The interplay between optimization and ML is one of the most important developments in modern computational science.
- Deep neural networks.
- Reinforcement learning.
- Meta Learning.
- Variational inference.
- Markov chain Monte Carlo.
- Federated Learning.

Given function $f : \mathbb{R}^d \to \mathbb{R}$ find $x^\star$ such that

$$f(x^\star) = \min_x f(x),$$

Given function $f : \mathbb{R}^d \to \mathbb{R}$ find $x^\star$ such that

$$f(x^\star) = \min_x f(x),$$

or at least x' such that $f(x') \leq f(x^\star) + \epsilon$.

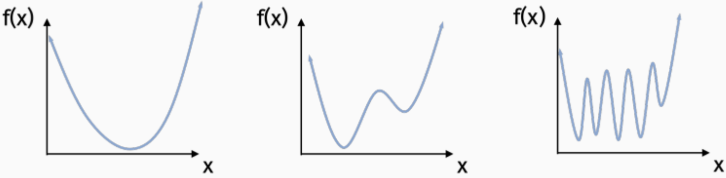
Given function $f : \mathbb{R}^d \to \mathbb{R}$ find $x^\star$ such that

$$f(x^\star) = \min_x f(x),$$

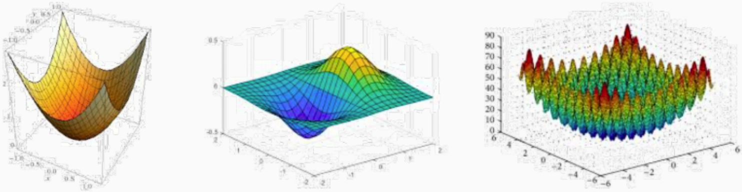
or at least x' such that $f(x') \leq f(x^\star) + \epsilon$.
Often additional constraints:

- $x_i > 0, \forall i \in [d]$.
- $\|x_2\| \leq R, \|x\|_1 \leq R$ (ℓ_2, ℓ_1 balls).
- $w^T x \leq c$ (hyperplane).
- $\Phi x = b$ (linear constraint)

Dimension $d = 1$:



Dimension $d = 2$:



In supervised learning, we want to learn a model that maps *inputs*

- numerical data vectors
- images, video
- text documents

In supervised learning, we want to learn a model that maps *inputs*

- numerical data vectors
- images, video
- text documents

to *predictions*

- numerical value (probability of mutation)
- label (is the image a human or a dragon?)
- decision (move bishop to G4)

Mathematical abstraction of supervised learning

Let M_x be a model with parameters $x = \{x_1, \ldots, x_k\}$ which takes as input a vector a and outputs a prediction.

Mathematical abstraction of supervised learning

Let M_x be a model with parameters $x = \{x_1, \ldots, x_k\}$ which takes as input a vector a and outputs a prediction.
For example, $M_x(a) = \mathrm{sign}(a^T x)$.

Mathematical abstraction of supervised learning

Let M_x be a model with parameters $x = \{x_1, \ldots, x_k\}$ which takes as input a vector a and outputs a prediction.
For example, $M_x(a) = \mathrm{sign}(a^T x)$.

In *supervised learning* we want to find a model that agrees with the data that you already have the answer for, i.e. datasets $a^{(i)}$ with output $y^{(i)}, i \in [n]$.

Mathematical abstraction of supervised learning

Let M_x be a model with parameters $x = \{x_1, \dots, x_k\}$ which takes as input a vector a and outputs a prediction.
For example, $M_x(a) = \mathrm{sign}(a^T x)$.

In *supervised learning* we want to find a model that agrees with the data that you already have the answer for, i.e. datasets $a^{(i)}$ with output $y^{(i)}, i \in [n]$.

$$\text{Find } x' \text{ such that } M_{x'}(a^{(i)}) \approx y^{(i)}, \forall i \in [n].$$

Mathematical abstraction of supervised learning

Let M_x be a model with parameters $x = \{x_1, \ldots, x_k\}$ which takes as input a vector a and outputs a prediction.
For example, $M_x(a) = \mathrm{sign}(a^T x)$.

In *supervised learning* we want to find a model that agrees with the data that you already have the answer for, i.e. datasets $a^{(i)}$ with output $y^{(i)}, i \in [n]$.

$$\text{Find } x' \text{ such that } M_{x'}(a^{(i)}) \approx y^{(i)}, \forall i \in [n].$$

Where is the optimization in all of these?

Loss functions

The loss function $L(\cdot, \cdot)$ is used as a measure of distance: $L(M_x(a), y)$ counts how far away is the prediction $M_x(a)$ from y.

- squared ℓ_2 loss: $|M_x(a) - y|^2$

The loss function $L(\cdot, \cdot)$ is used as a measure of distance: $L(M_x(a), y)$ counts how far away is the prediction $M_x(a)$ from y.

- squared ℓ_2 loss: $|M_x(a) - y|^2$
- absolute devation: $|M_x(a) - y|$

The loss function $L(\cdot,\cdot)$ is used as a measure of distance: $L(M_x(a), y)$ counts how far away is the prediction $M_x(a)$ from y.

- squared ℓ_2 loss: $|M_x(a) - y|^2$
- absolute devation: $|M_x(a) - y|$
- Hinge loss: $1 - y \cdot M_x(a)$

The loss function $L(\cdot, \cdot)$ is used as a measure of distance:
$L(M_x(a), y)$ counts how far away is the prediction $M_x(a)$ from y.

- squared ℓ_2 loss: $|M_x(a) - y|^2$
- absolute devation: $|M_x(a) - y|$
- Hinge loss: $1 - y \cdot M_x(a)$
- cross-entropy loss

Loss functions

The loss function $L(\cdot, \cdot)$ is used as a measure of distance: $L(M_x(a), y)$ counts how far away is the prediction $M_x(a)$ from y.

- squared ℓ_2 loss: $|M_x(a) - y|^2$
- absolute devation: $|M_x(a) - y|$
- Hinge loss: $1 - y \cdot M_x(a)$
- cross-entropy loss

Minimize the function

$$\sum_{i=1}^{n} L(M_x(a^{(i)}), y^{(i)}).$$

For $M_x(a) = x^T a$ and $L(z, y) = |z - y|^2$ we have

For $M_x(a) = x^T a$ and $L(z, y) = |z - y|^2$ we have

$$f(x) = \sum_{i=1}^{n} |x^T a^{(i)} - y^{(i)}|^2 = \|Ax - y\|_2^2,$$

where A is a matrix with $a^{(i)}$ as its i-th row and $y = [y^{(1)}, y^{(2)}, \ldots, y^{(n)}]^T$.

Gradient Descent

Gradient descent is a method for minimizing *convex* functions, but which also works surprisingly well in many practical scenarios.

Partial derivative:

$$\frac{\partial f}{\partial x_i} = \lim_{t \to 0} \frac{f(x + t \cdot e^{(i)}) - f(x)}{t}$$

Partial derivative:

$$\frac{\partial f}{\partial x_i} = \lim_{t \to 0} \frac{f(x + t \cdot e^{(i)}) - f(x)}{t}$$

Directional derivative:

$$D_v f(x) = \lim_{t \to 0} \frac{f(x + tv) - f(x)}{t}$$

Gradient:

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}(x), \frac{\partial f}{\partial x_2}(x), \ldots, \frac{\partial f}{\partial x_i}(x)\right]^T$$

Gradient:

$$\nabla f(x) = \left[\frac{\partial f}{\partial x_1}(x), \frac{\partial f}{\partial x_2}(x), \ldots, \frac{\partial f}{\partial x_i}(x)\right]^T$$

and its connection to directional derivative:

$$D_v f(x) = \nabla f(x)^T v$$

First order optimization

Given a function f to be minimized, we shall perform minimization by making

1. function accesses: evaluations of $f(x)$ for any x.

First order optimization

Given a function f to be minimized, we shall perform minimization by making

1. function accesses: evaluations of $f(x)$ for any x.
2. gradient accesses: evaluations of $\nabla f(x)$ for any x.

First order optimization

Given a function f to be minimized, we shall perform minimization by making

1. function accesses: evaluations of $f(x)$ for any x.
2. gradient accesses: evaluations of $\nabla f(x)$ for any x.

We will treat the evaluations as black boxes but depending on the problem they might be computationally expensive to implement.

Recall that we are given $a^{(1)}, \ldots, a^{(n)} \in \mathbb{R}^d$ and $y^{(1)}, \ldots, y^{(n)} \in \mathbb{R}$ and want to minimize

Gradient in Linear Regression

Recall that we are given $a^{(1)}, \ldots, a^{(n)} \in \mathbb{R}^d$ and $y^{(1)}, \ldots, y^{(n)} \in \mathbb{R}$ and want to minimize

$$f(x) = \sum_{i=1}^{n} \left(x^T a^{(i)} - y^{(i)} \right)^2 = \|Ax - y\|_2^2.$$

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{n} 2 \cdot (x^T a^{(i)} - y^{(i)}) \cdot a_j^{(i)} = 2(Ax - y)^T \cdot \underbrace{Ae_j}_{j\text{–}th \text{ column of A}}$$

Gradient in Linear Regression

Recall that we are given $a^{(1)}, \ldots, a^{(n)} \in \mathbb{R}^d$ and $y^{(1)}, \ldots, y^{(n)} \in \mathbb{R}$ and want to minimize

$$f(x) = \sum_{i=1}^{n} \left(x^T a^{(i)} - y^{(i)}\right)^2 = \|Ax - y\|_2^2.$$

$$\frac{\partial f}{\partial x_j} = \sum_{i=1}^{n} 2 \cdot (x^T a^{(i)} - y^{(i)}) \cdot a_j^{(i)} = 2(Ax - y)^T \cdot \underbrace{Ae_j}_{j-th \text{ column of A}}$$

$$\nabla f(x) = 2A^T(Ax - y).$$

Gradient Descent

Taylor approximation: $f(x+\delta) = f(x) + f(x)^T\delta + o(\|\delta\|_2^2)$.

Gradient Descent

Taylor approximation: $f(x+\delta) = f(x) + f(x)^T\delta + o(\|\delta\|_2^2)$.

Gradient descent is THE algorithm.

- For $i = 0$ to T (number of iterations)
  - $x^{(i+1)} = x^{(i)} - \eta \cdot \nabla f(x^{(i)})$ (η is the step)
- Return $\mathrm{argmin}_i x^{(i)}$

What happens?

When f is convex for sufficiently small η and sufficiently large T, gradient descent will converge to a global minimum:

$$f(x^{(T)}) \leq f(x^{\star}) + \epsilon.$$

See least squares regression, logistic and kernel regression, support vector machines etc

What happens?

When f is convex for sufficiently small η and sufficiently large T, gradient descent will converge to a global minimum:

$$f(x^{(T)}) \leq f(x^{\star}) + \epsilon.$$

See least squares regression, logistic and kernel regression, support vector machines etc

When f is non-convex for sufficiently small η and sufficiently large T, gradient descent will converge to a near stationary point:

$$\|\nabla f(x^{(T)})\|_2 \leq \epsilon.$$

The latter happens in neural networks.

Of course we are interested in the *rate of convergence.*

- Bounding the number of iteration in terms of ϵ, the starting point $x^{(0)}$ and the complexity of f.

Running time of gradient descent

Of course we are interested in the *rate of convergence.*

- Bounding the number of iteration in terms of ϵ, the starting point $x^{(0)}$ and the complexity of f.
- Depending on the assumptions on f, you get different convergence rates.

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for any $x, y \in \mathbb{R}^d$ and any $\lambda \in [0, 1]$ we have

$$(1 - \lambda)f(x) + \lambda f(y) \geq f((1 - \lambda)x + \lambda y)).$$

Convex function

A function f is convex if and only if for all x, y we have

$$f(x+z) \geq f(x) + \nabla f(x)^T z.$$

Alternative definition of convexity

Convex function

A function f is convex if and only if for all x, y we have

$$f(x+z) \geq f(x) + \nabla f(x)^T z.$$

Equivalently

$$f(x) - f(y) \leq \nabla f(x)^T (x - y).$$

Convex function

A function f is convex if and only if for all x, y we have

$$f(x+z) \geq f(x) + \nabla f(x)^T z.$$

Equivalently

$$f(x) - f(y) \leq \nabla f(x)^T (x-y).$$

1D analogue: $f(x) - f(y) \leq f'(x)(x-y)$.

Back to gradient descent

- f is convex

Back to gradient descent

- f is convex
- f is Lipschitz, i.e. $\forall x \|\nabla f(x)\|_2 \leq G$.

Back to gradient descent

- f is convex
- f is Lipschitz, i.e. $\forall x \|\nabla f(x)\|_2 \leq G$.
- good starting point x_0 s.t. $\|x^\star - x^{(0)}\|_2 \leq R$.

- f is convex
- f is Lipschitz, i.e. $\forall x \|\nabla f(x)\|_2 \leq G$.
- good starting point x_0 s.t. $\|x^\star - x^{(0)}\|_2 \leq R$.

- $\eta = \frac{R}{G\sqrt{T}}$
- For $i = 0$ to T (number of iterations)
  - $x^{(i+1)} = x^{(i)} - \eta \cdot \nabla f(x^{(i)})$
- Return $\mathrm{argmin}_i x^{(i)}$

Convergence Bound

If $T \geq \frac{R^2G^2}{\epsilon^2}$ and $\eta = \frac{R}{G\sqrt{T}}$ then $f(x^{(T)}) \leq f(x^\star) + \epsilon$.

Convergence Bound

If $T \geq \frac{R^2 G^2}{\epsilon^2}$ and $\eta = \frac{R}{G\sqrt{T}}$ then $f(x^{(T)}) \leq f(x^\star) + \epsilon$.

The “progress” claim: For all $i = 0, 1, \ldots, T$ we have

$$f(x^{(i)}) - f(x^\star) \leq \frac{\|x^{(i)} - x^\star\|_2^2 - \|x^{(i+1)} - x^\star\|_2^2}{2\eta} + \frac{\eta G^2}{2}.$$

Let's telescope

The "progress" claim: For all $i = 0, 1, \ldots, T$ we have

$$f(x^{(i)}) - f(x^\star) \leq \frac{\|x^{(i)} - x^\star\|_2^2 - \|x^{(i+1)} - x^\star\|_2^2}{2\eta} + \frac{\eta G^2}{2}.$$

Let's telescope

The "progress" claim: For all $i = 0, 1, \ldots, T$ we have

$$f(x^{(i)} - f(x^\star) \leq \frac{\|x^{(i)} - x^\star\|_2^2 - \|x^{(i+1)} - x^\star\|_2^2}{2\eta} + \frac{\eta G^2}{2}.$$

$$\sum_{i=0}^{T-1}[f(x^{(i)} - f(x^\star)] \leq \frac{\|x^{(0)} - x^\star\|_2^2 - \|x^{(T)} - x^\star\|_2^2}{2\eta} + \frac{T\eta G^2}{2}.$$

Let's telescope

The "progress" claim: For all $i = 0, 1, \ldots, T$ we have

$$f(x^{(i)} - f(x^\star) \leq \frac{\|x^{(i)} - x^\star\|_2^2 - \|x^{(i+1)} - x^\star\|_2^2}{2\eta} + \frac{\eta G^2}{2}.$$

$$\sum_{i=0}^{T-1} [f(x^{(i)} - f(x^\star)] \leq \frac{\|x^{(0)} - x^\star\|_2^2 - \|x^{(T)} - x^\star\|_2^2}{2\eta} + \frac{T\eta G^2}{2}.$$

$$\frac{1}{T}\sum_{i=0}^{T-1} [f(x^{(i)} - f(x^\star)] \leq \frac{R^2}{2T\eta} + \frac{\eta G^2}{2}$$

Convergence Bound

If $T \geq \frac{R^2G^2}{\epsilon^2}$ and $\eta = \frac{R}{G\sqrt{T}}$ then $f(x^{(T)}) \leq f(x^\star) + \epsilon$.

Convergence Bound

If $T \geq \frac{R^2G^2}{\epsilon^2}$ and $\eta = \frac{R}{G\sqrt{T}}$ then $f(x^{(T)}) \leq f(x^\star) + \epsilon$.

By our setting of parameters we have

$$\mathrm{argmin}_i x^{(i)} \leq \frac{1}{T}\sum_{i=0}^{T-1}[f(x^{(i)} - f(x^\star)] \leq \epsilon$$

Convergence Bound

If $T \geq \frac{R^2G^2}{\epsilon^2}$ and $\eta = \frac{R}{G\sqrt{T}}$ then $f(x^{(T)}) \leq f(x^\star) + \epsilon$.

By our setting of parameters we have

$$\mathrm{argmin}_i x^{(i)} \leq \frac{1}{T}\sum_{i=0}^{T-1}[f(x^{(i)} - f(x^\star)] \leq \epsilon$$

Convex Set

A set $S \subseteq \mathbb{R}^d$ is *convex* if and only

$$\forall x, y \in S, \forall \lambda \in [0, 1] : \lambda x + (1 - \lambda) y \in S.$$

Any line segment the endpoints of which are in S belongs totally into S.

Projection

Usually we are not interested in optimizing f over the whole space but rather over a convex domain.

Projection

Usually we are not interested in optimizing f over the whole space but rather over a convex domain.

For example $\min f(x)$ subject to $\Pi x = b$, where $\Pi \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$.

Projection

Usually we are not interested in optimizing f over the whole space but rather over a convex domain.

For example $\min f(x)$ subject to $\Pi x = b$, where $\Pi \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m$.

- From the points which satisfy a particular set of linear constraints, finding the one with the minimum $f(x)$.

Projection

Usually we are not interested in optimizing f over the whole space but rather over a convex domain.

For example $\min f(x)$ subject to $\Pi x = b$, where $\Pi \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m$.

- From the points which satisfy a particular set of linear constraints, finding the one with the minimum $f(x)$.
- The set $S := \{x : \Pi x = b\}$ is convex, since

Usually we are not interested in optimizing f over the whole space but rather over a convex domain.

For example $\min f(x)$ subject to $\Pi x = b$, where $\Pi \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m$.

- From the points which satisfy a particular set of linear constraints, finding the one with the minimum $f(x)$.
- The set $S := \{x : \Pi x = b\}$ is convex, since $\Pi(\lambda x + (1 - \lambda)y) = \lambda \Pi x + (1 - \lambda)\Pi y = \lambda b + (1 - \lambda)y = 1$

Usually we are not interested in optimizing f over the whole space but rather over a convex domain.

For example $\min f(x)$ subject to $\Pi x = b$, where $\Pi \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m$.

- From the points which satisfy a particular set of linear constraints, finding the one with the minimum $f(x)$.
- The set $S := \{x : \Pi x = b\}$ is convex, since $\Pi(\lambda x + (1 - \lambda)y) = \lambda \Pi x + (1 - \lambda)\Pi y = \lambda b + (1 - \lambda)y = 1$
- The ℓ_1 ball $S : \{x : \|x\| \leq 1\}$ is a convex set.

Usually we are not interested in optimizing f over the whole space but rather over a convex domain.

For example $\min f(x)$ subject to $\Pi x = b$, where $\Pi \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m$.

- From the points which satisfy a particular set of linear constraints, finding the one with the minimum $f(x)$.
- The set $S := \{x : \Pi x = b\}$ is convex, since $\Pi(\lambda x + (1-\lambda)y) = \lambda \Pi x + (1-\lambda)\Pi y = \lambda b + (1-\lambda)y = 1$
- The ℓ_1 ball $S : \{x : \|x\| \leq 1\}$ is a convex set.
- The classical max-flow problem can be cast as optimizing $f(x) = \|x\|_\infty$ over a linear system (the flow constraints).

What is inherently wrong with GD here?

- For $i = 0$ to T (number of iterations)
  - $x^{(i+1)} = x^{(i)} - \eta \cdot \nabla f(x^{(i)})$
- Return $\mathrm{argmin}_i x^{(i)}$

What is inherently wrong with GD here?

- For $i = 0$ to T (number of iterations)
  - $x^{(i+1)} = x^{(i)} - \eta \cdot \nabla f(x^{(i)})$
- Return $\mathrm{argmin}_i x^{(i)}$

It could be that $x^{(i)}$ *do not* belong inside the convex set S.

Projected Gradient Descent

Force $x^{(i)}$ to be in S by projecting onto it.

- For $i = 0$ to T (number of iterations)
  - $y^{(i+1)} = x^{(i)} - \eta \cdot \nabla f(x^{(i)})$
  - $x^{(i+1)} = \mathrm{argmin}_{z \in S} \|z - y^{(i+1)}\|_2^2$
- Return $\mathrm{argmin}_i x^{(i)}$

Force $x^{(i)}$ to be in S by projecting onto it.

- For $i = 0$ to T (number of iterations)
  - $y^{(i+1)} = x^{(i)} - \eta \cdot \nabla f(x^{(i)})$
  - $x^{(i+1)} = \mathrm{argmin}_{z \in S} \|z - y^{(i+1)}\|_2^2$
- Return $\mathrm{argmin}_i x^{(i)}$

The projection operator $\Pi_S(y) = \mathrm{argmin}_{z \in S} \|z - y\|_2^2$.

Demands of First order projected GD

Given a function f to be minimized over a (usually convex) set, we shall perform minimization by making

1. function accesses: evaluations of $f(x)$ for any x.

Given a function f to be minimized over a (usually convex) set, we shall perform minimization by making

1. function accesses: evaluations of $f(x)$ for any x.
2. gradient accesses: evaluations of $\nabla f(x)$ for any x.

Demands of First order projected GD

Given a function f to be minimized over a (usually convex) set, we shall perform minimization by making

1. function accesses: evaluations of $f(x)$ for any x.
2. gradient accesses: evaluations of $\nabla f(x)$ for any x.
3. projection accesses: finding $\Pi_S(y)$.

Analysis the roughly the same with a catch:

Projection does not increase distances from points in S

If S is a convex set, then for any $y \in S$ we have

$$\|y - \Pi_S(x)\|_2 \leq \|y - x\|_2.$$

Analysis the roughly the same with a catch:

Projection does not increase distances from points in S

If S is a convex set, then for any $y \in S$ we have

$$\|y - \Pi_S(x)\|_2 \leq \|y - x\|_2.$$

Proof using the separating hyperplane theorem.

Projected Gradient Descent Analysis

PGD Convergence Bound

If f, S are convex and $T \geq \frac{R^2 G^2}{\epsilon^2}$ then $f(x') \leq f(x^\star) + \epsilon$.

Recap

- Gradient descent is a first-order method for minimizing convex functions.

Recap

- Gradient descent is a first-order method for minimizing convex functions.
- Projected Gradient descent is a first-order method for minimizing convex functions over convex domains.

Recap

- Gradient descent is a first-order method for minimizing convex functions.
- Projected Gradient descent is a first-order method for minimizing convex functions over convex domains.
- To achieve acurracy ϵ what we've seen in class needs
    1. Efficient ways to evaluate $f(x), \nabla f(x)$.

Recap

- Gradient descent is a first-order method for minimizing convex functions.
- Projected Gradient descent is a first-order method for minimizing convex functions over convex domains.
- To achieve acurracy ϵ what we've seen in class needs
    1. Efficient ways to evaluate $f(x), \nabla f(x)$.
    2. An upper bound on $\|\nabla f(x)\|_2$.

Recap

- Gradient descent is a first-order method for minimizing convex functions.
- Projected Gradient descent is a first-order method for minimizing convex functions over convex domains.
- To achieve acurracy ϵ what we've seen in class needs
    1. Efficient ways to evaluate $f(x), \nabla f(x)$.
    2. An upper bound on $\|\nabla f(x)\|_2$.
    3. A good initial point.

Recap

- Gradient descent is a first-order method for minimizing convex functions.
- Projected Gradient descent is a first-order method for minimizing convex functions over convex domains.
- To achieve acurracy ϵ what we've seen in class needs
  1. Efficient ways to evaluate $f(x), \nabla f(x)$.
  2. An upper bound on $\|\nabla f(x)\|_2$.
  3. A good initial point.
  4. A way to project (in case of PGD) onto S.

Recap

- Gradient descent is a first-order method for minimizing convex functions.
- Projected Gradient descent is a first-order method for minimizing convex functions over convex domains.
- To achieve acurracy ϵ what we've seen in class needs
    1. Efficient ways to evaluate $f(x), \nabla f(x)$.
    2. An upper bound on $\|\nabla f(x)\|_2$.
    3. A good initial point.
    4. A way to project (in case of PGD) onto S.
    5. Then $\approx \frac{1}{\epsilon^2}$ iterations suffice.

Thank you!