Optimization for large scale machine learning
ENS, M2 Info

Mathurin Massias
mathurin.massias@gmail.com

Goals of the class

In this class we will study optimization problems arising in Machine Learning. These problems are usually characterized by their large scale, which call for dedicated classes of algorithms. They also usually present some specific structure, that can be leveraged to obtain moderate precision solutions.

The main objectives are to:

1. understand the taxonomy of optimization problems and which algorithms can be used in which settings
2. understand convergence properties and functions properties that govern them
3. master theoretical tools and techniques to derive convergence proofs
4. get hands on practice to challenge theoretical results, in Python

General notational conventions

We almost always work in finite dimension and the optimization is performed over the set $\mathbb{R}^d$. In a Machine Learning context, the integer d will thus denote the dimension of the parameter space, while n will denote the number of samples (observed data points). We adopt the optimization convention, i.e. we write a linear system as $Ax = b$ (while statistics has $y = X\beta$, the inverse problems field writes $f = Au$, signal processing writes $y = \Phi x$...).

As much as possible we write x^* for the minimizer, but due to the various meaning of star (Fenchel conjugation, adjoint), we also write $\hat{x}$ as in the statistical literature.

We will say strictly positive and positive rather than non-negative and positive.

These conventions are meant to ease reading, but there can always be exceptions. It is anyways a good exercise to adapt to different notation.

1 Motivation: gradient descent on ordinary least squares

1.1 Introduction

In this class we will consider the following problem:

$$\inf_{x \in \mathcal{C}} f(x) \ , \tag{1}$$

where $\mathcal{C}$ is a subset of $\mathbb{R}^d$ and f is a real-valued function on a subset of $\mathbb{R}^d$.

Formally, solving (1) means to find the largest lower bound on all values $f(x)$ when x describes $\mathcal{C}$. This infimum may not exist, and if it exists it may not necessarily be attained, even for nicely-behaved f. The reader should mind that most of the time, we write $\min_{\mathcal{C}} f$ instead of $\inf_{\mathcal{C}} f$; this is an abuse of notation and does not mean that the infimum is attained (not even that it is finite).

In Machine Learning, rather that the smallest value taken by f, we are more interested in finding a point where it is reached, that is

$$x^* \in \operatorname*{argmin}_{\mathcal{C}} f(x) \ , \tag{2}$$

meaning in finding $x^* \in \mathcal{C}$ such that for all $x \in \mathcal{C}$, $f(x^*) \leq f(x)$. As we shall see, once computed this x^* is subsequently used in statistical tasks.

Example 1.1 (Hyperplane fitting aka Ordinary least squares)**.** *Let $n \in \mathbb{N}$, let $a_1, \dots, a_n \in \mathbb{R}^d$, let $b_1, \dots, b_n \in \mathbb{R}$. In a statistical setting, suppose that there is an approximate*[1] *linear relationship between the a_i's and the b_i's:*

$$b_i \approx a_i^\top x \ . \tag{3}$$

One may try to infer x from the observation of the pairs (a_i, b_i). From the optimization point-of-view, this task can be formulated as follows: amongst all linear functions/hyperplanes $a \mapsto a^\top x$, the one that "best fits" (in the squared loss sense; other senses are possible) the n pairs (a_i, b_i) is given by

$$x^* \in \operatorname*{argmin}_{x \in \mathbb{R}^d} \sum_1^n (b_i - a_i^\top x)^2 = \operatorname{argmin} \frac{1}{2}\|Ax - b\|^2 \ , \tag{4}$$

*where the **design matrix** $A \in \mathbb{R}^{n \times d}$ has i-th row a_i, and the **target vector** $b \in \mathbb{R}^n$ has i-th entry b_i. Equation (4) is called Ordinary Least Squares.*

*We already see a first issue: we have written "*the *one that best fits", but it may not be unique*[2]*, hence we have written $x^* \in$* argmin *and not $x^* =$* argmin*.*

This is also an illustration that the value of the infimum is not always very interesting (if $d \geq n$ it is often 0) while the minimizer is more useful: it allows to determine the whole function.

Least Squares form a statistical perspective Suppose we have two random variables A and B linked by

$$B \sim \mathcal{N}(A^\top x, \sigma^2) \ . \tag{5}$$

Consider a dataset of n independent observations, meaning realizations of (A, B). The conditional log-likelihood writes:

$$\prod_{i=1}^n \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|a_i^\top x - b_i\|^2}{2\sigma^2}\right) \tag{6}$$

[1] in a voluntarily vague sense, one possible meaning being (5)

[2] When is it not unique? And does there always exist one?

Maximization of the negative log likelihood gives Equation (4): the least squares solution is the maximum likelihood estimator for the linear model with Gaussian noise. Inferring x allow to predict a value b_{n+1} if one observes a new entry point a_{n+1}.

Note: it is perfectly possible to postulate a different generative model than (5), or simply not impose a generative model, and find x by minimizing another cost/loss function. This is the setting of empirical risk minimization (ERM): one finds the parameters of a model by minimizing a cost function, often –but not always– arising from likelihood maximization.

Example 1.2 (ℓ_1 and Huber regression)**.** *Ordinary least squares are strongly influenced by outliers: the squared loss gives a lot of importance to large differences: in the objective function (4) a mismatch of 2 between b_i and $a_i^\top x$ costs 4, but a mismatch of 10 costs 100. A more* robust *solution, that is less dominated by large mismatches (or "outliers"), can be found by minimizing the sum of absolute errors (the ℓ_1 loss):*

$$\operatorname*{argmin}_{x\in\mathbb{R}^d} \sum_{i=1}^{n} |a_i^\top x - b_i| = \|Ax - b\|_1 \ . \tag{7}$$

Equation (7) is unfortunately more complicated to solve, as we shall see. It can be reformulated as a Linear Program, but the cost to solve it scales very badly with d. A hybrid of Equations (4) and (7) involves the Huber loss TODO graph and def

$$\operatorname*{argmin}_{x\in\mathbb{R}^d} \sum_{1}^{n} \operatorname{huber}_\rho(a_i^\top x - b_i) \ , \tag{8}$$

that is less sensitive to outliers, optimization friendly, but relies on a parameter $\rho \in \mathbb{R}_+$ that needs to be chosen appropriately.

Similar considerations (introducing a model, introducing a cost) lead to a wealth of optimization problems in ML:

- convex ERM
- PCA/eigenvalue variational formulation
- Image reconstruction
- Deep learning, non convex ERM

With the following considerations:

- we are interested in a minimizer rather than in the minimum
- we want algorithms that work for very large scale d: frequently in the millions and for deep learning, sometimes more than 100 billion
- the objective functions are not generic, they often come from a statistical model
- the problem solved is often a tractable proxy for an unsolvable problem
- the objectives may have exploitable structure: finite sum (stochastic), composite sum (proximal)

1.2 Solving least squares

Equation (4) can be solved in closed-form with a dedicated algorithm such as the Cholesky decomposition (see exercise sheet); but ℓ_1 Huber and most ERM regression models cannot. We need general purposes algorithms to solve classes of similar problems. [3]

Key questions:

- convergence guarantees ? Local, global? Which properties of cost? objective function, iterates, subsequence of iterates? best iterate, last iterate, expectation.
- convergence speed/rate? Which properties?
- quality of solution wrt original problem?

One of the arguably simplest algorithm is gradient descent (GD). The gradient of a function points in the direction of maximal increase, namely:

$$\nabla f(x) = \operatorname*{argmax}_{v \in \mathbb{R}^d} f'(x; v) = \operatorname*{argmax}_{v} \lim_{\epsilon \to 0} \frac{f(x + \epsilon v) - f(x)}{\epsilon} \ . \tag{9}$$

Gradient descent starts at a point x_0 and moves in the opposite direction of the gradient with a stepsize $\eta > 0$:

$$x_{t+1} = x_t - \eta \nabla f(x_t) \ . \tag{10}$$

Proposition 1.3. *Assume that there exist strictly positive μ and L such that $\mu \operatorname{Id} \preceq A^\top A \preceq L \operatorname{Id}$, then we obtain a rate if $\eta \leq 2/L$.* [4] *The iterates of gradient descent satisfy have the following convergence rate:*

$$TODO \tag{11}$$

Proof. Gradient descent on ordinary least squares read

$$x_{t+1} = x_t - \eta A^\top (A x_t - b) \tag{12}$$

Let x^* be any minimizer (there exists at least one: exercise sheet). Then $A^\top A x^* = A^\top b$ (exercise sheet), and so:

$$x_{t+1} - x^* = (\operatorname{Id} - \eta A A^\top)(x_t - x^*) \ . \tag{13}$$

We can also obtain a rate in objective function:

$$f(x_t) - f(x^*) = \langle \nabla f(x^*), x_t - x^* \rangle + \frac{1}{2}(x_t - x^*) A^\top A (x_t - x^) \tag{14}$$

$$= \frac{1}{2}(x_t - x^*) A^\top A (x_t - x^*) \tag{15}$$

$$\leq \frac{L}{2} \|x_t - x^*\|^2 \ . \tag{16}$$

□

Various speed, number of computations needed to reach an ϵ solution

Compare with cost of Cholesky decomposition.

Illustration L and μ: curvature, eigenvalues of the Hessian.

[3] TODO Even if an exact solution can be computed exactly, in ML we are usually not interested in the exact solution (for example, because of noise in the data) and so we may save computation time by using an iterative algorithm up to moderate precision.

[4] when is this rate maximal?

1.3 Numerical puzzles

On Figure 1 is the numerically observed convergence rate of GD on OLS. A is 500 by 500 with i.i.d. normal entries so it's full rank, $\mu > 0$ and Proposition 1.3 applies. Yet the rate we see is clearly not linear. Why?

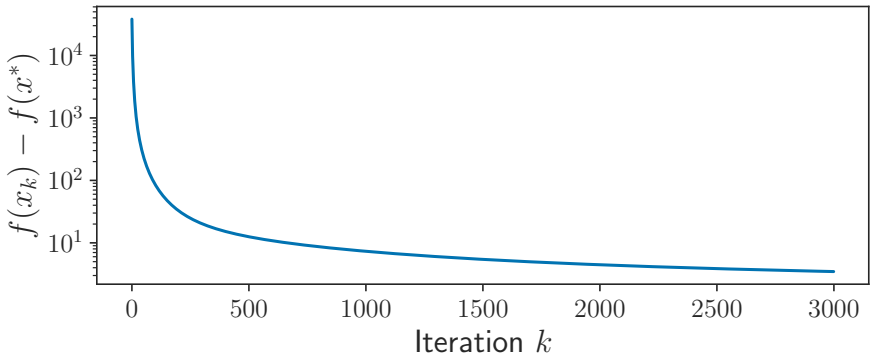

Figure 1: Objective convergence rate of GD on OLS on a random i.i.d. $A \in \mathbb{R}^{500\times500}$.

On Figure 2, A is still random but its shape is 200×300, so $A^\top A$ is not full rank, $\mu = 0$. Yet we see a clear linear convergence rate (numerical floating point errors kick in at iteration 1000). Why?

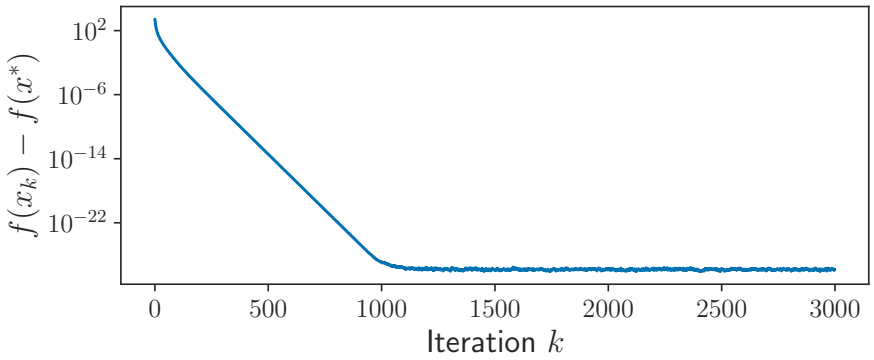

Figure 2: Objective convergence rate of GD on OLS on a random i.i.d. $A \in \mathbb{R}^{200\times300}$.

1.4 Exercises

Exercise 1.4. ☕ *Let $A \in \mathbb{R}^{n \times d}$. Show that* $\operatorname{Ker} A = \operatorname{Ker} A^*A$.
Show that for any $b \in \mathbb{R}^n$ there exist a solution to $A^*Ax = A^*b$.
☕☕ *Show that there does not always exist a solution to $A^*Ax = A^*b$ in the infinite dimensional case (when A is a bounded linear operator between infinite dimensional Hilbert spaces.)*

Exercise 1.5. ☕ *Let $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$. Show that solving Ordinary Least Squares:*

$$\min \frac{1}{2}\|Ax - b\|^2 \ , \tag{17}$$

amounts to solving $A^*Ax = A^*b$ (aka the normal equations*).*
Show that the set of solutions is:

$$(A^*A)^\dagger A^*b + \operatorname{Ker} A \ .$$

Exercise 1.6 (Least squares with intercept)**.** ☕☕ *An* intercept x_0 *is a constant scalar term in the linear prediction function, that becomes $a \mapsto a^\top x + x_0$. Fitting an intercept can be done by adding a column of 1s to A. Alternatively, show that the solution of least squares with intercept*

$$(\hat{x}, \hat{x}_0) \in \underset{x \in \mathbb{R}^d, x_0 \in \mathbb{R}}{\operatorname{argmin}} \frac{1}{2}\|Ax - b - x_0\mathbf{1}\|^2 \tag{18}$$

is given by:

$$\hat{x} = \hat{x}_c \ , \tag{19}$$

$$\hat{x}_0 = \frac{1}{n}\sum_1^n (a_i^\top \hat{x} - b_i) \ , \tag{20}$$

where $\hat{x}_c$ is the solution of least squares without intercept on centered data A_c and b_c (versions of A and b where the rowwise mean has been subtracted).

Exercise 1.7 (Gradient descent on isotropic parabola)**.** ☕ *Let $A \in \mathbb{R}^{n \times d}$ be such that its condition number*[5] *$\kappa(A)$ is equal to 1. Show that gradient descent with stepsize $1/L$ converges in a single iteration for the problem* $\min \frac{1}{2}\|Ax - b\|^2$.

[5] i.e. the ratio between the largest and the smallest eigenvalues of A^*A.

2 Convexity and gradient descent

In Section 1 we have seen that for gradient descent on ordinary least squares we can get very fast (so-called "linear") convergence rates in terms of function values and iterates. Two numbers governed this convergence: global upper and lower bounds L and μ on the Hessian.

Our proof was very ad hoc, relying on the explicit expression of the Hessian $A^\top A$. This technique doesn't work when the Hessian is not constant. The questions we try to answer in this section are:

- can we generalize the results of Proposition 1.3 to other objective functions
- under which conditions?
- what are the objective's property that influence the convergence rate we obtain?

How to measure convergence? We have seen convergence in iterates $x_k - x^*$; that is the most precise one can hope for. When the minimizer is not unique, we can generalize this to $\operatorname{dist}(x_k, \operatorname{argmin} f)$

We cannot analyze algorithm on a single function: we will never do better that the constant algorithm that starts at the minimizer of this function and stays there. Note that this algorithm is not very good on other functions. So we need to analyze algorithms on classes of function. What are the interesting classes?

2.1 Convexity

Definition 2.1. *A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if it lies below its cords:*

$$\forall x, y \in \mathbb{R}^d, \forall \lambda \in [0,1], \quad f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1 - \lambda f(y)) \ . \tag{21}$$

Convex functions are amenable to optimization because of the nice "local to global" properties they enjoy.

Proposition 2.2 (Local to global)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable convex function. Then all local minimizers of f are global minimizers, and*

$$\forall x, y \in \mathbb{R}^d, \quad f(x) - f(y) \leq \langle \nabla f(x), x - y \rangle \ .$$

Written as $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$ it tells that f lies above all of its tangents. From local information $\nabla f(x)$, we get an information about the whole behavior of f

Proposition 2.3 (Characterization on constrained minimizers)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function, let $\mathcal{C}$ be a non-empty closed and convex subset of $\mathbb{R}^d$. Then x^* is a minimizer of f restricted to $\mathcal{C}$ if and only if:*

$$\forall x \in \mathcal{C}, \quad \langle \nabla f(x^*), x^* - x \rangle \leq 0 \ . \tag{22}$$

aka the gradient of f points inwards $\mathcal{C}$ at x^*.

The proof is left as an exercize, and can be directly derived from Proposition 3.4 and **??**.

2.2 Gradient descent on convex functions

Proposition 2.4 (Gradient descent on convex smooth functions)**.** *Let f be a convex, L-Lipschitz differentiable function admitting at least one minimizer x^*. For $t \in \mathbb{N}$, the iterates of gradient descent $x_{k+1} = x_k - \eta \nabla f(x_k)$ satisfy*

$$\min_{1 \leq k \leq t} f(x_k) - f(x^*) \leq \frac{L\|x_0 - x^*\|}{\sqrt{t}} \ , \tag{23}$$

and

$$f(\tfrac{1}{t}\sum_{k=1}^{t} x_k) - f(x^*) \leq \frac{L\|x_0 - x^*\|}{\sqrt{t}} \ , \tag{24}$$

if the stepsize η is taken as $\eta = \frac{\|x_0 - x^*\|}{L\sqrt{t}}$.

Let us make a few observations before the proof:

- This algorithm is a bit weird: the number of iterations t must be known in advance to select the stepsize (we can get rid of this up to a slight worsening in the rate, with a decaying stepsize $\eta_t \propto 1/\sqrt{t}$).
- The algorithm can bounce, it is not a descent algorithm.
- The more iterations we do, the smaller we need to take the stepsize.
- The stepsize depends on the unknown quantity $\|x_0 - x^*\|$. We can get rid of this dependency by bounding $\|x_0 - x^*\|$ with e.g. the diameter of the domain.
- The kind of rate is not as strong as in the OLS case: we know nothing about the last iterate x_t and only have results on the ergodic (averaged) iterates or on the best iterate.

Proof. Let x^* be a minimizer of f. For $k \in \mathbb{N}$, using convexity of f, definition of x_{k+1} and the parallelogram identity,

$$\begin{aligned} f(x_k) - f(x^*) &\leq \langle \nabla f(x_k), x_k - x^* \rangle & (25)\\ &\leq \frac{1}{\eta}\langle x_k - x_{k+1}, x_k - x^* \rangle & (26)\\ &\leq \frac{1}{2\eta}\left(\|x_k - x_{k+1}\|^2 + \|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2\right) \ . & (27) \end{aligned}$$

Summing, the telescopic series cancels out, and since f is L-Lipschitz we can bound $\|x_{k+1} - x_k\| = \eta\|\nabla f(x_k)\|$ by ηL, so:

$$\begin{aligned} \frac{1}{t}\sum_{k=1}^{t} f(x_k) - f(x^*) &\leq \frac{1}{2t\eta}(t\eta^2 L^2 + \|x_0 - x^*\|^2 - \|x_{t+1} - x^*\|^2) & (28)\\ &\leq \frac{\eta L^2}{2} + \frac{\|x_0 - x^*\|^2}{2t\eta} & (29) \end{aligned}$$

Minimizing the RHS in η gives $\eta = \frac{R}{L\sqrt{t}}$ and the upper bound has value $\frac{RL}{\sqrt{t}}$. Jensen's inequality concludes. □

The projected gradient descent miracle Let $\mathcal{C}$ be a non-empty closed convex subset of $\mathbb{R}^d$. Consider the following constrained minimization problem:

$$\min_{x \in \mathcal{C}} f(x) \ , \tag{30}$$

and the *projected gradient descent* algorithm:

$$x_{k+1} = \Pi_{\mathcal{C}}(x_k - \eta \nabla f(x_k)) \ . \tag{31}$$

Then we have exactly the same rate as in Proposition 2.4: adding the constraint does not hurt the rate. Of course there is a caveat: we need to be able to compute $\Pi_{\mathcal{C}}$, which is an optimization problem in itself that can be quite hard. But for many $\mathcal{C}$'s it is OK (ℓ_1, ℓ_2, ℓ_∞ unit-balls).

The $1/\sqrt{t}$ rate is quite poor: to reach a precision of ε, one needs $\mathcal{O}(1/\varepsilon^2)$ iterations. With more assumptions on f, it can be improved to $\mathcal{O}(1/t)$, as we shall see in **??**.

Definition 2.5. *f is L-smooth if its gradient is L-Lipschitz*

Lemma 2.6 (Descent lemma)**.** *Let f be a L-smooth function.*

$$f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2}\|x - y\|^2 \ . \tag{32}$$

Proof. Notice that:

$$f(x) - f(y) = \int_0^1 \frac{d}{dt} f(y + t(x - y)) dt = \int_0^1 \langle \nabla f(y + t(x - y)), x - y \rangle dt \ . \tag{33}$$

Subtract $\langle \nabla f(y), x - y \rangle$, use the hypothesis on f, conclude. □

Lemma 2.7 (Cocoercivity of the gradient)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth function.*

$$\forall x, y \in \mathbb{R}^d, \quad \langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{1}{L}\|\nabla f(x) - \nabla f(y)\| \ . \tag{34}$$

Proposition 2.8 (Convergence rate of gradient descent on L-smooth functions)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex L-smooth function with non empty set of minimizers. Then the iterates of gradient descent with stepsize $< 2/L$ converge at the rate:*

$$f(x_k) - f(x^*) \leq TODO \frac{1}{2Lk} \tag{35}$$

This rate is much better, allows to take constant stepsize.

Proof. □

Finally we improve the rate to a linear one in the case where f is more than convex.

Definition 2.9. *f is μ-strongly convex:*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y) - \frac{\mu}{2}\lambda(1 - \lambda)\|x - y\|^2 \ . \tag{36}$$

Note: this is the only true definition of strong convexity. Other definitions are rather characterizations in special cases. In particular, many other definitions assume that the underlying norm is the Euclidean one, a requirement that this definition does not have. This plays a role in Section 17.

2.3 Exercises

2.3.1 Convexity

Exercise 2.10. ☕ *Show that local minimizers of convex functions are global minimizers.*

Exercise 2.11 (Pointwise supremum preserves convexity)**.** ☕ *Let $(f_i)_I$ be a family of convex functions (not necessarily countable). Show that $x \mapsto \sup_{i\in I} f_i(x)$ is convex.*

Exercise 2.12 (Precomposition by linear operator preserves convexity)**.** ☕ *Let $f : \mathcal{Y} \to \mathbb{R}$ be a convex function and $A : \mathcal{X} \to \mathcal{Y}$ a linear operator. Show that $f(A\cdot)$ is convex (on $\mathcal{X}$).*

Exercise 2.13 (Misconceptions on existence of minimizers)**.** ☕ *Provide an example of convex function which does not admit a minimizer.*
What if the function is continuous and lower bounded?

Exercise 2.14 (Continuity)**.** ☕☕☕ *Show that a convex function is locally Lipschitz (hence continuous) on the interior of its domain.*

2.3.2 Gradient

Exercise 2.15. ☕ *Provide an example of setting where the gradient is not equal to the vector of partial derivatives.*

Exercise 2.16. ☕ *Show that the gradient of a function is orthogonal to the level lines of that function.*

Exercise 2.17. ☕ *Compute the gradients and Hessians of $x \mapsto \|x\|^2$, $x \mapsto \|x\|$, $x \mapsto a^\top x$. Is it true that the gradient of $x \mapsto \frac{1}{2}x^\top Ax$ is equal to Ax?*

Exercise 2.18. ☕☕ *Compute the gradient of the logdet function, $M \mapsto \log\det(M)$.*

Exercise 2.19. ☕ *Let $A \in \mathbb{R}^{n\times d}$, $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}$ defined as $g(x) = f(Ax)$ for all $x \in \mathbb{R}^d$. Show that*

$$\nabla g(x) = A^*\nabla f(Ax) \ ,$$
$$\nabla^2 g(x) = A^*\nabla^2 f(Ax)A \ .$$

Exercise 2.20 (**??** in discrete time)**.** ☕☕ *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex, twice differentiable L-smooth function.*
Show that the iterates of gradient descent on f with step size $0 < \alpha < 2/L$ have decreasing gradient norm.
Can you show it if f is not twice differentiable?
Is it still true when f is not convex?

Exercise 2.21. ☕ *Provide a finite dimensional example of convex L-smooth function f such that gradient descent with stepsize $< 2/L$ diverges.*