0 Optimization Basics

Goal

Vector, matrix, norm, gradient and Hessian. Convex sets and functions. Fermat's optimality condition. Gradient algorithms and variants.

Alert 0.1: Convention

Gray boxes are not required hence can be omitted for unenthusiastic readers.

This note is a quick summary of the background needed in this course, and we are going to review most of it as we go. You do not need to fully understand everything here immediately. We collect the bits here simply for your convenience and the ease of later reference.

Definition 0.2: Vector space

Throughout the course, our universe is typically a (linear) vector space V over the real scalar field $\mathbb{R}$. On V we have two operators: addition $+ : \mathsf{V} \times \mathsf{V} \to \mathsf{V}$ and (scalar) multiplication $\cdot : \mathbb{R} \times \mathsf{V} \to \mathsf{V}$. Together they satisfy the following axioms:

- Addition commutativity: $\forall \mathbf{u}, \mathbf{v} \in \mathsf{V},\ \mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$;
- Addition associativity: $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathsf{V}, (\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$;
- Identity of addition: $\exists \mathbf{0} \in \mathsf{V}$ such that $\forall \mathbf{v} \in \mathsf{V}, \mathbf{0} + \mathbf{v} = \mathbf{v}$;
- Inverse of addition: $\forall \mathbf{v} \in \mathsf{V}, \exists$ unique $\mathbf{u} \in \mathsf{V}$ such that $\mathbf{v} + \mathbf{u} = \mathbf{0}$, in which case we denote $\mathbf{u} = -\mathbf{v}$;
- Identity of multiplication: $\forall \mathbf{v} \in \mathsf{V}, 1 \cdot \mathbf{v} = \mathbf{v}$, where $1 \in \mathbb{R}$ is the usual constant 1;
- Compatibility: $\forall a, b \in \mathbb{R}, \mathbf{v} \in \mathsf{V}, a(b\mathbf{v}) = (ab)\mathbf{v}$;
- Distributivity over vector addition: $\forall a \in \mathbb{R}, \mathbf{u}, \mathbf{v} \in \mathsf{V}, a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$;
- Distributivity over scalar addition: $\forall a, b \in \mathbb{R}, \mathbf{v} \in \mathsf{V}, (a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$.

These axioms are so natural that you may question why do we bother to formalize them? Well, take two images/documents/graphs/speeches/DNAs/etc., how do you add them? multiply with scalars? inverse? Not so obvious... Representing objects as vectors in a vector space so that we can exploit linear algebra tools is arguably one of the most important ideas in ML/AI.

Example 0.3: Euclidean space

The most common vector space we are going to need is the d-dimensional Euclidean space $\mathbb{R}^d$. Each vector $\mathbf{v} \in \mathbb{R}^d$ can be identified with a d-tuple: $\mathbf{v} = (v_1, v_2, \ldots, v_d)$. The addition and multiplication operators are defined element-wise, and we can easily verify the axioms:

- Let $\mathbf{u} = (u_1, u_2, \ldots, u_d)$, then $\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2, \ldots, u_d + v_d)$;
- Verify associativity yourself;
- $\mathbf{0} = (0, 0, \ldots, 0)$;
- $-\mathbf{v} = (-v_1, -v_2, , \ldots, -v_d)$;
- Obvious;
- $\mathbf{v} = (v_1, v_2, \ldots, v_d)$;

- Verify distributivity yourself;
- Verify distributivity yourself.

Definition 0.4: Vectors

- A vector is simply an array of real numbers $\mathbf{v} = (v_1, \dots, v_d)$, where d is the length (dimension) and we use v_j to index the j-th entry.
- index in math (and Julia) starts from 1 while in many programming languages (e.g. Python) index starts from 0. Know your poison!
- we have row vectors as above and column vectors which are transpose of row vectors, such as $\mathbf{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_d \end{pmatrix}$. (Sometimes we use square brackets, instead of the round ones.)
- When treated as a (degenerate) matrix, a row vector has size $1 \times d$ and a column vector has size $d \times 1$.
- By default, vectors are column vectors.

Definition 0.5: Inner product

For two vectors $\mathbf{u}, \mathbf{v}$ of the same size, their inner product (a.k.a. dot product, scalar product) is a real number:

$$\langle \mathbf{u}, \mathbf{v} \rangle := \sum_{j=1}^{d} u_j v_j.$$

Two vectors $\mathbf{u}$ and $\mathbf{v}$ are orthogonal, in notation $\mathbf{u} \perp \mathbf{v}$, iff $\langle \mathbf{u}, \mathbf{v} \rangle = 0$.
The following are obviously true. In fact, they form the abstract definition of an inner product:

- Symmetry: $\langle \mathbf{u}, \mathbf{v} \rangle = \langle \mathbf{v}, \mathbf{u} \rangle$.
- Positive definiteness: $\langle \mathbf{u}, \mathbf{u} \rangle \geq 0$, with equality iff $\mathbf{u} = \mathbf{0}$.
- Linearity: $\langle \alpha \mathbf{u} + \beta \mathbf{v}, \mathbf{w} \rangle = \alpha \langle \mathbf{u}, \mathbf{w} \rangle + \beta \langle \mathbf{v}, \mathbf{w} \rangle$ for any $\alpha, \beta \in \mathbb{R}$ and three vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$.

Definition 0.6: Matrix

- A matrix is a 2-D array of real numbers:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} = [\mathbf{a}_{:1}, \dots, \mathbf{a}_{:n}] = \begin{bmatrix} \mathbf{a}_{1:} \\ \vdots \\ \mathbf{a}_{m:} \end{bmatrix} \in \mathbb{R}^{m \times n},$$

where m is the number of rows and n is the number of columns. We use a_{ij} to index the (i, j)-th entry. We use $\mathbf{a}_{:j} =: \mathbf{a}_j$ to index the j-th column and $\mathbf{a}_{i:}$ to index the i-th row.

- Matrix transpose:

$$A^\top = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix} \in \mathbb{R}^{n\times m}$$

- A is symmetric if $A = A^\top$ (with necessarily $m = n$).

Alert 0.7: Notation

- Scalars are in lower-case, vectors are **boldface lower-case** and matrices are UPPER-CASE, e.g. we use A for a matrix, $\mathbf{a}_j$ for its j-th column vector and a_{ij} for its (i,j)-th scalar entry.
- a_j is the j-th scalar entry of the vector $\mathbf{a}$ while the boldface $\mathbf{a}_j$ is a vector that may have nothing to do with the vector $\mathbf{a}$!

Definition 0.8: Matrix Multiplication

- Given $A \in \mathbb{R}^{m\times n}, B \in \mathbb{R}^{n\times p}$, their product is defined as:

$$C = AB \in \mathbb{R}^{m\times p}, \quad \text{where} \quad c_{ik} = \sum_{j=1}^{n} a_{ij}b_{jk}.$$

  The number of columns in A must match the number of rows in B.
- For two column vectors $\mathbf{u}$ and $\mathbf{v}$ of the same size: $\langle \mathbf{u}, \mathbf{v}\rangle = \mathbf{u}^\top \mathbf{v}$.
- Prove $(AB)^\top = B^\top A^\top$.
- Row interpretation: $c_{ik} = \langle \mathbf{a}_{i:}, \mathbf{b}_{:k}\rangle$.
- Column interpretation: $C = \sum_{j=1}^{n} \mathbf{a}_{:j}\mathbf{b}_{j:}$.
- Complexity of the above definition: $O(mnp)$. Faster algorithms exist but the optimal algorithm is yet to be found.
- Inverse of a square matrix $A \in \mathbb{R}^{n\times n}$ is defined as the matrix B such that $AB = I$, where I is the identity matrix with $i_{jk} = 1$ iff $j = k$ and 0 otherwise. In notations: $B = A^{-1}$.
- $(A^{-1})^{-1} = A$, $(AB)^{-1} = B^{-1}A^{-1}$, and $(A^\top)^{-1} = (A^{-1})^\top$.

Definition 0.9: Eigenvalues

We call a nonzero (complex) scalar $\lambda \in \mathbb{C}$ and vector $\mathbf{v} \in \mathbb{C}^d$ eigenvalue and eigenvector of a square matrix $A \in \mathbb{R}^{d\times d}$ iff

$$A\mathbf{v} = \lambda\mathbf{v}.$$

- Every matrix has d eigenvalues (counting multiplicity).
- For symmetric (real) matrices, eigenvalues and eigenvectors are real. There are d orthogonal eigenvectors.

- A real symmetric matrix is called positive semidefinite (definite) iff all of its eigenvalues are nonnegative (positive). In notations, $A \succeq 0$ or $A \succ 0$. Similarly, we can define negative (semi)definiteness. A real symmetric matrix is indefinite if it is neither PSD or NSD.
- The singular value of any matrix (square or not) A is defined as the square root of (nonzero) eigenvalues of $AA^\top$.

All algorithms for computing eigenvalues must be iterative in nature! See this note for some details.

Definition 0.10: Norms and Cauchy-Schwarz inequality

For any vector $\mathbf{w} \in \mathbb{R}^d$, its Euclidean ($\ell_2$) norm (i.e., length) is defined as:

$$\|\mathbf{w}\|_2 := \sqrt{\mathbf{w}^\top \mathbf{w}} = \sqrt{\sum_{j=1}^d |w_j|^2}.$$

More generally, for any $p \geq 1$, we define the ℓ_p norm

$$\|\mathbf{w}\|_p := \big(\sum_{j=1}^d |w_j|^p\big)^{1/p}$$

while for $p = \infty$ we define the max norm

$$\|\mathbf{w}\|_\infty := \max_{j=1,\dots,d} |w_j|.$$

Even more generally, a norm is any function $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}_+$ that satisfies:

- (definite) $\|\mathbf{w}\| = 0 \iff \mathbf{w} = \mathbf{0}$
- (homogeneous) for all $\lambda \in \mathbb{R}$ and $\mathbf{w} \in \mathbb{R}^d$, $\|\lambda \mathbf{w}\| = |\lambda| \cdot \|\mathbf{w}\|$
- (triangle inequality) for all $\mathbf{w}$ and $\mathbf{z} \in \mathbb{R}^d$:

$$\|\mathbf{w} + \mathbf{z}\| \leq \|\mathbf{w}\| + \|\mathbf{z}\|.$$

The norm function is a convenient way to convert a vector quantity to a real number, for instance, to facilitate numerical comparison. Part of the business in machine learning is to understand the effect of different norms on certain learning problems, even though all norms are "formally equivalent:" for any two norms $\|\cdot\|$ and $|||\cdot|||$, there exist constants $c_d, C_d \in \mathbb{R}$ so that $\forall \mathbf{w} \in \mathbb{R}^d$,

$$c_d \|\mathbf{w}\| \leq |||\mathbf{w}||| \leq C_d \|\mathbf{w}\|.$$

The subtlety lies on the dependence of the constants c_d, C_d on the dimension d: could be exponential and could affect a learning algorithm a lot.

The dual (norm) $\|\cdot\|_\circ$ of the norm $\|\cdot\|$ is defined as:

$$\|\mathbf{z}\|_\circ := \max_{\|\mathbf{w}\|=1} \mathbf{w}^\top \mathbf{z} = \max_{\mathbf{w}\neq 0} \frac{\mathbf{w}^\top \mathbf{z}}{\|\mathbf{w}\|} = \max_{\|\mathbf{w}\|=1} |\mathbf{w}^\top \mathbf{z}| = \max_{\mathbf{w}\neq 0} \frac{|\mathbf{w}^\top \mathbf{z}|}{\|\mathbf{w}\|}.$$

From the definition it follows the important inequality:

$$\mathbf{w}^\top \mathbf{z} \leq |\mathbf{w}^\top \mathbf{z}| \leq \|\mathbf{w}\| \cdot \|\mathbf{z}\|_\circ.$$

The Cauchy-Schwarz inequality, which will be repeatedly used throughout the course, is essentially a self-duality property of the ℓ_2 norm:

$$\mathbf{w}^\top \mathbf{z} \leq |\mathbf{w}^\top \mathbf{z}| \leq \|\mathbf{w}\|_2 \cdot \|\mathbf{z}\|_2,$$

i.e., the dual norm of the ℓ_2 norm is itself. The dual norm of the ℓ_p norm is the ℓ_q norm, where

$$\infty \geq p, q \geq 1 \text{ and } \frac{1}{p} + \frac{1}{q} = 1.$$

Exercise 0.11: Norms

Prove the following:

- for any $\mathbf{w} \in \mathbb{R}^d$: $\|\mathbf{w}\|_p \to \|\mathbf{w}\|_\infty$ as $p \to \infty$
- for any $\infty \geq p \geq 1$, the ℓ_p norm is indeed a norm. What about $p < 1$?
- the dual of any norm $\|\cdot\|$ is indeed again a norm
- for any $\infty \geq p \geq q \geq 1$, $\|\mathbf{w}\|_p \leq \|\mathbf{w}\|_q \leq d^{\frac{1}{q}-\frac{1}{p}}\|\mathbf{w}\|_p$
- for any $\mathbf{w}, \mathbf{z} \in \mathbb{R}^d$: $\|\mathbf{w}+\mathbf{z}\|_2^2 + \|\mathbf{w}-\mathbf{z}\|_2^2 = 2(\|\mathbf{w}\|_2^2 + \|\mathbf{z}\|_2^2)$.

Definition 0.12: Matrix norms

For a matrix $A \in \mathbb{R}^{n\times d}$, we define its Frobenius norm

$$\|A\|_{\mathsf{F}} = \sqrt{\sum_{ij} a_{ij}^2},$$

which is essentially the matrix analogue of the vector Euclidean norm $\|\mathbf{w}\|_2$.

Given two vector norms $\|\cdot\|_{(1)}$ and $\|\cdot\|_{(2)}$ on $\mathbb{R}^d$ and $\mathbb{R}^n$, respectively, we define the induced matrix norm on $\mathbb{R}^{n\times d}$ as:

$$\|A\| = \max_{\|\mathbf{w}\|_{(1)}=1} \|A\mathbf{w}\|_{(2)}.$$

With the induced matrix norm, we have the following inequality:

$$\|A\mathbf{w}\|_{(2)} \leq \|A\| \cdot \|\mathbf{w}\|_{(1)}.$$

In particular, with $\|\cdot\|_{(1)} = \|\cdot\|_2$ and $\|\cdot\|_{(2)} = \|\cdot\|_2$, we obtain the spectral norm, which coincides with the largest singular value of A.

It is known that

$$\|A\|_{\text{sp}} \leq \|A\|_{\mathsf{F}} \leq \sqrt{\text{rank}(A)}\|A\|_{\text{sp}}.$$

Remark 0.13: Pseudo-inverse

Recall that the Moore-Penrose pseudo-inverse $A^\dagger$ of any matrix $A \in \mathbb{R}^{n\times d}$ is the unique matrix $G \in \mathbb{R}^{d\times n}$ so that

$$AGA = A, \quad GAG = G, \quad (AG)^\top = AG, \quad (GA)^\top = GA.$$

In particular, if $A = USV^\top$ is the thin SVD (singular value decomposition) of A, then $A^\dagger = VS^{-1}U^\top$. If A is in fact invertible, then $A^\dagger = A^{-1}$.

We can use pseudo-inverse to solve the linear least squares problem. Indeed, it is known that $W^\star := A^\dagger C B^\dagger$ is a solution for the minimization problem:

$$\min_W \quad \|AWB - C\|_{\mathsf{F}}^2.$$

In fact, $W^\star$ is the unique solution that enjoys the minimum Frobenius norm. See this lecture note for proofs and more related results.

Definition 0.14: Gradient and Hessian (for the brave hearts)

Recall that the gradient of a smooth function $f : \mathbb{R}^d \to \mathbb{R}^m$ at $\mathbf{w}$ is defined as the linear mapping $\nabla f(\mathbf{w}) : \mathbb{R}^d \to \mathbb{R}^m$ so that:

$$\lim_{\mathbf{0}\neq\Delta\mathbf{w}\to\mathbf{0}} \frac{\|f(\mathbf{w}+\Delta\mathbf{w}) - f(\mathbf{w}) - [\nabla f(\mathbf{w})](\Delta\mathbf{w})\|}{\|\Delta\mathbf{w}\|} = 0, \tag{0.1}$$

where say the norm $\|\cdot\|$ is the Euclidean norm. Or equivalently in big-o notation:

$$\|f(\mathbf{w}+\Delta\mathbf{w}) - f(\mathbf{w}) - [\nabla f(\mathbf{w})](\Delta\mathbf{w})\| = o(\|\Delta\mathbf{w}\|).$$

As $\mathbf{w}$ varies, we may think of the gradient as the (nonlinear) mapping:

$$\nabla f : \mathbb{R}^d \to \mathcal{L}(\mathbb{R}^d, \mathbb{R}^m), \qquad \mathbf{w} \mapsto [\nabla f(\mathbf{w}) : \mathbb{R}^d \to \mathbb{R}^m],$$

where $\mathcal{L}(\mathbb{R}^d, \mathbb{R}^m)$ denotes the class of linear mappings from $\mathbb{R}^d$ to $\mathbb{R}^m$, or equivalently the class of matrices $\mathbb{R}^{m\times d}$.

We can iterate the above definition. In particular, replacing f with ∇f we define the Hessian $\nabla^2 f : \mathbb{R}^d \to \mathcal{L}(\mathbb{R}^d, \mathcal{L}(\mathbb{R}^d, \mathbb{R}^m)) \simeq \mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d, \mathbb{R}^m)$ of f as the gradient of the gradient ∇f, where $\mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d, \mathbb{R}^m)$ denotes the class of bilinear mappings from $\mathbb{R}^d \times \mathbb{R}^d$ to $\mathbb{R}^m$.

Definition 0.15: Gradient and Hessian through partial derivatives

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a real-valued smooth function. We can define its gradient through partial derivatives:

$$[\nabla f(\mathbf{w})]_j = \frac{\partial f}{\partial w_j}(\mathbf{w}).$$

Note that the gradient $\nabla f(\mathbf{w}) \in \mathbb{R}^d$ has the same size as the input $\mathbf{w}$.

Similarly, we can define the Hessian through partial derivatives:

$$[\nabla^2 f(\mathbf{w})]_{ij} = \frac{\partial^2 f}{\partial w_i \partial w_j}(\mathbf{w}) = \frac{\partial^2 f}{\partial w_j \partial w_i}(\mathbf{w}) = [\nabla^2 f(\mathbf{w})]_{ji},$$

where the second equality holds as long as f is twice-differentiable. Note that the Hessian is a symmetric matrix $\nabla^2 f(\mathbf{w}) \in \mathbb{R}^{d\times d}$ with the same number of rows/columns as the size of the input $\mathbf{w}$.

Example 0.16: Quadratic function

Consider the following quadratic function

$$f : \mathbb{R}^d \to \mathbb{R}, \quad \mathbf{w} \mapsto \mathbf{w}^\top Q\mathbf{w} + \mathbf{p}^\top \mathbf{w} + \alpha, \tag{0.2}$$

where $Q \in \mathbb{R}^{d\times d}$, $\mathbf{p} \in \mathbb{R}^d$, and $\alpha \in \mathbb{R}$. We can write explicitly:

$$f(\mathbf{w}) = \alpha + \sum_{k=1}^d w_k \left[p_k + \sum_{l=1}^d q_{kl} w_l \right],$$

whence follows

$$\begin{aligned}
[\nabla f(\mathbf{w})]_j = \frac{\partial f}{\partial w_j}(\mathbf{w}) &= \sum_{k=1}^d \left[\frac{\partial w_k}{\partial w_j} \left(p_k + \sum_{l=1}^d q_{kl} w_l \right) + w_k \frac{\partial \left(p_k + \sum_{l=1}^d q_{kl} w_l \right)}{\partial w_j} \right] \\
&= \left(p_j + \sum_{l=1}^d q_{jl} w_l \right) + \sum_{k=1}^d w_k q_{kj}
\end{aligned}$$

$$= p_j + [(Q + Q^\top)\mathbf{w}]_j.$$

That is, collectively

$$\nabla f(\mathbf{w}) = \mathbf{p} + (Q + Q^\top)\mathbf{w}.$$

Similarly,

$$[\nabla^2 f(\mathbf{w})]_{ij} = \frac{\partial^2 f}{\partial w_i \partial w_j}(\mathbf{w}) = \frac{\partial \left(p_j + \sum_{l=1}^d q_{jl} w_l + \sum_{k=1}^d w_k q_{kj}\right)}{\partial w_i} = q_{ji} + q_{ij}.$$

That is, collectively

$$\nabla^2 f(\mathbf{w}) \equiv Q + Q^\top.$$

The formula further simplifies if we assume Q is symmetric, i.e. $Q = Q^\top$ (which is usually the case).

As demonstrated above, using partial derivatives to derive the gradient and Hessian is straightforward but tedious. Fortunately, we need only do this once: derive and memorize a few, and then resort to the chain rule.

Example 0.17: Quadratic function (for the brave hearts)

Another way to derive the gradient and Hessian is to guess and then verify the definition (0.1). Using the quadratic function (0.2) again as an example. We need to verify:

$$\begin{aligned} o(\|\Delta\mathbf{w}\|) &= \|f(\mathbf{w}+\Delta\mathbf{w}) - f(\mathbf{w}) - [\nabla f(\mathbf{w})](\Delta\mathbf{w})\| \\ &= \|(\mathbf{w}+\Delta\mathbf{w})^\top Q(\mathbf{w}+\Delta\mathbf{w}) + \mathbf{p}^\top(\mathbf{w}+\Delta\mathbf{w}) - \mathbf{w}^\top Q\mathbf{w} - \mathbf{p}^\top\mathbf{w} - [\nabla f(\mathbf{w})](\Delta\mathbf{w})\| \\ &= \|\mathbf{w}^\top Q\Delta\mathbf{w} + (\Delta\mathbf{w})^\top Q\mathbf{w} + \mathbf{p}^\top\Delta\mathbf{w} - [\nabla f(\mathbf{w})](\Delta\mathbf{w}) + (\Delta\mathbf{w})^\top Q(\Delta\mathbf{w})\|, \end{aligned}$$

whence we guess

$$\nabla f(\mathbf{w}) = \mathbf{p} + (Q + Q^\top)\mathbf{w}$$

so that we can cancel out the first four terms (that are all linear in $\Delta\mathbf{w}$). It is then easy to verify that the remaining term indeed satisfies

$$\|(\Delta\mathbf{w})^\top Q(\Delta\mathbf{w})\| = o(\|\Delta\mathbf{w}\|).$$

Similarly, we can guess and verify the Hessian:

$$\begin{aligned} o(\|\Delta\mathbf{w}\|) &= \|\nabla f(\mathbf{w}+\Delta\mathbf{w}) - \nabla f(\mathbf{w}) - [\nabla^2 f(\mathbf{w})](\Delta\mathbf{w})\| \\ &= \|(Q + Q^\top)\Delta\mathbf{w} - [\nabla^2 f(\mathbf{w})](\Delta\mathbf{w})\|, \end{aligned}$$

from which it is clear that $\nabla^2 f(\mathbf{w}) = Q + Q^\top$ would do.

Definition 0.18: Convex set

A point set $C \subseteq \mathsf{V}$ is called convex if

$$\forall \mathbf{w}, \mathbf{z} \in C,\ [\mathbf{w}, \mathbf{z}] := \{\lambda\mathbf{w} + (1-\lambda)\mathbf{z} : \lambda \in [0,1]\} \subseteq C.$$

Elements in the "interval" $[\mathbf{w}, \mathbf{z}]$ are called convex combinations of $\mathbf{w}$ and $\mathbf{z}$.

Theorem 0.19: Intersection and union of convex sets

Arbitrary intersection and increasing union of convex sets are convex. ■

Thus, $\liminf_\alpha C_\alpha := \cup_\alpha \cap_{\beta\geq\alpha} C_\beta$ is convex. However, arbitrary union hence $\limsup_\alpha C_\alpha := \cap_\alpha \cup_{\beta\geq\alpha} C_\beta$ may not be convex.

Exercise 0.20: Hyperplane and halfspace

Verify the convexity of the hyperplane and halfspace:

$$\partial \mathsf{H}_{\mathbf{w},b} := \{\mathbf{x}\in\mathbb{R}^d : \langle \mathbf{x},\mathbf{w}\rangle + b = 0\}$$
$$\mathsf{H}_{\mathbf{w},b} := \{\mathbf{x}\in\mathbb{R}^d : \langle \mathbf{x},\mathbf{w}\rangle + b \leq 0\}$$

(The partial notation ∂ in front of a set means boundary.)

Exercise 0.21: Polyhedron

A polyhedron is some finite intersection of halfspaces:

$$P := \bigcap_{i=1,\dots,n} \mathsf{H}_{\mathbf{w}_i,b_i} = \{\mathbf{x}\in\mathbb{R}^d : W\mathbf{x}+\mathbf{b}\leq \mathbf{0}\}.$$

Prove any polyhedron is convex.

If a polyhedron is bounded, then we call it a polytope. Prove the following:

- the unit ball of ℓ_1 norm $\{\mathbf{w} : \|\mathbf{w}\|_1 \leq 1\}$ is a polytope;
- the unit ball of ℓ_∞ norm $\{\mathbf{w} : \|\mathbf{w}\|_\infty \leq 1\}$is a polytope;
- the unit ball of ℓ_2 norm $\{\mathbf{w} : \|\mathbf{w}\|_2 \leq 1\}$ is not a polytope.

Theorem 0.22: Convex sets as intersection of halfspaces

Any closed convex set is intersection of halfspaces:

$$C = \bigcap_{i\in I} \mathsf{H}_{\mathbf{w}_i,b_i}$$

■

The subtlety is that for non-polyhedral convex sets, the index set I is infinite. For instance:

$$\{\mathbf{x} : \|\mathbf{x}\|_2 \leq 1\} = \bigcap_{\mathbf{w}:\|\mathbf{w}\|_2=1} \mathsf{H}_{\mathbf{w},-1}.$$

Definition 0.23: Convex function (Jensen, 1905)

The extended real-valued function $f : \mathsf{V} \to (-\infty,\infty]$ is called convex if Jensen's inequality holds:

$$\forall \mathbf{w},\mathbf{z}\in\mathsf{V}, \forall\lambda\in(0,1),\ f(\lambda\mathbf{w}+(1-\lambda)\mathbf{z}) \leq \lambda f(\mathbf{w}) + (1-\lambda) f(\mathbf{z}). \tag{0.3}$$

It is necessary that the (effective) domain of f, i.e. $\operatorname{dom} f := \{\mathbf{w}\in\mathsf{V} : f(\mathbf{w}) < \infty\}$, is a convex set.

We call f *strictly* convex iff the equality in (0.3) holds only when $\mathbf{w} = \mathbf{z}$.

A function f is (strictly) concave iff $-f$ is (strictly) convex.

According to wikipedia, Jensen (Danish) never held any academic position and proved his mathematical results in his spare time.

Jensen, J. L. W. V. (1905). "Om konvekse Funktioner og Uligheder mellem Middelværdier". *Nyt Tidsskrift for Matematik B*, vol. 16, pp. 49–68.

Exercise 0.24: Affine = convex and concave

Recall that a function $\ell : \mathsf{V} \to \mathbb{R}$ is affine if for all $\alpha \in \mathbb{R}$ and $\mathbf{w}, \mathbf{z} \in \mathsf{V}$:

$$\ell(\alpha\mathbf{w} + (1-\alpha)\mathbf{z}) = \alpha\ell(\mathbf{w}) + (1-\alpha)\ell(\mathbf{z}).$$

Prove that a function is both convex and concave iff it is affine.

Remark 0.25: Convexity is remarkably important!

In the above definition of convexity, we have used the fact that V is a vector space, so that we can add vectors and multiply them with (real) scalars. It is quite remarkable that such a simple definition leads to a huge body of interesting results, a tiny part of which we shall be able to present below.

Definition 0.26: Epigraph and sub-level sets

The epigraph of a function f is defined as the set of points lying on or above its graph:

$$\operatorname{epi} f := \{(\mathbf{w}, t) \in \mathsf{V} \times \mathbb{R} : f(\mathbf{w}) \le t\}.$$

It is clear that the epigraph of a function completely determines it:

$$f(\mathbf{w}) = \min\{t : (\mathbf{w}, t) \in \operatorname{epi} f\}.$$

So two functions are equal iff their epigraphs (sets) are the same. Again, we see that functions and sets are somewhat equivalent.

The sub-level sets of f are defined as:

$$\forall t \in \mathbb{R},\ L_t := [\![f \le t]\!] := \{\mathbf{w} \in \mathsf{V} : f(\mathbf{w}) \le t\}.$$

Sub-level sets also completely determine the function:

$$f(\mathbf{w}) = \min\{t : \mathbf{w} \in L_t\}.$$

Each sub-level set is clearly a section of the epigraph.

Exercise 0.27: Calculus for convexity

Prove the following:

- Any norm is convex;
- If f and g are convex, then for any $\alpha, \beta \ge 0$, $\alpha f + \beta g$ is also convex; (what about $-f$?)
- If $f : \mathbb{R}^d \to \mathbb{R}$ is convex, then so is $\mathbf{w} \mapsto f(A\mathbf{w} + \mathbf{b})$;
- If f_t is convex for all $t \in T$, then $f := \sup_{t\in T} f_t$ is convex;
- If $f(\mathbf{w}, t)$ is jointly convex in $\mathbf{w}$ and t, then $\mathbf{w} \mapsto \inf_{t\in T} f(\mathbf{w}, t)$ is convex;

- f is a convex function iff its epigraph $\operatorname{epi} f$ is a convex set;
- If $f : C \to \mathbb{R}$ is convex, then the perspective function $g(\mathbf{w}, t) := tf(\mathbf{w}/t)$ is convex on $C \times \mathbb{R}_{++}$;
- If $f(\mathbf{w}, \mathbf{z})$ is convex, then for any $\mathbf{w}$, $f_{\mathbf{w}} := f(\mathbf{w}, \cdot)$ is convex and similarly for $f^{\mathbf{z}} := f(\cdot, \mathbf{z})$. (what about the converse?)
- All sub-level sets of a convex function are convex, but a function with all sub-level sets being convex need not be convex (these are called quasi-convex functions).

Definition 0.28: Fenchel conjugate function

For any extended real-valued function $f : \mathsf{V} \to (-\infty, \infty]$ we define its Fenchel conjugate function as:

$$f^*(\mathbf{w}^*) := \sup_{\mathbf{w}} \langle \mathbf{w}; \mathbf{w}^* \rangle - f(\mathbf{w}).$$

According to one of the rules in Exercise 0.27, f^* is always a convex function (of $\mathbf{w}^*$).

If $\operatorname{dom} f$ is nonempty and closed, and f is continuous, then

$$f^{**} := (f^*)^* = f.$$

This remarkable property of convex functions will be used later in the course.

Theorem 0.29: Verifying convexity

Let $f : \mathbb{R}^d \to \mathbb{R}$ be differentiable.

- *f is convex iff for all $\mathbf{w}, \mathbf{z}$:*

$$f(\mathbf{z}) \geq f(\mathbf{w}) + \langle \mathbf{z} - \mathbf{w}; \nabla f(\mathbf{w}) \rangle . \tag{0.4}$$

- *If f is twice differentiable, then f is convex iff for all $\mathbf{w}$, $\nabla^2 f(\mathbf{w}) \succeq \mathbf{0}$, i.e. positive semidefinite.*
- *If for all $\mathbf{w}$, $\nabla^2 f(\mathbf{w}) \succ \mathbf{0}$, i.e. positive definite, then f is* strictly *convex.*

Proof: Using the definition of convexity and chain rule:

$$\langle \mathbf{z} - \mathbf{w}; \nabla f(\mathbf{w}) \rangle = \lim_{t \to 0^+} \frac{f(\mathbf{w} + t(\mathbf{z} - \mathbf{w})) - f(\mathbf{w})}{t} \leq \frac{tf(\mathbf{z}) + (1-t)f(\mathbf{w}) - f(\mathbf{w})}{t} = f(\mathbf{z}) - f(\mathbf{w}).$$

Rearranging completes the proof of the only if part of the first item. To see the if part, let $\mathbf{u}_t = t\mathbf{z} + (1-t)\mathbf{w}$ for any $t \in [0, 1]$. Then, using (0.4):

$$\begin{aligned} f(\mathbf{z}) &\geq f(\mathbf{u}_t) + \langle \mathbf{z} - \mathbf{u}_t; \nabla f(\mathbf{u}_t) \rangle = f(\mathbf{u}_t) + (1-t) \langle \mathbf{z} - \mathbf{w}; \nabla f(\mathbf{u}_t) \rangle \\ f(\mathbf{w}) &\geq f(\mathbf{u}_t) + \langle \mathbf{w} - \mathbf{u}_t; \nabla f(\mathbf{u}_t) \rangle = f(\mathbf{u}_t) - t \langle \mathbf{z} - \mathbf{w}; \nabla f(\mathbf{u}_t) \rangle \end{aligned}$$

Multiplying the first and second by t and $(1-t)$ respectively and adding them up proves convexity. ■

The function $f(x) = x^4$ is strictly convex but its Hessian vanishes at its minimum $x = 0$.

Fix any $\mathbf{w}$ and take any direction $\mathbf{d}$. Consider the univariate function $h(t) := f(\mathbf{w} + t\mathbf{d})$. We verify that $h''(t) = \langle \mathbf{d}; \nabla^2 f(\mathbf{w} + t\mathbf{d})\mathbf{d} \rangle \geq 0$. In other words, a convex function has increasing derivative along any direction $\mathbf{d}$ (starting from any point $\mathbf{w}$).

Exercise 0.30: Example convex functions

Prove the following functions are convex (Exercise 0.27 and Theorem 0.29 may be handy):

- affine functions: $f(\mathbf{w}) = \mathbf{w}^\top \mathbf{w} + b$;
- exponential function: $f(w) = \exp(w)$;
- entropy: $f(w) = w \log w$ with $w \geq 0$ and $0 \log 0 := 0$;
- log-sum-exp: $f(\mathbf{w}) = \log \sum_{j=1}^{d} \exp(w_j)$; (its gradient is the so-called `softmax`, to be discussed later)

(You may appreciate the epigraph more after the last exercise.)

Definition 0.31: Optimization problem

Consider a function $f : \mathbb{R}^d \to \mathbb{R}$, we are interested in the minimization problem:

$$\mathfrak{p}_* := \inf_{\mathbf{w} \in C} f(\mathbf{w}), \tag{0.5}$$

where $C \subseteq \mathbb{R}^d$ represents the constraint that $\mathbf{w}$ *must* satisfy.

Historically, the minimization problem (0.5) was motivated by the need of solving nonlinear equations $h(\mathbf{w}) = 0$ (Cauchy, 1847; Curry, 1944), which can be reformulated as a least squares minimization problem $\min_{\mathbf{w}} h^2(\mathbf{w})$. As pointed out by Polyak (1963), the minimization problem itself has become ubiquitous, and often does not have to correspond to solving any nonlinear equation.

We remind that the minimum value $\mathfrak{p}_*$ is an extended real number in $[-\infty, \infty]$ (where $\mathfrak{p}_* = \infty$ iff $C = \emptyset$). When $\mathfrak{p}_*$ is finite, any feasible $\mathbf{w} \in C$ such that $f(\mathbf{w}) = \mathfrak{p}_*$ is called a minimizer. Minimum value always exists while minimizers may not!

Cauchy, M. A.-L. (1847). "Méthode générale pour la résolution des systémes d'équations simultanées". *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, vol. 25, no. 2, pp. 536–538.
Curry, H. B. (1944). "The Method of Steepest Descent for Non-linear Minimization Problems". *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261.
Polyak, B. T. (1963). "Gradient methods for the minimization of functionals". *USSR Computational Mathematics and Mathematical Physics*, vol. 3, no. 4, pp. 643–653.

Definition 0.32: Level-bounded

A function $f : \mathbb{R}^d \to (-\infty, \infty]$ is said to be level bounded iff for all $t \in \mathbb{R}$, the sublevel set $[\![f \leq t]\!]$ is bounded. Equivalently, f is level bounded iff $\|\mathbf{w}\| \to \infty \implies f(\mathbf{w}) \to \infty$.

Theorem 0.33: Existence of minimizer

A continuous and level-bounded function $f : \mathbb{R}^d \to (-\infty, \infty]$ has a minimizer. ■

To appreciate this innocent theorem, let $f(w) = \exp(w)$:

- Is f continuous?
- Is f level-bounded?
- What is the minimum value of f? Does f has a minimizer on $\mathbb{R}$?

Definition 0.34: Extrema of an unconstrained function

Recall that $\mathbf{w}$ is a local minimizer of f if there exists an (open) neighborhood $\mathcal{N}$ of $\mathbf{w}$ so that for all $\mathbf{z} \in \mathcal{N}$ we have $f(\mathbf{w}) \le f(\mathbf{z})$. In case when $\mathcal{N}$ can be chosen to be the entire space $\mathbb{R}^d$, we say $\mathbf{w}$ is a global minimizer of f with the notation $\mathbf{w} \in \operatorname{argmin} f$.

By definition, a global minimizer is always a local minimizer while the converse is true only for a special class of functions (knowns as invex functions). The global minimizer may not be unique (take a constant function) but the global minimum value is.

The definition of a local (global) maximizer is analogous.

Exercise 0.35: Properties of extrema

Prove the following:

- $\mathbf{w}$ is a local (global) minimizer of f iff $\mathbf{w}$ is a local (global) maximizer of $-f$.
- $\mathbf{w}$ is a local (global) minimizer of f iff $\mathbf{w}$ is a local (global) minimizer of $\lambda f + c$ for any $\lambda > 0$ and $c \in \mathbb{R}$.
- If $\mathbf{w}$ is a local (global) minimizer (maximizer) of f, then it is a local (global) minimizer (maximizer) of $g(f)$ for any increasing function $g : \mathbb{R} \to \mathbb{R}$. What if g is *strictly* increasing?
- $\mathbf{w}$ is a local (global) minimizer (maximizer) of a positive function f iff it is a local (global) maximizer (minimizer) of $1/f$.
- $\mathbf{w}$ is a local (global) minimizer (maximizer) of a positive function f iff it is a local (global) minimizer (maximizer) of $\log f$.

Alert 0.36: The epigraph trick

Often, we rewrite the optimization problem

$$\inf_{\mathbf{w} \in C} \ f(\mathbf{w})$$

as the equivalent one:

$$\inf_{(\mathbf{w}, t) \in \operatorname{epi} f \cap (C \times \mathbb{R})} \ t, \tag{0.6}$$

where the newly introduced variable t is jointly optimized with $\mathbf{w}$. The advantages of (0.6) include:

- the objective in (0.6) is a simple, canonical linear function $\langle (\mathbf{0}, 1); (\mathbf{w}, t) \rangle$;
- the constraints in (0.6) may reveal more (optimization) insights.

Theorem 0.37: Local is global under convexity

Any local minimizer of a convex function (over some convex constraint set) is global.

Proof: Let $\mathbf{w}$ be a local minimizer of a convex function f. Suppose there exists $\mathbf{z}$ such that $f(\mathbf{z}) < f(\mathbf{w})$. Take convex combination and appeal to the definition of convexity in Definition 0.23:

$$\forall \lambda \in (0, 1), \ f(\lambda \mathbf{w} + (1 - \lambda)\mathbf{z}) \le \lambda f(\mathbf{w}) + (1 - \lambda) f(\mathbf{z}) < f(\mathbf{w}),$$

contradicting to the local minimality of $\mathbf{w}$. ■

In fact, in Theorem 0.38 we will see that any stationary point of a convex function is its global minimizer.

Theorem 0.38: Fermat's necessary condition for extrema

A necessary condition for $\mathbf{w}$ *to be a local minimizer of a differentiable function* $f : \mathbb{R}^d \to \mathbb{R}$ *is*

$$\nabla f(\mathbf{w}) = \mathbf{0}.$$

(Such points are called stationary, a.k.a. critical.) For convex f *the necessary condition is also sufficient.*

Proof: Suppose t is a local minimizer of a univariate function $g : \mathbb{R} \to \mathbb{R}$, then apply the definition of derivative we may easily deduce that $g'(t) \le 0$ and $g'(t) \ge 0$, i.e. $g'(t) = 0$.

Now if $\mathbf{w}$ is a local minimizer of $f : \mathbb{R}^d \to \mathbb{R}$, then $t = 0$ is a local minimizer of the univariate function $g(t) := f(\mathbf{w} + t\mathbf{d})$ for any $\mathbf{d}$. Apply the previous result for univariate functions and the chain rule:

$$g'(0) = \langle \mathbf{d}; \nabla f(\mathbf{w}) \rangle = 0.$$

Since $\mathbf{d}$ was arbitrary, we must have $\nabla f(\mathbf{w}) = \mathbf{0}$.

For convex function f, the sufficiency follows from (0.4). ■

Take $f(w) = w^3$ and $w = 0$ we see that this necessary condition is not sufficient for nonconvex functions.

For local maximizers, we simply negate the function and apply the theorem to $-f$ instead.

Example 0.39: The difficulty of satisfying a constraint

Consider the trivial problem:

$$\min_{w \ge 1} \ w^2,$$

which admits a unique minimizer $w_\star = 1$. However, if we ignore the constraint and set the derivative to zero we would obtain $w = 0$, which does not satisfy the constraint!

We can apply Theorem 0.38 only when there is no constraint. We will introduce the Lagrangian to "remove" constraints below.

A common trick is to ignore any constraint and apply Theorem 0.38 anyway to derive a "bogus" minimizer $\mathbf{w}_\star$. Then, we verify if $\mathbf{w}_\star$ satisfies the constraint: If it does, then we actually find a minimizer for the *constrained* problem! For instance, hand the constraint above been $w \ge -1$ we would be fine by ignoring the constraint. Needless to say, this trick only works occasionally.

Remark 0.40: Iterative algorithm

The prevailing algorithms in machine learning are iterative in nature, i.e., we will construct a sequence $\mathbf{w}_0, \mathbf{w}_1, \ldots$, which will hopefully "converge" to something we contend with.

Algorithm 0.41: Algorithm of feasible direction

We remind that line 4 below is possible because our universe is a vector space! This testifies again the (obvious?) importance of making machine learning amenable to linear algebra.

Algorithm: Algorithm of feasible direction

Input: $\mathbf{w}_0 \in \operatorname{dom} f \cap C$

1 **for** $t = 0, 1, \ldots$ **do**

2 choose direction $\mathbf{d}_t$ // e.g. $\limsup_{t\to\infty} \langle \mathbf{d}_t; \nabla f(\mathbf{w}_t)\rangle > 0$

3 choose step size $\eta_t > 0$

4 $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \mathbf{d}_t$ // update

5 $\mathbf{w}_{t+1} \leftarrow \mathrm{P}_C(\mathbf{w}_{t+1})$ // optional projection step

Intuitively, Algorithm 0.41 first finds a direction $\mathbf{d}_t$, and then moves the iterate $\mathbf{w}_t$ along the direction. How far we move away from the current iterate $\mathbf{w}_t$ is determined by the step size (assuming $\mathbf{d}_t$ is normalized). To motivate the selection of the direction, apply Taylor's expansion:

$$f(\mathbf{w}_{t+1}) = f(\mathbf{w}_t - \eta_t \mathbf{d}_t) = f(\mathbf{w}_t) - \eta_t \langle \mathbf{d}_t; \nabla f(\mathbf{w}_t)\rangle + o(\eta_t),$$

where $o(\eta_t)$ is the small order term. Clearly, if $\langle \mathbf{d}_t; \nabla f(\mathbf{w}_t)\rangle > 0$ and η_t is small, then $f(\mathbf{w}_{t+1}) < f(\mathbf{w}_t)$, i.e. the algorithm is descending hence making progress. Typical choices for the direction include:

- Gradient Descent (GD): $\mathbf{d}_t = \nabla f(\mathbf{w}_t)$;
- Newton: $\mathbf{d}_t = [\nabla^2 f(\mathbf{w}_t)]^{-1} \nabla f(\mathbf{w}_t)$;
- Stochastic Gradient Descent (SGD): $\mathbf{d}_t = \xi_t$, $\mathbb{E}(\xi_t) = \nabla f(\mathbf{w}_t)$.

Remark 0.42: Randomness helps?

Note that if we start from a stationary point $\mathbf{w}_0$, i.e. $\nabla f(\mathbf{w}_0) = \mathbf{0}$, then gradient descent will not move, no matter what step size we choose! This is why such points are called stationary in the first place. On the other hand, stochastic gradient descent may still move because of the random noise added to it.

Remark 0.43: More on SGD

In machine learning we typically minimize some average loss over a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$:

$$\inf_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}; \mathbf{x}_i, y_i) \quad =: \quad \inf_{\mathbf{w}} \hat{\mathbb{E}} \ell(\mathbf{w}; \mathbf{x}, y), \tag{0.7}$$

where $(\mathbf{x}, y)$ is randomly chosen from the training set $\mathcal{D}$ and the empirical expectation is taken w.r.t. $\mathcal{D}$. Computing the gradient obviously costs $O(n)$ since we need to go through each sample in the training set $\mathcal{D}$. On the other hand, if we take a random sample $(\mathbf{x}, y)$ from the training set, and compute

$$\xi = \nabla \ell(\mathbf{w}; \mathbf{x}, y).$$

Obviously, $\hat{\mathbb{E}}\xi$ equals the gradient but computing ξ is clearly much cheaper. In practice, one usually sample a mini-batch, and compute the (average) gradient over the mini-batch.

Exercise 0.44: Perceptron as SGD

Recall the perceptron update: if $y(\langle \mathbf{x}, \mathbf{w}\rangle + b) \le 0$, then

$$\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}, \; b \leftarrow b + y.$$

Construct a loss function $\ell\big(y(\langle \mathbf{x}, \mathbf{w}\rangle + b)\big)$ in (15.2) so that perceptron reduces to SGD (with step size say $\eta \equiv 1$).

Remark 0.45: Step sizes

Let us mention a few ways to choose the step size η_t:

- Cauchy's rule (Cauchy, 1847), where the existence of the minimizer is assumed:

$$\eta_t \in \operatorname*{argmin}_{\eta \geq 0} f(\mathbf{w}_t - \eta \mathbf{d}_t).$$

- Curry's rule (Curry, 1944), where the finiteness of η_t is assumed:

$$\eta_t = \inf\{\eta \geq 0 : f'(\mathbf{w}_t - \eta \mathbf{d}_t) = 0\}.$$

- Constant rule: $\eta_t \equiv \eta > 0$.
- Summable rule: $\sum_t \eta_t = \infty, \sum_t \eta_t^2 < \infty$, e.g. $\eta_t = O(1/t)$;
- Diminishing rule: $\sum_t \eta_t = \infty, \lim_t \eta_t = 0$, e.g. $\eta_t = O(1/\sqrt{t})$.

The latter three are most common, especially for SGD.

Note that under the condition $\langle \mathbf{d}_t; \nabla f(\mathbf{w}_t) \rangle > 0$, the function $h(\eta) := f(\mathbf{w} - \eta \mathbf{d}_t)$ is decreasing for small positive η. Therefore, Curry's rule essentially selects the smallest local minimizer of h while Cauchy's rule selects the global minimizer of h. Needless to say, Cauchy's rule leads to larger per-step decrease of the function value but is also computationally more demanding. Under both Cauchy's and Curry's rule, we have the orthogonality property:

$$\langle \mathbf{d}_t; \nabla f(\mathbf{w}_{t+1}) \rangle = 0,$$

i.e., the gradient at the next iterate $\mathbf{w}_{t+1}$ is orthogonal to the current direction vector $\mathbf{d}_t$. This explains the zigzag behaviour in gradient algorithms.

Cauchy, M. A.-L. (1847). "Méthode générale pour la résolution des systémes d'équations simultanées". *Comptes rendus hebdomadaires des séances de l'Académie des sciences*, vol. 25, no. 2, pp. 536–538.

Curry, H. B. (1944). "The Method of Steepest Descent for Non-linear Minimization Problems". *Quarterly of Applied Mathematics*, vol. 2, no. 3, pp. 258–261.

Definition 0.46: Lagrangian dual

Consider the canonical optimization problem:

$$\inf_{\mathbf{w} \in C \subseteq \mathbb{R}^d} f(\mathbf{w}) \tag{0.8}$$

$$\text{s.t. } \mathbf{g}(\mathbf{w}) \leq \mathbf{0}, \tag{0.9}$$

$$\mathbf{h}(\mathbf{w}) = \mathbf{0}, \tag{0.10}$$

where $f : \mathbb{R}^d \to \mathbb{R}$, $\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^n$, and $\mathbf{h} : \mathbb{R}^d \to \mathbb{R}^m$. The set C is retained here to represent "simple" constraints that is more convenient to deal with directly than put into either (0.9) or (0.10). (Alternatively, one may always put $C = \mathbb{R}^d$.)

The (nonlinear) constraints (0.9) are difficult to deal with. Fortunately, we can introduce the Lagrangian multipliers (a.k.a. dual variables) $\boldsymbol{\mu} \in \mathbb{R}^n_+$, $\boldsymbol{\nu} \in \mathbb{R}^m$ to move constraints into the Lagrangian:

$$L(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\nu}) := f(\mathbf{w}) + \boldsymbol{\mu}^\top \mathbf{g}(\mathbf{w}) + \boldsymbol{\nu}^\top \mathbf{h}(\mathbf{w}).$$

We can now rewrite the original problem (0.8) as the following fancy min-max problem:

$$\mathfrak{p}_\star := \inf_{\mathbf{w} \in C} \sup_{\boldsymbol{\mu} \geq \mathbf{0}, \boldsymbol{\nu}} L(\mathbf{w}; \boldsymbol{\mu}, \boldsymbol{\nu}). \tag{0.11}$$

(Here $\mathfrak{p}$ stands for primal, as opposed to the dual below.) Indeed, if we choose some $\mathbf{w} \in C$ that does not satisfy either of the two constraints in (0.9)-(0.10), then there exist $\boldsymbol{\mu} \in \mathbb{R}^n_+$ and $\nu \in \mathbb{R}^m$ such that

$L(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\nu}) \to \infty$. Thus, in order to minimize w.r.t. $\mathbf{w}$, we are forced to choose $\mathbf{w} \in C$ to satisfy both constraints in (0.9)-(0.10), in which case $\max_{\boldsymbol{\mu}\geq\mathbf{0},\boldsymbol{\nu}} L(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\nu})$ simply reduces to $f(\mathbf{w})$ (by setting for instance $\boldsymbol{\mu}=\mathbf{0},\boldsymbol{\nu}=\mathbf{0}$). In other words, the explicit constraints in (0.8) have become implicit in (0.11)!

The Lagrangian dual simply swaps the order of min and max:

$$\mathfrak{d}^\star := \sup_{\boldsymbol{\mu}\geq\mathbf{0},\boldsymbol{\nu}} \inf_{\mathbf{w}\in C} L(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\nu}). \tag{0.12}$$

(Here $\mathfrak{d}$ stands for dual.) We emphasize the dimension of the dual variable $\boldsymbol{\mu}$ is the number of inequality constraints while that of $\boldsymbol{\nu}$ is the number of equality constraints; they are different from the dimension of the (primal) variable $\mathbf{w}$. Note also that there is no constraint on $\mathbf{w}$ in the dual problem (0.12) (except the simple one $\mathbf{w} \in C$ which we do not count ☺), implicit or explicit!

In some sense, the Lagrangian dual is "the trick" that allows us to remove complicated constraints and apply Theorem 0.38.

Theorem 0.47: Weak duality

For any function $f : \mathsf{W} \times \mathsf{Z} \to \mathbb{R}$, we have

$$\inf_{\mathbf{w}} \sup_{\mathbf{z}} f(\mathbf{w},\mathbf{z}) \quad \geq \quad \sup_{\mathbf{z}} \inf_{\mathbf{w}} f(\mathbf{w},\mathbf{z}).$$

Proof: Left as exercise. ■

We immediately deduce that the Lagrangian dual (0.12) is a lower bound of the original problem (0.8), i.e. $\mathfrak{p}_\star \geq \mathfrak{d}^\star$. When equality is achieved, we say strong duality holds. For instance, if f, g_i for all $i = 1,\dots,n$, and C are convex, $\mathbf{h}$ is affine, and some mild regularity condition holds (e.g. Slater's condition), then we have strong duality for the Lagrangian in Definition 0.46.

Exercise 0.48: Does the direction matter?

Derive the Lagrangian dual of the following problem:

$$\begin{aligned} &\sup_{\mathbf{w}\in C} f(\mathbf{w}) \\ &\text{s.t. } \mathbf{g}(\mathbf{w}) \geq \mathbf{0} \\ &\quad\ \mathbf{h}(\mathbf{w}) = \mathbf{0}. \end{aligned}$$

(Start with reducing to (0.8) and then see if you can derive directly.)

Definition 0.49: KKT conditions (Karush, 1939; Kuhn and Tucker, 1951)

The inner minimization in the Lagrangian dual (0.12) can usually be solved in closed-form:

$$\mathsf{W}(\boldsymbol{\mu},\boldsymbol{\nu}) := \operatorname*{argmin}_{\mathbf{w}\in C} L(\mathbf{w};\boldsymbol{\mu},\boldsymbol{\nu}). \tag{0.13}$$

When $C = \mathbb{R}^d$, applying Theorem 0.38 we obtain the stationary condition (0.15) below. Plugging any minimizer $\mathsf{W}(\boldsymbol{\mu},\boldsymbol{\nu})$ back into (0.12) we obtain the dual problem:

$$\sup_{\boldsymbol{\mu}\in\mathbb{R}^n_+,\boldsymbol{\nu}\in\mathbb{R}^m} L(\mathsf{W}(\boldsymbol{\mu},\boldsymbol{\nu});\boldsymbol{\mu},\boldsymbol{\nu}) \quad \equiv \quad -\inf_{\boldsymbol{\mu}\in\mathbb{R}^n_+,\boldsymbol{\nu}\in\mathbb{R}^m} -L(\mathsf{W}(\boldsymbol{\mu},\boldsymbol{\nu});\boldsymbol{\mu},\boldsymbol{\nu}). \tag{0.14}$$

Compared to the original problem (0.8) that comes with complicated constraints (0.9)-(0.10), the dual problem (0.14) has only very simple nonnegative constraint on $\boldsymbol{\mu}$. Thus, we may try to solve the Lagrangian dual (0.14) instead! In fact, we have the following necessary conditions for $\mathbf{w}$ to minimize the original problem (0.8) and for $\boldsymbol{\mu},\boldsymbol{\nu}$ to maximize the dual problem (0.14):

- primal feasibility:

$$\mathbf{g}(\mathbf{w}) \leq \mathbf{0},\ h(\mathbf{w}) = \mathbf{0},\ \mathbf{w} \in C;$$

- dual feasibility:

$$\boldsymbol{\mu} \geq \mathbf{0};$$

- stationarity: $\mathbf{w} \in \mathsf{W}(\boldsymbol{\mu}, \boldsymbol{\nu})$; for $C = \mathbb{R}^d$, we simply have

$$\nabla f(\mathbf{w}) + \sum_{i=1}^{n} \mu_i \nabla g_i(\mathbf{w}) + \sum_{j=1}^{m} \nu_j \nabla h_j(\mathbf{w}) = \mathbf{0}; \tag{0.15}$$

- complementary slackness:

$$\langle \mathbf{g}(\mathbf{w}); \boldsymbol{\mu} \rangle = 0.$$

Note that from primal and dual feasibility we always have

$$\forall i = 1, \dots, n, \quad \mu_i g_i(\mathbf{w}) \leq 0.$$

Thus, complementary slackness actually implies equality in all n inequalities above.

When strong duality mentioned in Theorem 9.3 holds, the above KKT conditions are also sufficient!

Karush, W. (1939). "Minima of Functions of Several Variables with Inequalities as Side Constraints". MA thesis. University of Chicago.

Kuhn, H. W. and A. W. Tucker (1951). "Nonlinear programming". In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pp. 481–492.

Algorithm 0.50: Dual gradient descent (Uzawa, 1958)

We can apply Algorithm 0.41 to the dual problem (0.14), equipped with the projection onto nonnegative orthants (see Exercise 2.10 later). After solving the dual, we may attempt to recover the primal through (0.13). In both steps we sidestep any complicated constraints! However, one needs to always keep the following Alert 0.51 in mind.

Uzawa, H. (1958). "Iterative methods for concave programming". In: *Studies in linear and non-linear programming*. Ed. by K. J. Arrow, L. Hurwicz, and H. Uzawa. Standford University Press, pp. 154–165.

Alert 0.51: Instability

A popular way to solve the primal problem (0.8) is to solve the dual (0.14) first. Then, with the optimal dual variable $(\boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star)$ at hand, we "recover" the primal solution by

$$\mathsf{W}(\boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star) = \operatorname*{argmin}_{\mathbf{w} \in C}\ L(\mathbf{w}; \boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star).$$

The minimizing set W always contains all minimizers of the primal problem (0.8). Thus, if W is a singleton, everything is fine. Otherwise W may actually contain points that are not minimizers of the primal problem! Fortunately, we need only verify primal feasibility in order to identify the true minimizers of the primal (0.8) (when strong duality holds). The problem is, in practice, our algorithm only returns one "solution" from W, and if it fails primal feasibility, we may not be able to fetch a different "solution" from W.

Example 0.52: Instability

Let us consider the trivial problem:

$$\min_w 0, \quad \text{s.t.} \quad w \geq 0,$$

whose minimizers are $\mathbb{R}_+$. We derive the Lagrangian dual:

$$\max_{\mu \geq 0} \min_w -\mu w \qquad = \qquad \max_{\mu \geq 0} \begin{cases} 0, & \text{if } \mu = 0 \\ -\infty, & \text{if } \mu > 0 \end{cases}.$$

Clearly, we have $\mu^\star = 0$. Fixing $\mu^\star = 0$ and solving

$$\min_w -\mu^\star w$$

gives us $\mathsf{W} = \mathbb{R}$ which strictly contains the primal solutions $\mathbb{R}_+$! Verifying primal feasibility $w \geq 0$ then identifies true minimizers.

Example 0.53: (Kernel) ridge regression

Recall ridge regression:

$$\begin{aligned} &\min_{\mathbf{w},\mathbf{z}} \ \tfrac{1}{2}\|\mathbf{z}\|_2^2 + \tfrac{\lambda}{2}\|\mathbf{w}\|_2^2 \\ &\text{s.t.} \ \ X\mathbf{w} - \mathbf{y} = \mathbf{z}, \end{aligned}$$

where we *introduced* an "artificial" constraint (and variable). Derive the Lagrangian dual:

$$\max_{\boldsymbol{\nu}} \min_{\mathbf{w},\mathbf{z}} \ \tfrac{1}{2}\|\mathbf{z}\|_2^2 + \tfrac{\lambda}{2}\|\mathbf{w}\|_2^2 + \boldsymbol{\nu}^\top (X\mathbf{w} - \mathbf{y} - \mathbf{z}),$$

Applying Theorem 0.38 to the inner minimization problem:

$$\mathbf{w}_\star = -X^\top \boldsymbol{\nu}/\lambda, \quad \mathbf{z}_\star = \boldsymbol{\nu} \tag{0.16}$$

Plugging it back in (and simplify) we obtain the dual:

$$\max_{\boldsymbol{\nu}} -\tfrac{1}{2\lambda}\|X^\top \boldsymbol{\nu}\|_2^2 - \boldsymbol{\nu}^\top \mathbf{y} - \tfrac{1}{2}\|\boldsymbol{\nu}\|_2^2$$

Applying Theorem 0.38 again we obtain:

$$\boldsymbol{\nu}^\star = -(XX^\top/\lambda + I)^{-1}\mathbf{y}$$

and hence from (0.16) we have

$$\mathbf{w}_\star = X^\top (XX^\top + \lambda I)^{-1}\mathbf{y}.$$

On the other hand, it is known that the solution of ridge regression is

$$\mathbf{w}_\star = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y}$$

Verify the two solutions of $\mathbf{w}_\star$ are the same. (In fact, we have accidentally proved the Sherman-Morrison formula.)

Algorithm 0.54: Gradient descent ascent (GDA)

When we cannot solve the inner minimization problem in the Lagrangian dual (0.12), an alternative is to iteratively perform one gradient descent step on the inner minimization and then perform another gradient ascent step on the outer maximization. This idea can be traced back to (at least) Brown and Neumann (1950) and Arrow and Hurwicz (1958).

More generally, let us consider the min-max optimization problem:

$$\inf_{\mathbf{w}\in\mathsf{W}} \sup_{\mathbf{z}\in\mathsf{Z}} f(\mathbf{w},\mathbf{z}).$$

Algorithm: Gradient descent ascent for min-max

Input: $(\mathbf{w}_0,\mathbf{z}_0)\in \operatorname{dom} f\cap(\mathsf{W}\times\mathsf{Z})$

$s_{-1}=0, (\bar{\mathbf{w}}_{-1},\bar{\mathbf{z}}_{-1})=(\mathbf{0},\mathbf{0})$ // optional

for $t=0,1,\dots$ **do**

  choose step size $\eta_t>0$

  $\mathbf{w}_{t+1}\leftarrow \mathrm{P}_{\mathsf{W}}[\mathbf{w}_t-\eta_t\nabla_{\mathbf{w}} f(\mathbf{w}_t,\mathbf{z}_t)]$ // GD on minimization

  $\mathbf{z}_{t+1}\leftarrow \mathrm{P}_{\mathsf{Z}}[\mathbf{z}_t+\eta_t\nabla_{\mathbf{z}} f(\mathbf{w}_t,\mathbf{z}_t)]$ // GA on maximization

  $s_t\leftarrow s_{t-1}+\eta_t$

  $(\bar{\mathbf{w}}_t,\bar{\mathbf{z}}_t)\leftarrow \frac{s_{t-1}(\bar{\mathbf{w}}_{t-1},\bar{\mathbf{z}}_{t-1})+\eta_t(\mathbf{w}_t,\mathbf{z}_t)}{s_t}$ // averaging: $(\bar{\mathbf{w}}_t,\bar{\mathbf{z}}_t)\leftarrow \sum_{k=1}^t \eta_k(\mathbf{w}_k,\mathbf{z}_k)/\sum_k \eta_k$

Variations of Algorithm 0.54 include (but are not limited to):

- use different step sizes on $\mathbf{w}$ and $\mathbf{z}$;
- use $\mathbf{w}_{t+1}$ in the update on $\mathbf{z}$ (or vice versa);
- use stochastic gradients in both steps;
- after every update in $\mathbf{w}$, perform k updates in $\mathbf{z}$ (or vice versa);

Brown, G. W. and J. v. Neumann (1950). "Solutions of Games by Differential Equations". In: *Contributions to the Theory of Games I*. Ed. by H. W. Kuhn and A. W. Tucker. Princeton University Press, pp. 73–79.

Arrow, K. J. and L. Hurwicz (1958). "Gradient method for concave programming I: Local results". In: *Studies in linear and non-linear programming*. Ed. by K. J. Arrow, L. Hurwicz, and H. Uzawa. Standford University Press, pp. 117–126.

Example 0.55: Vanilla GDA may never converge for any step size ☞ code

Let us consider the following simple problem:

$$\min_{w\in[-1,1]} \max_{z\in[-1,1]} wz \quad\equiv\quad \max_{z\in[-1,1]} \min_{w\in[-1,1]} wz.$$

The left-hand side is equivalent as $\min_{w\in[-1,1]}|w|$ hence with unique minimizer $w^*=0$. Similarly, the right-hand side is equivalent as $\max_{z\in[-1,1]}-|z|$ hence with unique maximizer $z^*=0$. It follows that the optimal saddle-point (equilibrium) is $w^*=z^*=0$.

If we run vanilla (projected) GDA with step size $\eta_t\ge 0$, then

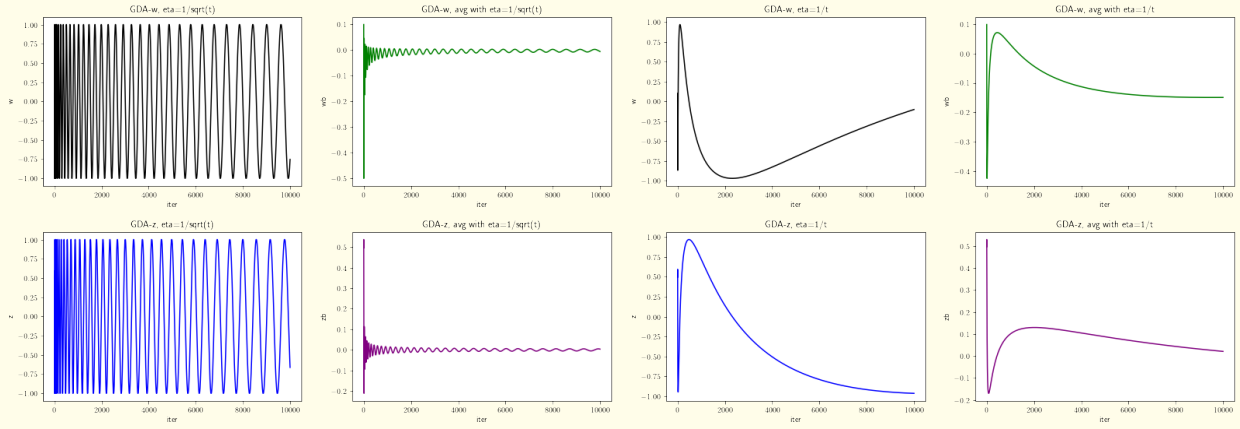
$$w_{t+1}=[w_t-\eta_t z_t]_{-1}^1$$
$$z_{t+1}=[z_t+\eta_t w_t]_{-1}^1,$$

where $[u]_{-1}^1:=(u\wedge 1)\vee(-1)$ is the projection of u onto the interval $[-1,1]$. Thus, we have

$$w_{t+1}^2+z_{t+1}^2\ge 1\wedge[(w_t-\eta_t z_t)^2+(z_t+\eta_t w_t)^2]=1\wedge[(1+\eta_t^2)(w_t^2+z_t^2)]\ge 1\wedge(w_t^2+z_t^2).$$

Therefore, if we do *not* initialize at the equilibrium $w^*=z^*=0$, then the norm of (w_t,z_t) will always be lower bounded by $1\wedge\|\binom{w_0}{z_0}\|>0=\|\binom{w^*}{w^*}\|$. In other words, (w_t,z_t) will not converge to (w^*,z^*).

Indeed, the first and third column plots below verify the above result. Interestingly, with averaging (i.e. Line 6-7 of Algorithm 0.54) and appropriate step size η_t, we can recover convergence (second column), although the convergence can be quite slow (last column) or even misleading (if $\eta_t \to 0$ too fast, such as $\eta_t \propto 1/t^2$; figures not shown).



In fact, we can generalize the above failure to any bilinear matrix game:

$$\min_{\mathbf{w}\in\mathsf{W}} \max_{\mathbf{z}\in\mathsf{Z}} \mathbf{w}^\top A\mathbf{z},$$

where W and Z are closed sets and $A \in \mathbb{R}^{d\times d}$ is nonsingular. Suppose there exists $\epsilon > 0$ so that any equilibrium $(\mathbf{w}^*, \mathbf{z}^*)$ is ϵ away from the boundary: $\operatorname{dist}(\mathbf{w}^*, \partial\mathsf{W}) \wedge \operatorname{dist}(\mathbf{z}^*, \partial\mathsf{Z}) > \epsilon$, then we know $A\mathbf{z}^* = \mathbf{0}$ hence $\mathbf{z}^* = \mathbf{0}$ and similarly $\mathbf{w}^* = \mathbf{0}$. It follows that vanilla projected gradient

$$\mathbf{w}_{t+1} = \mathrm{P}_{\mathsf{W}}(\mathbf{w}_t - \eta_t A\mathbf{z}_t)$$
$$\mathbf{z}_{t+1} = \mathrm{P}_{\mathsf{Z}}(\mathbf{z}_t + \eta_t A^\top \mathbf{w}_t)$$

will not converge to any equilibrium. Indeed, for vanilla gradient to converge, the projection will eventually be vacuous (otherwise we are ϵ away), but then for the equilibrium $(\mathbf{w}^* = \mathbf{0}, \mathbf{z}^* = \mathbf{0})$:

$$\|\mathbf{w}_{t+1}\|_2^2 + \|\mathbf{z}_{t+1}\|_2^2 = \|\mathbf{w}_t\|_2^2 + \|\mathbf{z}_t\|^2 + \eta_t^2 \left\| \begin{bmatrix} \mathbf{0} & A^\top \\ A & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{z}_t \\ \mathbf{w}_t \end{bmatrix} \right\|_2^2 = (1 + \eta_t^2 \sigma_{\min}^2(A))(\|\mathbf{w}_t\|_2^2 + \|\mathbf{z}_t\|^2),$$

which is strictly lower bounded by $\|\mathbf{w}_0\|_2^2 + \|\mathbf{z}_0\|_2^2$ if the starting point $(\mathbf{w}_0, \mathbf{z}_0)$ does not coincide with the equilibrium.

Algorithm 0.56: Alternating

The following alternating algorithm is often applied in practice for solving the joint minimization problem:

$$\inf_{\mathbf{w}\in\mathsf{W}} \inf_{\mathbf{z}\in\mathsf{Z}} f(\mathbf{w}, \mathbf{z}).$$

Algorithm: Alternating minimization for min-min

Input: $(\mathbf{w}_0, \mathbf{z}_0) \in \operatorname{dom} f \cap (\mathsf{W} \times \mathsf{Z})$

1 **for** $t = 0, 1, \ldots$ **do**

2 $\quad \mathbf{w}_{t+1} \leftarrow \operatorname{argmin}_{\mathbf{w}\in\mathsf{W}} f(\mathbf{w}, \mathbf{z}_t)$ `// exact minimization`

3 $\quad \mathbf{z}_{t+1} \leftarrow \operatorname{argmin}_{\mathbf{z}\in\mathsf{Z}} f(\mathbf{w}_{t+1}, \mathbf{z})$ `// exact minimization`

It is tempting to adapt alternating to solve min-max problems. The resulting algorithm, when compared to dual gradient (see Algorithm 0.50), is more aggressive in optimizing $\mathbf{z}$: dual gradient only takes a gradient

ascent step while the latter finds the exact maximizer. Surprisingly, being more aggressive (and spending more effort) here actually hurts (sometimes).

Example 0.57: When to expect alternating to work?

Let us consider the simple problem

$$\min_{w\in[-1,1]} \min_{z\in[-1,1]} wz,$$

where two minimizers $\{(1,-1),(-1,1)\}$ exist. Let us apply the alternating Algorithm 0.56. Choose any $w>0$, we obtain

$$z\leftarrow -1, w\leftarrow 1, z\leftarrow -1,$$

which converges to an optimal solution after 1 iteration!

To compare, let us consider the "twin" problem:

$$\min_{w\in[-1,1]} \max_{z\in[-1,1]} wz,$$

where a unique equilibrium $(w^*,z^*)=(0,0)$ exists. Choose any $w>0$ (the analysis for $w<0$ is similar). Let us perform the inner maximization exactly to obtain $z=1$. With $z=1$ in mind, we perform the outer inner minimization exactly to obtain $w=-1$. Keep iterating, we obtain the cycle

$$z\leftarrow -1, w\leftarrow 1, z\leftarrow 1, w\leftarrow -1, z\leftarrow -1,$$

which is bounded away from the equilibrium! Of course, if we perform averaging along the trajectory we again obtain convergence. Dual gradient essentially avoids the oscillating behaviour of alternating by (implicitly) averaging.