Professor Amir Ali Ahmadi Scribe: Amy Hua

*** denotes a section that is currently being edited.

Contents

1 Introduction

1.1 What is optimization?

Roughly, we can think of optimization as the science of making the most out of every situation. Common themes arise:

- You make decisions and choose one of many alternatives.
- You hope to maximize or minimize something (you have an objective).
- You cannot make arbitrary decisions. Life comes with constraints.

We will be learning techniques for dealing with problems that appear daily in industry, science, and engineering. We typically model a scenario with a precise mathematical description and care about finding the best solution; if we can't find the best solution, we want to know how far off we are. Examples of optimization problems include:

- Finance: In what proportions should we invest in 500 stocks? We aim to maximize return or minimize risk. Constraints can be that no more than a quarter of your money can be in a single stock, transaction costs must be less than \$70, and the return rate must be greater than 2%.
- Control engineering: How do we drive an autonomous vehicle from A to B? We can aim to minimize fuel consumption or minimize travel time. Constraints can be that the speed must be less than 40 mph and that the path must be smooth.
- Machine learning: How do we assign likelihoods to emails being spam? We want to minimize the probability of a false positive or to penalize overfitting on the training set. Constraints can be that the misclassification error on the training set must be less than 5% and that the probability of a false negative must be less than 0.15.

This class will provide a broad introduction to "optimization from a computational viewpoint." Optimization and computing are very close areas of applied math. Several basic topics in scientific computing are either special cases or fundamental ingredients of more elaborate optimization algorithms (examples include least squares, root finding, solving linear systems, solving linear inequalities, approximation and fitting, matrix factorizations, conjugate gradients, and more).

Let's play two optimization games:

1. *Let's ship some oil together!* The rules are to ship as much oil from the Source node to the Target node, given two constraints: we must satisfy that the capacities of edges are not exceeded and maintain conservation of flow (what goes in must come out, for each node, except for Source and Target). There is a way to automate finding the right cut. Our proof of optimality is to examine the max cut. (Review the Ford-Fulkerson algorithm.) Note that this game is solvable in polynomial time.
2. *Two final exams in one day? No thanks.* Here's the scenario: the department chair would like to schedule final exams for a number of courses such that as many exams as possible are on the same day, but no student should have more than one exam on

the same day. The nodes of the graph are the courses, and there is an edge between two nodes if and only if there is at least one student taking both courses. To schedule as many exams as possible on the same day, we are looking for the largest collection of nodes such that no two nodes share an edge.

How do we prove that the solution is optimal? How do we prove that the optimal solution is unique? Of course, there is always a proof: we can try all subsets with seven nodes (3432 possibilities), but this has exponential runtime. This problem is NP-complete.

Notice that with these two games, finding the best solution might be too much to hope for; however, techniques from optimization (particularly from convex optimization) often allow us to find high-quality solutions with performance guarantees.

1.2 Modelling problems as mathematical programs

We can formulate the first game in terms of **decision variables**, an **objective function**, and a set of **constraints**.

For the first game, with nodes $S, A, B, C, \ldots, G, T$, we can represent our decision variables as $x_{SA}, x_{AB}, \ldots, x_{GT}$, the capacities of the edges between pairs of nodes. Our objective function and constraints are represented by the optimization problem

$$\begin{aligned} \max_{x_{SA}, x_{SB}, x_{SC}, \ldots} \quad & x_{SA} + x_{SB} + x_{SC} \\ \text{s.t.} \quad & x_{SA}, x_{AB}, \ldots, x_{GT} \geq 0 \\ & x_{SA} \leq 6, x_{AB} \leq 2, \ldots, x_{GT} \leq 12 \\ & x_{SA} = x_{AB} + x_{AD} + x_{AE} \\ & \ldots \end{aligned}$$

For the second game, our decision variables are $x_1, x_2, \ldots, x_{14},\ x_i \in \{0, 1\},\ i = 1, 2, \ldots, 14$ (whether a node is included or not). Our objective function is $\max \sum_{i=1}^{14} x_i$ subject to the constraints $x_i(1 - x_i) = 0, i = 1, \ldots, 14, x_1 + x_2 \leq 1, x_1 + x_8 \leq 1, \ldots$—that we only include one node per edge. We can also represent the constraints with $x_i x_j = 0$ if nodes i and j are connected or with $x_i + x_j \leq 1$ if nodes i and j are connected.

Caution: just because we can write something as a mathematical program or optimization problem, it doesn't mean that we can solve it. An example: we can write Fermat's Last Theorem (for any power $n > 2$, $x^n + y^n = z^n$ has no solution over positive integers) as an optimization problem:

$$\begin{aligned} \min_{x,y,z,n} \quad & (x^n + y^n - z^n)^2 \\ \text{s.t.} \quad & x \geq 1, y \geq 1, z \geq 1, n \geq 3 \\ & \sin^2(\pi x) + \sin^2(\pi y) + \sin^2(\pi z) + \sin^2(\pi n) = 0. \end{aligned}$$

However, there are no solutions over the positive integers for $n \geq 3$. In fact, if you show that the optimal value of the optimization problem is non-zero, you prove Fermat's conjecture!

2 Mathematical review

Remark 2.1. About notation, context will be important. Generally, vectors will be represented with lowercase letters, while matrices will be represented with uppercase letters. All vectors used in this class will be column vectors.

2.1 Linear algebra

We will briefly review the concepts of inner products, norms, eigenvalues and eigenvectors, and positive semidefinite matrices.

2.1.1 Inner products and norms

Definition 2.2. (Inner product). A function $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is an inner product if it satisfies the following properties:

1. **Positivity**: $\langle x, x \rangle \geq 0$ and $\langle x, x \rangle = 0 \iff x = 0$.
2. **Symmetry**: $\langle x, y \rangle = \langle y, x \rangle$.
3. **Additivity**: $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$.
4. **Homogeneity**: $\langle rx, y \rangle = r\langle x, y \rangle \ \forall \ r \in \mathbb{R}$.

Remark 2.3. Let's manipulate some norms:

$$\langle x, y + z \rangle = \langle x, y \rangle + \langle x, z \rangle,$$

$$\langle x, y + z \rangle \overset{(2)}{=} \langle y + z, x \rangle \overset{(3)}{=} \langle y, x \rangle + \langle z, x \rangle \overset{(2)}{=} \langle x, y \rangle + \langle x, z \rangle,$$

$$\langle x, ry \rangle = r\langle x, y \rangle.$$

Remark 2.4. For $\langle \cdot, \cdot \rangle : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, the inner product is just $\langle x, y \rangle = x \cdot y$.

For $\langle \cdot, \cdot \rangle : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$, the inner product is

$$\langle x, y \rangle = x_1 y_1 + x_2 y_2 = \text{length}(x) \cdot \text{length}(y) \cdot \cos(\theta),$$

where θ is the angle between vectors x and y. Note that the inner product is positive when $\theta < 90$ degrees, negative when $\theta > 90$ degrees, and zero when $\theta = 90$ degrees.

Example 2.5. (Dot product, or standard inner product, or Euclidean inner product). $\langle x, y \rangle = \sum_{i=1}^n x_i y_i = x^T y$, where $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n$. We can check that all four properties are satisfied.

$$\langle x, x \rangle = \sum_{i=1}^n x_i^2 \geq 0 \text{ and } \langle x, x \rangle = 0 \iff x = 0.$$

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

$$\langle x+y,z\rangle = \sum_{i=1}^{n}(x_i+y_i)z_i = \sum_{i=1}^{n} x_i z_i + \sum_{i=1}^{n} y_i z_i = \langle x,z\rangle + \langle y,z\rangle.$$

$$\langle rx,y\rangle = \sum_{i=1}^{n} r x_i y_i = r\sum_{i=1}^{n} x_i y_i = r\langle x,y\rangle \ \forall\ r\in\mathbb{R}.$$

Example 2.6. ($n=2$). Claim: The following is an inner product.

$$\langle x,y\rangle = 5x_1y_1 + 8x_2y_2 - 6x_1y_2 - 6x_2y_1.$$

Why?

$$\langle x,x\rangle = (x_1-2x_2)^2 + (2x_1-2x_2)^2 \geq 0.$$

(Note that we still need to prove the other properties of inner product.)

Definition 2.7. (Orthogonal). We say that $x,y\in\mathbb{R}^n$ are orthogonal (with respect to some inner product) if $\langle x,y\rangle = 0$. Note that the zero vector is orthogonal to every other vector, by this definition.

Definition 2.8. (Vector norm). A function $f:\mathbb{R}^n\to\mathbb{R}$ is a norm if the following are satisfied:

1. **Positivity**: $f(x)\geq 0\ \forall\ x$ and $f(x)=0 \iff x=0$.
2. **Homogeneity**: $f(rx) = |r|f(x)\ \forall\ r\in\mathbb{R}$.
3. **Triangle inequality**: $f(x+y)\leq f(x)+f(y)$.

Example 2.9. Some examples of norms, where $x\in\mathbb{R}^n$, include:

- ℓ_1-norm: $\|x\|_1 = \sum_{i=1}^{n}|x_i|$.
- ℓ_2-norm: $\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$.
- ℓ_p-norm: $\|x\|_p = \left(\sum_{i=1}^{n}|x_i|^p\right)^{1/p}$ for $p\geq 1$.
- ℓ_∞-norm: $\|x\|_\infty = \max_i |x_i|$.
- Quadratic norms: $\|x\|_Q = \sqrt{x^TQx}$ is a norm for any $Q\succeq 0$. $\|x\|_2$ is a special case where $Q=I$.

Remark 2.10. $\|x\|_1 \geq \|x\|_2 \geq \|x\|_\infty$.

Remark 2.11. When no index is specified on a norm (e.g., $\|\cdot\|$), the assumed norm is the Euclidean norm $\|\cdot\|_2$.

Remark 2.12. Given any inner product $\langle x,y\rangle$, one can construct a norm given by $\|x\| = \sqrt{\langle x,x\rangle}$. So the ℓ_2-norm comes from the standard inner product. But not every norm comes from an inner product (e.g., $\|\cdot\|_1$).

Claim 2.13. $\|x\|_1 = \sum_{i=1}^{n}|x_i|$ *is a norm.*

Proof. 1. Positivity obviously holds.

2. $\|rx\|_1 = \sum_{i=1}^n |rx_i| = |r| \sum_{i=1}^n |x_i| = |r| \|x\|_1$.
3. $\|x+y\|_1 = \sum_{i=1}^n |x_i + y_i| \leq \sum_{i=1}^n (|x_i| + |y_i|) = \sum |x_i| + \sum |y_i| = \|x\|_1 + \|y\|_1$.

□

Definition 2.14. (Cauchy-Schwarz inequality). Let $\|x\| = \sqrt{\langle x, x \rangle}$. Then,

$$|\langle x, y \rangle| \leq \|x\| \cdot \|y\|.$$

Furthermore, we have $|\langle x, y \rangle| = \|x\| \cdot \|y\|$ iff $x = \alpha y$ for some $\alpha \in \mathbb{R}$.

Proof. We have that $\|x\|^2 = \langle x, x \rangle$.

- *Case 1*: Suppose $\|x\| = \|y\| = 1$.

  $$\|x - y\|^2 \geq 0 \Longrightarrow \langle x - y, x - y \rangle \geq 0 \Longrightarrow \langle x, x \rangle + \langle y, y \rangle - 2\langle x, y \rangle \geq 0 \Longrightarrow \langle x, y \rangle \leq 1.$$

- *Case 2*: Suppose $\|x\| \neq 1$ or $\|y\| \neq 1$. Now consider two vectors, which have norm 1: $\frac{x}{\|x\|}$ and $\frac{y}{\|y\|}$.

  $$\left\| \frac{x}{\|x\|} \right\| = \frac{1}{\|x\|} \|x\| = 1 \Longrightarrow \frac{1}{\|x\|} \cdot \frac{1}{\|y\|} \langle x, y \rangle \leq 1 \Longrightarrow \langle x, y \rangle \leq \|x\| \cdot \|y\|.$$

  We have proven this for any x, so we can also write:

  $$\langle -x, y \rangle \leq \|x\| \cdot \|y\| \Longrightarrow \langle x, y \rangle \geq -\|x\| \cdot \|y\|.$$

  These two results $\Longrightarrow |\langle x, y \rangle| \leq \|x\| \cdot \|y\|$.

□

Definition 2.15. (Matrix norm). We can also define norms on matrices—functions $\|\cdot\| : \mathbb{R}^{m \times n} \to \mathbb{R}$ that satisfy the same properties as in the definition of vector norms. Note that the vector norm and the matrix norm have the same notation, so context is important.

Definition 2.16. (Induced norm). Consider any vector norm $\|\cdot\|_* : \mathbb{R}^k \to \mathbb{R}$. The induced norm $\|\cdot\|_* : \mathbb{R}^{m \times n} \to \mathbb{R}$ on the space of $m \times n$ matrices is defined as

$$\|A\|_* = \max\{\|Ax\|_* : x \in \mathbb{R}^n \text{ and } \|x\|_* = 1\}.$$

Definition 2.17. (Frobenius norm). The Frobenius norm $\|\cdot\|_F : \mathbb{R}^{m \times n} \to \mathbb{R}$ is defined by

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2}.$$

The Frobenius norm is an example of a matrix norm that is not induced by a vector norm. $\|I_{n \times n}\|_* = 1$ for any induced norm $\|\cdot\|_*$ but $\|I_{n \times n}\|_F = n$.

Definition 2.18. (Submultiplicative norm). A matrix norm is submultiplicative if it satisfies the following inequality:

$$\|AB\| \leq \|A\| \cdot \|B\|.$$

Some facts follow:

- All induced norms are submultiplicative.
- The Frobenius norm is submultiplicative.
- Not every matrix norm is submultiplicative: $\|\cdot\| : A \to \max_{i,j} |a_{i,j}|$, for instance.

Take $A = B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$. Then $\|A\| \cdot \|B\| = 1 \cdot 1 = 1$.
But $AB = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}$. Hence $\|AB\| = 2$ and $\|A\| \cdot \|B\| < \|AB\|$.

2.1.2 Eigenvalues and eigenvectors

Definition 2.19. (Eigenvalue and eigenvector). Given a matrix $A \in \mathbb{R}^{n \times n}$, we say that a scalar $\lambda \in \mathbb{C}$ is an eigenvalue of A if it satisfies $Av = \lambda v$ for some vector $v \neq 0$, which is called the eigenvector. Note that A is $n \times n$, v is $n \times 1$, and λ is 1×1, and that the eigenvalues and eigenvectors can be complex. To find the eigenvalue and eigenvector, solve for λ:

$$Av - \lambda v = 0 \implies (A - \lambda I)v = 0 \implies \det(A - \lambda I) = 0,$$

which gives us a degree n characteristic polynomial equation in λ. Sanity checks that we can perform are: the sum of the eigenvalues must equal $\mathrm{Tr}(A)$, and the product of the eigenvalues must equal $\det(A)$.

Definition 2.20. (Symmetric matrix). A matrix A is symmetric if $A = A^T$.

Theorem 2.21. *All eigenvalues of a symmetric matrix are real.*

Proof. Assume $A = A^T$. Suppose that λ is an eigenvalue, so $Ax = \lambda x$ for some eigenvector $x \neq 0$. Note that λ can be $a + ib$; we will show that $b = 0$.

$$x = \begin{pmatrix} a_1 + ib_1 \\ \vdots \\ a_n + ib_n \end{pmatrix}.$$

We can conjugate x and λ, resulting in $x^* = (a_1 - ib_1, \ldots, a_n - ib_n)^T$ and $\lambda^* = a - ib$.

$$\begin{aligned} Ax &= \lambda x \\ \implies x^* A^* &= \lambda^* x^* \\ \implies x^* A &= \lambda^* x^* \\ \implies x^* Ax &= \lambda x^* x \\ \implies x^* Ax &= \lambda^* x^* x \\ \implies \lambda x^* x &= \lambda^* x^* x \\ \implies (\lambda - \lambda^*) x^* x &= 0. \end{aligned}$$

By assumption, $x \neq 0$, so $\lambda - \lambda^* = 0 \implies \lambda = \lambda^* \implies a - ib = a + ib \implies b = 0$. □

Remark 2.22. Any real symmetric $n \times n$ matrix has a set of n real eigenvectors that are mutually orthogonal.

Definition 2.23. A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is:

- **Positive semidefinite (psd)** ($A \succeq 0$) if $x^T A x \geq 0 \ \forall \ x \in \mathbb{R}^n$. This implies that all diagonals of A are ≥ 0 (see the following question).
- **Positive definite (pd)** ($A \succ 0$) if $x^T A x > 0 \ \forall \ x \in \mathbb{R}^n, \ x \neq 0$.
- **Negative semidefinite (nsd)** ($A \preceq 0$) if $-A$ is psd.
- **Negative definite (nd)** ($A \prec 0$) if $-A$ is pd.
- **Indefinite** if it is neither psd nor nsd.

Question 2.24. Is $\begin{pmatrix} 5 & 2 \\ 2 & -3 \end{pmatrix}$ positive semidefinite? No: take $x = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \implies x^T A x = -3 < 0$. This indicates that all diagonal entries of A must be nonnegative if $A \succeq 0$.

Theorem 2.25. *A symmetric matrix A is positive semidefinite (i.e., $A \succeq 0$) if and only if all eigenvalues of $A \geq 0$.*

Proof. (Positive semidefiniteness). *Forward direction*: If $\lambda < 0$, A cannot be positive semidefinite since $Ax = \lambda x \implies x^T A x = \lambda x^T x$, and $x^T x = \|x\|^2 > 0$. *Backward direction*: □

Theorem 2.26. *A is positive definite ($A \succ 0$) if and only if all eigenvalues of $A > 0$.*

Theorem 2.27. *As a result of the previous two theorems, A is negative semidefinite (resp. negative definite) if and only if all eigenvectors of A are nonpositive (reps. negative).*

Definition 2.28. (Sylvester's criterion).

- A is positive definite if and only if all n leading principal minors of $A > 0$ (and all eigenvalues of $A > 0$). A **leading principal minor** is any submatrix that takes up an upper-left part of A. There are always n leading principal minors.

  In the 2×2 case, where $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \succ 0$, we must have the following:

  - $a > 0$,
  - $\det(A) = ac - b^2 > 0$.

  In the 3×3 case, where $A = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix} \succ 0$, we must have the following:

  - $a > 0$
  - $ad - b^2 > 0$
  - $\det(A) > 0$.

- A is positive semidefinite if and only if all $2^n - 1$ principal minors of $A \geq 0$ (and all eigenvalues of $A \geq 0$). A **principal minor** is the determinant of a sublock of A such that the same rows and columns are taken (there are $2^n - 1$ principal minors because you can choose to include or not include a row/column, and you cannot choose 0 rows/columns).

  In the 2×2 case, where $A = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \succeq 0$, there are 3 principal minors to track:

  - $a \geq 0$,
  - $c \geq 0$,
  - $\det(A) = ac - b^2 \geq 0$.

  In the 3×3 case, let $A = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}$. We must have the following:

  - $a \geq 0$
  - $d \geq 0$
  - $f \geq 0$
  - $ad - b^2 \geq 0$
  - $af - c^2 \geq 0$
  - $df - e^2 \geq 0$
  - $\det(A) \geq 0$.

2.2 Multivariable calculus

We will briefly review the concepts of affine and quadratic functions, continuity, gradients, Hessians, and Taylor expansion. In addition, we will cover the proper terminology used in optimization.

2.2.1 Elements of differential calculus

Definition 2.29. (Continuous). We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is continuous at a point $a \in \text{dom}(f)$, if

$$\forall\ \varepsilon \geq 0,\ \exists\ \delta \geq 0 \text{ s.t. } \|x - a\| \leq \delta \Longrightarrow \|f(x) - f(a)\| \leq \varepsilon.$$

(Recall that $\|x\| = \sqrt{x^T x} = \sqrt{\sum_{i=1}^n x_i^2}$.) A function $f : \mathbb{R}^n \to \mathbb{R}$ is continuous if it is continuous at every point over its domain.

Remark 2.30. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ given as $f = \begin{pmatrix} f_1 \\ \vdots \\ f_m \end{pmatrix}$ is continuous if and only if each entry $f_i : \mathbb{R}^n \to \mathbb{R}$ is continuous.

There are three classes of functions that we deal with; typically, these functions are $f : \mathbb{R}^n \to \mathbb{R}$.

1. **Linear function**: $f(x) = c^T x$ for some vector $c \in \mathbb{R}^n,\ c \neq 0$. Note that $f(\alpha x) = \alpha f(x)\ \forall\ x \in \mathbb{R}^n,\ \forall\ \alpha \in \mathbb{R}$. Any linear function can be represented as $f(x) = Ax$, where $A \in \mathbb{R}^{m \times n}$. The special case when $m = 1$, how we first defined a linear function, will be encountered a lot.

2. **Affine function**: $f(x) = c^T x + b$ for some vector $c \in \mathbb{R}^n,\ c \neq 0,\ b \in \mathbb{R}$. These functions have an offset, so they do not need to go through the origin. In general, a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is affine if there exists a linear function $L : \mathbb{R}^n \to \mathbb{R}^m$ and a vector $y \in \mathbb{R}^m$ such that $f(x) = L(x) + y\ \forall\ x \in \mathbb{R}^n$.

3. **Quadratic function**: $f(x) = x^T Q x + c^T x + b$ for some $Q \in \mathbb{R}^{n \times n},\ c \in \mathbb{R}^n,\ b \in \mathbb{R}$, where Q is symmetric. We call a function $f : \mathbb{R}^n \to \mathbb{R}$ that can be represented as $f(x) = x^T Q x$ a **quadratic norm**; Q is an $n \times n$ matrix that we can assume to be symmetric (i.e., $Q = Q^T$) without loss of generality[1]. Thus, a quadratic function is a function that is the sum of a quadratic norm and an affine function.

Example 2.31. (Quadratic function). $f(x) = 3x_1^2 + 4x_1x_2 - 5x_2^2 + 6x_1 - 2x_2 + 7$. We recognize immediately that $f(x)$ is quadratic, due to the $6x_1 - 2x_2 + 7$ affine term. What are Q, c, and b?

First, note that $b = 7$.

Next, since $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, we see that $c = \begin{pmatrix} 6 \\ -2 \end{pmatrix}$, as $c^T x = \begin{pmatrix} 6 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$.

Finally, $Q = \begin{pmatrix} 3 & 4/2 \\ 4/2 & -5 \end{pmatrix}$. Note that we can look at the coefficients of the perfect powers (e.g., x_1^2, x_2^2) and put them on the diagonal of Q. We divide 4 by 2 because, otherwise, we would get 8 when we multiply it out.

Let's check our work:

$$\begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 3 & 2 \\ 2 & -5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 3x_1 + 2x_2 \\ 2x_1 - 5x_2 \end{pmatrix} = 3x_1^2 + 2x_1x_2 + 2x_1x_2 - 5x_2^2.$$

Definition 2.32. (Partial derivative). The partial derivative of a function $f : \mathbb{R}^n \to \mathbb{R}$ with respect to a variable x_i is given by

$$\frac{\partial f}{\partial x_i} = \lim_{\alpha \to 0} \frac{f(x + \alpha e_i) - f(x)}{\alpha},$$

where e_i is the i-th standard basis vector in $\mathbb{R}^n$.

Definition 2.33. (Jacobian matrix). For a function $f : \mathbb{R}^n \to \mathbb{R}^m$ given as $f(x) = (f_1(x), \ldots, f_m(x))^T$, the Jacobian matrix is the $m \times n$ matrix of first partial derivatives:

$$J_f(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{pmatrix}.$$

[1] If Q is not symmetric, we can define $Q_0 = \frac{1}{2}(Q + Q^T)$, which is a symmetric matrix, and we would have $x^T Q x = x^T Q_0 x$. This follows because $x^T \left(\frac{1}{2}(Q + Q^T)\right) x = \frac{1}{2} x^T Q x + \frac{1}{2} x^T Q^T x = \frac{1}{2}\langle x, Qx \rangle + \frac{1}{2}\langle Qx, x \rangle = \langle x, Qx \rangle = x^T Q x$.

The first order approximation of f near a point x_0 is obtained using the Jacobian matrix: $A(x) = f(x_0) + J_f(x_0)^T(x - x_0)$. Note that this is an affine function of x.

Definition 2.34. (Gradients). For a function $f : \mathbb{R}^n \to \mathbb{R}$, the gradient $\nabla f(x)$ is a vector in $\mathbb{R}^n$ defined as

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix} = J_f(x)^T.$$

This vector is very important in optimization. At every point, the gradient vector points in a direction where the function grows most rapidly.

Definition 2.35. (Hessian). For a function $f : \mathbb{R}^n \to \mathbb{R}$ that is twice differentiable, the Hessian matrix $\nabla^2 f(x)$ of f at a point x is an $n \times n$ symmetric matrix given by

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_1} \\ \vdots & & \vdots \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}.$$

Note that the Hessian matrix, when f is twice continuously differentiable, is always symmetric since partial derivatives commute: $\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$.

Example 2.36. $f(x) = x_1 x_2^3 + 3x_1^2 x_2$.

$$\nabla f(x) = \begin{pmatrix} x_2^3 + 6x_1 x_2 \\ 3x_1^2 + 3x_1 x_2^2 \end{pmatrix},$$

$$\nabla^2 f(x) = \begin{pmatrix} 6x_2 & 3x_2^2 + 6x_1 \\ 3x_2^2 + 6x_1 & 6x_1 x_2 \end{pmatrix}.$$

Definition 2.37. (Level sets). Given a function $f : \mathbb{R}^n \to \mathbb{R}$ and a scalar $\alpha \in \mathbb{R}$,

- The α**-level set** of f is given by $\{x \in \mathbb{R}^n \mid f(x) = \alpha\}$.
- The α**-sublevel set** of f is given by $\{x \in \mathbb{R}^n \mid f(x) \leq \alpha\}$.

In homeworks, we plot level sets in MATLAB using `contour` and `ezplot`.

Remark 2.38. At any point $x_0 \in \mathbb{R}^n$, the gradient vector $\nabla f(x_0)$ is orthogonal to the tangent to the level set going through x_0.

Remark 2.39. (Gradients and Hessians of affine and quadratic functions).
$f(x) = c^T x + b \to \nabla f(x) = c$ and $\nabla^2 f(x) = 0_{n \times n}$.
$f(x) = x^T Q x + c^T x + b \to \nabla f(x) = 2Qx + c$ and $\nabla^2 f(x) = 2Q$, an $n \times n$ symmetric matrix.

Remark 2.40. The following outline some practical rules for differentiation:

- The sum rule: If $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^m$, then $J_{f+g}(x) = J_f(x) + J_g(x)$.

- The product rule: Let $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ be two differentiable functions. Define $h : \mathbb{R}^n \to \mathbb{R}$ by $h(x) = f(x)^T g(x)$. Then h is also differentiable, and we have:

$$J_h(x) = f(x)^T J_g(x) + g(x)^T J_f(x),$$

$$\nabla h(x) = J_h(x)^T.$$

- The chain rule: Let $f : \mathbb{R} \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}$. We suppose that g is differentiable on an open set $D \subset \mathbb{R}^n$ and that $f : (a, b) \to D$ is differentiable on (a, b). Then the composite function $h : (a, b) \to \mathbb{R}$ given by $h(t) = g(f(t))$ is differentiable on (a, b), and

$$h'(t) = \nabla g(f(t))^T \begin{pmatrix} f_1'(t) \\ \vdots \\ f_n'(t) \end{pmatrix}.$$

- A special case of the chain rule that comes up a lot: $g : \mathbb{R}^n \to \mathbb{R}$, $h : \mathbb{R}^n \to \mathbb{R}$. Define

$$h(t) = g(\overline{x} + t\overline{y}),$$

where $\overline{x}, \overline{y} \in \mathbb{R}^n$ are fixed vectors and $\overline{x} + t\overline{y} \in \mathbb{R}^n$.

$$h'(t) = \overline{y}^T \nabla g(\overline{x} + t\overline{y}),$$

where$\overline{y}^T$ is $1 \times n$, and $\nabla g(\overline{x} + t\overline{y})$ is $n \times 1$.

$$h''(t) = \overline{y}^T (\nabla^2 g(\overline{x} + t\overline{y}))\overline{y},$$

where $\overline{y}^T$ is $1 \times n$, $\nabla^2 g(\overline{x} + t\overline{y})$ is $n \times n$, and $\overline{y}$ is $n \times 1$.

2.2.2 Taylor expansion

Theorem 2.41. *Suppose that you have $f : \mathbb{R} \to \mathbb{R}$ that $\in C^m$ (m times continuously differentiable). Then*

$$f(x) = f(x_0) + \frac{1}{1!}(x - x_0)f'(x_0) + \frac{1}{2!}(x - x_0)^2 f''(x_0) + \ldots + \frac{1}{m!}(x - x_0)^m f^{(m)}(x_0) + o(|x - x_0|^m).$$

The $o(\cdot)$ term is the error term:

$$\lim_{x \to x_0} \frac{o(|x - x_0|^m)}{|x - x_0|^m} = 0.$$

The error goes to zero faster than x^m does as $x \to x_0$.

Example 2.42. For $f(x) = e^x$,

$$f(x) \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

$1 + x$ is the first order Taylor expansion, $1 + x + \frac{x^2}{2}$ is the second order Taylor expansion, and so on.

We now generalize the Taylor expansion to multivariable functions.

Definition 2.43. (First order Taylor expansion in dimension n).

$$f(x) \approx f(x_0) + \frac{1}{1!}\nabla^T f(x_0)(x - x_0),$$

or

$$f(x) = f(x_0) + \frac{1}{1!}\nabla^T f(x_0)(x - x_0) + o(\|x - x_0\|^1).$$

Definition 2.44. (Second order Taylor expansion in dimension n).

$$f(x) = f(x_0) + \nabla^T f(x_0)(x - x_0) + \frac{1}{2!}(x - x_0)^T \nabla^2 f(x_0)(x - x_0) + o(\|x - x_0\|^2).$$

3 Unconstrained optimization

3.1 Notation and terminology of optimization problems

Let us describe the standard terminology used in an optimization problem. In abstract form, the optimization problem is to

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \Omega. \end{aligned}$$

x is our set of **decision variables**, $f(x) : \mathbb{R}^n \to \mathbb{R}$ is the **objective function**, and we optimize subject to $\Omega \subseteq \mathbb{R}^n$, the **constraint set or feasible set**. For instance, Ω might contain $f_1(x) \leq 1, f_2(x) \geq 5$.

An **optimal solution (or solution or global solution or global optimal solution)** is a point $x^* \in \Omega$ that satisfies $f(x^*) \leq f(x)\ \forall\ x \in \Omega$. ($x^*$ must also be feasible.) We write the solution as

$$x^* = \arg\min_{x \in \Omega} f(x),$$

and the **optimal value** is $f^* := f(x^*)$ if x^* exists. Such a solution may not exist or may not be unique: we might have that "the problem is unbounded below," "the optimal value is negative infinity," or "the optimal solution doesn't exist." Note that even if x^* does not exist, it can be well-defined. Think about e^{-x}, which doesn't have an optimal solution but has an optimal value $f^* = 0$; in this case, we can replace the term "minimum" with "infimum."

Theorem 3.1. *(Weierstrass theorem). If f is continuous and Ω is compact (closed and bounded), then an optimal solution to the minimization problem above is guaranteed to exist.*

Remark 3.2. Note that $\max f(x)$ s.t. $x \in \Omega = -\min -f(x)$ s.t. $x \in \Omega$. The optimal solution doesn't change, and the optimal value only changes its sign.

Unconstrained optimization is just when we take $\Omega = \mathbb{R}^n$ (decision variables are not constrained at all). Unconstrained optimization is by no means trivial. (There are many NP-hard unconstrained optimization problems.)

We now discuss unconstrained local and global minima. Define

$$B_\epsilon(\overline{x}) = \{x \in \mathbb{R}^n \mid \|x - \overline{x}\| \leq \epsilon\},$$

the ball centered at $\overline{x}$ with radius ϵ.

Definition 3.3. Consider our generic optimization problem. We say that a point $\overline{x}$ is a

- **local minimum** if $\exists\ \delta > 0$ s.t. $f(\overline{x}) \leq f(x)\ \forall\ x \in B_\delta(\overline{x})$.
- **strict local minimum** if $\exists\ \delta > 0$ s.t. $f(\overline{x}) < f(x)\ \forall\ x \in B_\delta(\overline{x}),\ x \neq \overline{x}$ (i.e., if you compare me to my neighbors, I am strictly better than them, with the exception of myself).
- **global minimum** if $f(\overline{x}) \leq f(x)\ \forall\ x \in \mathbb{R}^n$.
- **strict global minimum** if $f(\overline{x}) < f(x)\ \forall\ x \in \mathbb{R}^n,\ x \neq \overline{x}$.

3.2 Examples of unconstrained optimization

We discuss two examples of unconstrained optimization.

3.2.1 Fermat-Weber problem

The first example involves the Fermat-Weber problem (facility location planning). One example question we ask is: where should you live in the U.S. to be close to your loved ones? For instance, your loved ones could be your mom, dad, your lover, your grandma, your second lover (!), and your best friend. Each person i has location $x_i \in \mathbb{R}^2$, for $i = 1, \ldots, n$, so $x_i = \begin{pmatrix} x_{i,1} \\ x_{i,2} \end{pmatrix}$. Your location is $y \in \mathbb{R}^2$ and is the decision variable. The optimization problem is

$$\min_{y \in \mathbb{R}^2} \sum_{i=1}^{m} \|y - x_i\|.$$

(Recall that for $u \in \mathbb{R}^n$, $\|u\| = \sqrt{u^T u} = \sqrt{u_1^2 + \ldots + u_n^2}$.)

The weighted version of the problem, when you care more about some loved ones than others, is

$$\min_{y \in \mathbb{R}^n} \sum_{i=1}^{m} w_i \|y - x_i\|, \ w_i \geq 0 \in \mathbb{R}.$$

As we will see later, this problem is easy to solve because it has a nice structure called convexity. However, just tweaking the problem a little bit (say you want to be far from some subset of your friends and family) can turn this problem into a "hard" problem.

3.2.2 Least squares problem

Our second example is the least squares problem. We have data $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$. We aim to solve

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2,$$

or

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{n} (a_i^T x - b_i)^2.$$

Note that

$$A = \begin{pmatrix} a_1^T \\ \vdots \\ a_m^T \end{pmatrix}, \ x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \ b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix},$$

where $a_1^T, \ldots, a_m^T$ are row vectors. Usually, $m >> n$, so the linear equations are overdetermined. The least squares solution finds a "good" x for the m equations. We outline some of the many applications of least squares.

Example 3.4. (Application 1: Data fitting). We have data (t_i, y_i), $i = 1, \ldots, m$. We aim to find the best cubic fit $P(t) = c_3 t^3 + c_2 t^2 + c_1 t + c_0$. Our objective function is

$$\min_{c_0, c_1, c_2, c_3} \sum_{i=1}^{m} (P(t_i) - y_i)^2.$$

We can also write the objective function as

$$\min_{c_0,c_1,c_2,c_3} \left\| \begin{pmatrix} 1 & t_1 & t_1^2 & t_1^3 \\ 1 & t_2 & t_2^2 & t_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t_m & t_m^2 & t_m^3 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} - \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \right\|^2 .$$

Question 3.5. How do we solve an overdetermined system of equations?

Example 3.6. Let's examine the example of a simple linear predictor for the stock price $s(t)$ of some company on day t. Our model is

$$s(t) = a_1 s(t-1) + a_2 s(t-2) + a_3 s(t-3) + a_4 s(t-4).$$

Our decision variables are a_1, a_2, a_3, a_4. Say we have 90 days worth of data.
We aim to solve

$$\min_{a_i} \|Ax - b\| = \min_{a_i} \left\| \begin{pmatrix} s(4) & s(3) & s(2) & s(1) \\ s(5) & s(4) & s(3) & s(2) \\ \vdots & \vdots & \vdots & \vdots \\ s(89) & s(88) & s(87) & s(86) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} - \begin{pmatrix} s(5) \\ s(6) \\ \vdots \\ s(90) \end{pmatrix} \right\| .$$

3.3 Unconstrained optimality conditions

Optimality conditions are results that give us some structural information about the properties of optimal solutions.

We will use $a \implies b$ to denote that a is **necessary** for b and $a \impliedby b$ to denote that b is **sufficient** for a.

Theorem 3.7. *(First order necessary condition (FONC) for (local) optimality). If $\overline{x}$ is an unconstrained local minimum of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, then we must have that $\nabla f(\overline{x}) = 0$.*

Proof. Suppose $\overline{x}$ is a local min, but $\nabla f(\overline{x}) \neq 0$. Say that, WLOG, $\frac{\partial f}{\partial x_1}(\overline{x}) > 0$. Let $g(\alpha) = f(\overline{x} - \alpha e_i)$, where $g : \mathbb{R} \to \mathbb{R}$. ($g$ looks at how f changes as we move away from $\overline{x}$. We add the negative sign so that we move in the opposite direction of the gradient; if the gradient is non-zero, we try to find a "better" point.) So $g'(\alpha) = -e_i^T \nabla f(\overline{x} - \alpha e_i)$. Note that

$$g'(0) = -e_i^T \nabla f(\overline{x}) < 0$$

and, by definition,

$$g'(0) = \lim_{\alpha \to 0} \frac{g(\alpha) - g(0)}{\alpha} < 0.$$

If we had local optimality of $\overline{x}$, we would have $g(0) = f(\overline{x}) \leq f(\overline{x} - \alpha e_i) = g(\alpha)$. However, this is contradicted by the previous line, thus contradicting local optimality of $\overline{x}$. □

Remark 3.8. Does $\nabla f(\overline{x}) = 0$ imply that $\overline{x}$ is a local min? This condition is necessary but not sufficient for local optimality—$\overline{x}$ could be a local max or a saddle point. However, we'll see later that in the presence of convexity, this condition is in fact sufficient for local (and global!) optimality.

Definition 3.9. (Critical point or stationary point). A point $\overline{x}$ satisfying $\nabla f(\overline{x}) = 0$ is called a critical point or a stationary point.

Theorem 3.10. *(Second order necessary condition (SONC) for (local) optimality). If $\overline{x}$ is an unconstrained local minimum for a twice continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, then, in addition to $\nabla f(\overline{x}) = 0$, we must have that $\nabla^2 f(\overline{x}) \succeq 0$ (i.e., the Hessian at $\overline{x}$ is positive semidefinite).*

Proof. Pick any vector $y \in \mathbb{R}^n$. We need to show that $y^T \nabla^2 f(\overline{x}) y \geq 0$. With $\alpha \in \mathbb{R}$, the second order Taylor expansion of f around $\overline{x}$ is

$$f(\overline{x} + \alpha y) = f(\overline{x}) + \alpha y^T \nabla f(\overline{x}) + \frac{\alpha^2}{2} y^T \nabla^2 f(\overline{x}) y + o(\alpha^2 \|y\|^2).$$

We know that $\nabla f(\overline{x}) = 0$ by the previous theorem. So

$$\frac{f(\overline{x} + \alpha y) - f(\overline{x})}{\alpha^2} = \frac{1}{2} y^T \nabla^2 f(\overline{x}) y + \frac{o(\alpha^2 \|y\|^2)}{\alpha^2}.$$

$$0 \leq y^T \nabla^2 f(\overline{x}) y + 0$$

by local optimality of $\overline{x}$ and since $\lim_{\alpha \to 0} \frac{o(\alpha^2 \|y\|^2}{\alpha^2} = 0$, for any y. (At some point, we will fall into the ball of local optimality.) □

Lemma 3.11. *$eig(B + \gamma I) = eig(B) + \gamma$.*

Proof.

$$\begin{aligned} &\det(B + \gamma I - \lambda I) = 0 \\ \implies &\det(B + (\gamma - \lambda) I) = 0 \\ \implies &\det(B - (-\gamma + \lambda) I) = 0 \end{aligned}$$

If λ is an eigenvalue of $B + \gamma I$, then $-\gamma + \lambda$ is an eigenvalue of B. Thus,

$$\lambda_B = -\gamma + \lambda \implies \lambda = \lambda_B + \gamma.$$

□

Theorem 3.12. *(Second order sufficient condition (SOSC) for (local) optimality). Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. If for a point $\overline{x} \in \mathbb{R}^n$ we have $\nabla f(\overline{x}) = 0$ and $\nabla^2 f(\overline{x}) \succ 0$ (i.e., the Hessian at $\overline{x}$ is positive definite), then $\overline{x}$ is a strict local min of f.*

Proof. We know that $\nabla^2 f(\overline{x}) \succ 0$ implies that all eigenvalues of $\nabla^2 f(\overline{x}) > 0$. Let λ be the minimum eigenvalue of $\nabla^2 f(\overline{x})$. We know that $\lambda > 0$. This implies that $\nabla^2 f(\overline{x}) - \lambda I \succeq 0$, using the previous lemma. We thus have

$$\begin{aligned} &\nabla^2 f(\overline{x}) - \lambda I \succeq 0 \\ \implies &y^T (\nabla^2 f(\overline{x}) - \lambda I) y \geq 0 \\ \implies &y^T \nabla^2 f(\overline{x}) y \geq \lambda \|y\|^2, \end{aligned}$$

using $y^T y = \|y\|^2$. Taylor expansion gives us

$$f(\overline{x}+y) = f(\overline{x}) + \nabla^T f(\overline{x})y + \frac{1}{2}y^T\nabla^2 f(\overline{x})y + o(\|y\|^2).$$

The term $\nabla^T f(\overline{x})y = 0$, so we can write

$$\begin{aligned} f(\overline{x}+y) - f(\overline{x}) &= \frac{1}{2}y^T\nabla^2 f(\overline{x})y + o(\|y\|^2) \\ \implies f(\overline{x}+y) - f(\overline{x}) &\geq \frac{1}{2}\lambda\|y\|^2 + o(\|y\|^2) \\ \implies f(\overline{x}+y) - f(\overline{x}) &\geq \|y\|^2\left(\frac{\lambda}{2} + \frac{o(\|y\|^2)}{\|y\|^2}\right). \end{aligned}$$

Taylor's theorem tells us that

$$\lim_{\|y\|\to 0}\frac{o(\|y\|^2)}{\|y\|^2} = 0 \implies \exists\ \delta > 0 \text{ s.t. } \|y\| \leq \delta \implies \left|\frac{o(\|y\|^2)}{\|y\|^2}\right| < \frac{\lambda}{2},$$

where the third equation follows from the definition of the limit. Thus,

$$\exists\ \delta > 0 \text{ s.t. } \|y\| < \delta \implies f(\overline{x}+y) - f(\overline{x}) > 0 \implies f(\overline{x}) < f(\overline{x}+y) \implies \overline{x} \text{ is a strict local min.}$$

□

Remark 3.13. $\nabla f(\overline{x}) = 0,\ \nabla^2 f(\overline{x}) \succeq 0$ is not sufficient for local optimality. Think of $f(x) = x^3, f'(0) = f''(0) = 0$.

Remark 3.14. $\nabla^2 f(\overline{x}) \succ 0$ is not necessary for (even strict global) optimality. Consider $f(x) = x^4, f''(0) = 0$, but $x = 0$ is a strict min.

Example 3.15. We are given some function

$$f(x) = \frac{1}{2}x_1^2 + x_1x_2 + 2x_2^2 - 4x_1 - 4x_2 - x_2^3.$$

Find all local minima and maxima. Is any of them global?

$$\nabla f(x) = \begin{pmatrix} x_1 + x_2 - 4 \\ x_1 + 4x_2 - 4 - 3x_2^2 \end{pmatrix} = 0 \implies \overline{x} = \begin{pmatrix} 4 \\ 0 \end{pmatrix},\ \tilde{x} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}.$$

$\overline{x}$ and $\tilde{x}$ are candidates for local/global min/max.

$$\nabla^2 f(x) = \begin{pmatrix} 1 & 1 \\ 1 & 4 - 6x_2 \end{pmatrix} \implies \nabla^2 f(\tilde{x}) = \begin{pmatrix} 1 & 1 \\ 1 & -2 \end{pmatrix} \not\succeq 0, \not\preceq 0.$$

So by SONC, $\tilde{x}$ is not a local min nor a local max.

$$\nabla^2 f(\overline{x}) = \begin{pmatrix} 1 & 1 \\ 1 & 4 \end{pmatrix} \succ 0,$$

so by SOSC, $\overline{x}$ is a strict local min.

Is $\overline{x}$ a global min? No—we can see $f(0,\gamma) \to -\infty$ as $\gamma \to \infty$. Thus, a global min does not exist.

Remark 3.16. Whenever we do not discuss convexity, every application of FONC, SONC, and SOSC refer to local optimization—we cannot draw any claims about global optimality here.

3.4 Solution to least squares

Remark 3.17. Let's revisit the least squares problem

$$\min_{x \in \mathbb{R}^n} \|Ax - b\|^2,$$

where we are given input $A \in \mathbb{R}^{m \times n}$ (the columns of A are linearly independent) and $b \in \mathbb{R}^m$. This is an unconstrained optimization problem, and the objective function is differentiable as many times as we want.

Let $f(x) = \|Ax - b\|^2 = (Ax - b)^T(Ax - b) = x^T A^T Ax - 2x^T A^T b + b^T b$.

$$\nabla f(x) = 2A^T Ax - 2A^T b,$$

where $\nabla f(x) \in \mathbb{R}^n$. We set $\nabla f(x) = 0 \implies A^T Ax = A^T b \implies x = (A^T A)^{-1} A^T b$, the famous **least squares solution**. $A^T Ax = A^T b$ are called **normal equations**.

$(A^T A)^{-1}$ exists because the only point in the nullspace of $A^T A$ is zero. Why is this true?

$$A^T Ax = 0 \implies x^T A^T Ax = 0 \implies (Ax)^T (Ax) = 0 \implies \|Ax\|^2 = 0 \implies Ax = 0,$$

using the positivity of norms property. Since the columns of A are linearly independent, and $Ax = \begin{pmatrix} c_1 & c_2 & \dots & c_n \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = x_1 \begin{pmatrix} c_1 \end{pmatrix} + x_2 \begin{pmatrix} c_2 \end{pmatrix} + \dots x_n \begin{pmatrix} c_n \end{pmatrix} \implies (x_1, \dots, x_n)^T = 0$, or $x = 0$.

$\nabla^2 f(x) = 2A^T A$. We want to prove that this matrix is positive definite. Note that

$$y^T A^T Ay = (Ay)^T (Ay) = \|Ay\|^2 \geq 0 \ \forall \ y \implies A^T A \succeq 0.$$

We cannot say anything about this; while this passes the necessary condition, we still need the sufficiency condition to make a claim. (Counterexample: $y = x^3$, $f'(0) = 0$, $f''(0) = 0$.)

Is $A^T A \succ 0$? We already showed that $y \neq 0 \implies Ay \neq 0$, since the columns of A are linearly independent. This implies that $\|Ay\|^2 > 0 \ \forall \ y \neq 0 \implies A^T A \succ 0$. Thus,

$$x = (A^T A)^{-1} A^T b$$

is a strict local min (and a strict global min by convexity, as we will see later).

4 Convex optimization

Convex optimization problems are just a special case of *constrained* optimization problems

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \Omega, \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a "convex function" and $\Omega \subseteq \mathbb{R}^n$ is a "convex set."

Why do we want to study convex optimization? Arguably, these optimization problems are the broadest class of optimization problems that we can solve efficiently. They also have nice geometric properties (think convex analysis), as all local minima are automatically global minima. Furthermore, lots of problems in applications happen to be convex (such as many applications in OR, SML, Controls, etc.). There is also nice high-level software that allows modelling and solving of convex optimization problems (such as MATLAB CVX).

4.1 Convex sets

Definition 4.1. (Convex set). A set $\Omega \subseteq \mathbb{R}^n$ is convex if

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

$\lambda x + (1-\lambda)y$ is a **convex combination** of x and y. When $\lambda = 0$, we are at y, and when $\lambda = 1$, we are at x. Additionally, if $\lambda = \frac{1}{2}$, for instance, we are simply averaging x and y and generating a point on the line segment connecting $x, y \in \Omega$.

Example 4.2. An example of a nonconvex set is a heart. For instance, if we have an x in the upper left lobe of the heart and a y in the upper right lobe of the heart, the line segment connecting the two leaves the set.

Definition 4.3. (Midpoint convex). A set $\Omega \subseteq \mathbb{R}^n$ is midpoint convex if $\forall\ x, y \in \Omega \implies \frac{x+y}{2} \in \Omega$ (the set is closed). A convex set $\implies$ midpoint convex (we can take $\lambda = \frac{1}{2}$). The converse is not always true: take $S = \{\text{all rationals in } [0,1]\}$. S is midpoint convex but not convex. However, a midpoint convex set is convex if S is closed.

We discuss some common convex sets in optimization. We leave it as an exercise to prove convexity in each case.

- **Hyperplanes**: $\{x \in \mathbb{R}^n \mid a^T x = b\}$ for some $a \in \mathbb{R}^n,\ a \neq 0,\ b \in \mathbb{R}$. Note that hyperplanes are level sets of affine functions.
- **Half-spaces**: $\{x \in \mathbb{R}^n \mid a^T x \leq b\}$ for some $a \in \mathbb{R}^n,\ a \neq 0,\ b \in \mathbb{R}$. To prove that a half-space is convex, we take two points $x, y \in S = \{x \in \mathbb{R}^n \mid a^T x \leq b\}$ and $\lambda \in [0,1]$ and show that $\lambda x + (1-\lambda)y \in S$:

  $$a^T(\lambda x + (1-\lambda)y) = \lambda a^T x + (1-\lambda)a^T y \leq \lambda b + (1-\lambda)b = b \implies \lambda x + (1-\lambda)y \in S.$$

  Note that half-spaces are sublevel sets of affine functions.

- **Euclidean balls**: $\{x \in \mathbb{R}^n \mid \|x - x_c\| \leq r\}$ for some $x_c \in \mathbb{R}^n$ and scalar $r \geq 0$. We need to show that $x, y \in S = \{x \in \mathbb{R}^n \mid \|x - x_c\| \leq r\},\ \lambda \in [0,1] \implies \lambda x + (1-\lambda)y \in S$. We can write

  $$\|\lambda x + (1-\lambda)y - x_c\| = \|\lambda(x - x_c) + (1-\lambda)(y - x_c)\| \leq \|\lambda(x - x_c)\| + \|(1-\lambda)(y - x_c)\|,$$

  using the triangle inequality. Using the homogeneity property of norms, we can express

  $$\|\lambda(x - x_c)\| + \|(1-\lambda)(y - x_c)\| = \lambda\|x - x_c\| + (1-\lambda)\|y - x_c\| \leq \lambda r + (1-\lambda)r = r.$$

- **Ellipsoids**: $\{x \in \mathbb{R}^n \mid (x - x_c)^T P(x - x_c) \leq r\}$ for some symmetric $n \times n$ $P \succ 0,\ x_c \in \mathbb{R}^n,\ r \geq 0$. (When $P = I$, we are dealing with Euclidean balls.) Here's an example of an ellipsoid:

  $$S = \{x \in \mathbb{R}^2 \mid 5x_1^2 + x_2^2 \leq 1\},$$

  where $r = 1,\ x_c = \begin{pmatrix} 0 \\ 0 \end{pmatrix},\ P = \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix}$. Note that ellipsoids are sublevel sets of convex quadratic functions. We can prove that ellipsoids are convex sets more easily once we have seen convex functions and quasiconvex functions, so stay tuned.

- **Polyhedra**: $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ for some $A \in \mathbb{R}^{m \times n},\ b \in \mathbb{R}^m$. A polyhedron is the solution set of finitely many linear inequalities. We can interpret polyhedra as intersections of half-spaces $a_i^T x \leq b_i$ for $i = 1, \ldots, m$, where a_i^T is the i-th row of A. Polyhedra are convex as they are the intersection of half-spaces. These are the feasible sets of linear programs (LPs), as we will see later.

Definition 4.4. ("Fancier" convex sets).

- The set of all $n \times n$ (symmetric) positive semidefinite matrices, $S_+^{n \times n} = \{P \in S^{n \times n} \mid P \succeq 0\}$, is convex. In other words, $A \succeq 0,\ B \succeq 0 \implies \lambda A + (1-\lambda)B \succeq 0\ \forall\ \lambda \in [0,1]$.

  Here's the proof: we take any $x \in \mathbb{R}^n, A \succeq 0, B \succeq 0, \lambda \in [0,1]$. $x^T(\lambda A + (1-\lambda)B)x = \lambda x^T A x + (1-\lambda)x^T B x \geq 0$ since $A \succeq 0 \implies x^T A x \geq 0$ and $B \succeq 0 \implies x^T B x \geq 0$.

- The set of nonnegative polynomials in n variables and of degree d is convex. A polynomial $p(x_1, \ldots, x_n)$ is nonnegative if $p(x) \geq 0\ \forall\ x \in \mathbb{R}^n$; an example is

  $$p(x_1, \ldots, x_n) = 5x_1^4 + 2x_1x_2 - 3x_2^2.$$

  An example of such a set could be

  $$\{(c_1, c_2) \mid 2x_1^4 + x_2^4 + c_1x_1x_2^3 + c_2x_1^3x_2 \geq 0\ \forall\ (x_1, x_2) \in \mathbb{R}^2\}.$$

Remark 4.5. The intersection of two convex sets is convex: if S_1 is convex and S_2 is convex, then $S_1 \cap S_2$ is convex.

Proof. Take $x, y \in S_1 \cap S_2$. We need to show that for all $\lambda \in [0,1]$, $\lambda x + (1-\lambda)y \in S_1 \cap S_2$. Since $x, y \in S_1$, and S_1 is convex, $\lambda x + (1-\lambda)y \in S_1$. Similarly, since $x, y \in S_2$, and S_2 is convex, $\lambda x + (1-\lambda)y \in S_2$. □

Remark 4.6. The union of two convex sets may not be convex.

4.2 Convex functions

Definition 4.7. (Convex function). A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if the domain of f is a convex set and if, $\forall\ x, y \in \text{domain}(f)$ and $\forall\ \lambda \in [0, 1]$, we have

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y),$$

where $\lambda x + (1-\lambda)y$ is a convex combination of x and y. In other words, we can take any two points x, y: f evaluated at any convex combination must be no larger than the same convex combination of $f(x)$ and $f(y)$ if f is convex. Geometrically, the line segment connecting $(x, f(x))$ and $(y, f(y))$ sits above the graph of f.

A classic example of a convex function is $y = x^2$. An example of a function that is not convex is $y = -x^2$.

Example 4.8. Some more examples of convex functions in one dimension ($n = 1$) are:

- $e^{ax}\ \forall\ a \in \mathbb{R}$.
- $-\log x$.
- $x^a,\ a \geq 1,\ \text{dom}\ \mathbb{R}^+$.
- $|x|$.

Definition 4.9. A function $f : \mathbb{R}^n \to \mathbb{R}$ is

- **Concave** if $f(\lambda x + (1 - \lambda)y) \geq \lambda f(x) + (1 - \lambda)f(y)\ \forall\ \lambda \in [0, 1],\ \forall\ x, y$.
- **Strictly convex** if $f(\lambda x + (1 - \lambda)y) < \lambda f(x) + (1 - \lambda)f(y)\ \forall\ \lambda \in (0, 1),\ \forall\ x, y, x \neq y$.
- **Strictly concave** if $f(\lambda x + (1-\lambda)y) > \lambda f(x) + (1-\lambda)f(y),\ \forall\ \lambda \in (0, 1),\ \forall\ x, y, x \neq y$.

Observe that

$$f \text{ is convex } \iff -f \text{ is concave}$$

and

$$f \text{ is strictly convex } \iff -f \text{ is strictly concave.}$$

This is good because when we want to minimize, we minimize a convex function, and when we want to maximize, we maximize a concave function. Note that the optimal solutions stay the same for minimization and maximization problems, while the optimal value changes sign.

Remark 4.10. f strictly convex $\implies$ f is convex. Note that the converse is not true: a counterexample is a horizontal line.

We now cover some common convex functions in optimization:

- **Affine functions**: $f(x) = a^T x + b$ for any $a \in \mathbb{R}^n,\ b \in \mathbb{R}$. Affine functions are both convex (but not strictly convex) and concave (but not strictly concave). $\forall\ \lambda \in [0, 1]$,

$$\begin{aligned} f(\lambda x + (1 - \lambda)y) &= a^T(\lambda x + (1 - \lambda)y) + b \\ &= \lambda a^T x + (1 - \lambda)a^T y + b \\ &= \lambda a^T x + (1 - \lambda)a^T y + \lambda b + (1 - \lambda)b \\ &= \lambda(a^T x + b) + (1 - \lambda)(a^T y + b) \\ &= \lambda f(x) + (1 - \lambda)f(y). \end{aligned}$$

- **Some quadratic functions** are convex. Quadratic functions take the form $f(x) = x^TQx + b^Tx + c$, where $Q \in \mathbb{R}^{n\times n}$ is symmetric, $b \in \mathbb{R}^n,\ c \in \mathbb{R}$.
  - Convex $\iff Q \succeq 0$.
  - Strictly convex $\iff Q \succ 0$.
  - Concave $\iff Q \preceq 0$.
  - Strictly concave $\iff Q \prec 0$.

  This is easy to prove from the second order characterization of convexity (coming up!).

- **Any norm** is convex. Recall that any function f is a **norm** if it satisfies:
  - $f(\alpha x) = |\alpha| f(x)\ \forall\ \alpha \in \mathbb{R}$ (homogeneity property)
  - $f(x + y) \leq f(x) + f(y)$ (triangle inequality)
  - $f(x) \geq 0\ \forall\ x, f(x) = 0 \iff x = 0$ (positivity property).

  Here's the proof: for $\lambda \in [0, 1]$, $f(\lambda x + (1-\lambda)y) \leq f(\lambda x) + f((1-\lambda)y) = \lambda f(x) + (1-\lambda)f(y)$, using the triangle inequality and the homogeneity property.

Definition 4.11. (Midpoint convex). A function $f : \mathbb{R}^n \to \mathbb{R}$ is midpoint convex if its domain is a convex set and, for all x, y in its domain, we have

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

All convex functions are midpoint convex. Continuous midpoint convex functions are convex.

Theorem 4.12. *(Convexity along all lines). A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if $g : \mathbb{R} \to \mathbb{R}$ given by $g(t) = f(x + ty)$ is convex (as a univariate function) $\forall\ x \in dom(f),\ y \in \mathbb{R}^n$. ($dom(g)$ is all t for which $x + ty \in dom(f)$.)*

Proof. Suppose that for some x, y, $g(\alpha) = f(x + \alpha y)$ were not convex. Then there exists $\lambda \in [0, 1]$, α_1, α_2 such that

$$g(\lambda\alpha_1 + (1-\lambda)\alpha_2) > \lambda g(\alpha_1) + (1-\lambda)g(\alpha_2).$$

Then, we must have

$$f(x+(\lambda\alpha_1+(1-\lambda)\alpha_2)y) = f(\lambda(x+\alpha_1 y)+(1-\lambda)(x+\alpha_2 y)) > \lambda f(x+\alpha_1 y)+(1-\lambda)f(x+\alpha_2 y).$$

□

4.3 Connections between convex sets and convex functions

We will see a couple connections between convex sets and convex functions: via epigraphs and via sublevel sets.

4.3.1 Epigraphs

Definition 4.13. (Epigraph). The epigraph (epi(f)) of a function $f : \mathbb{R}^n \to \mathbb{R}$ is a subset of $\mathbb{R}^{n+1}$ defined as

$$\{(x,t) \in \mathbb{R}^{n+1} \mid f(x) \leq t, x \in \mathbb{R}^n, t \in \mathbb{R}\}.$$

Theorem 4.14. *f is convex (as a function) $\iff$ epi(f) is convex (as a set).*

Proof. We first prove that epi(f) is a convex set $\implies$ f is a convex function. Suppose that f were not convex. Then there must exist $x, y \in \mathbb{R}^n,\ \lambda \in [0,1]$ such that $f(\lambda x + (1-\lambda)y) > \lambda f(x) + (1-\lambda)f(y)$. We pick $(x, f(x)), (y, f(y)) \in \text{epi}(f)$. The previous statement implies that $(\lambda x + (1-\lambda)y, \lambda f(x) + (1-\lambda)f(y)) \notin \text{epi}(f)$. Thus, epi($f$) is not convex.

Now, we prove that f is convex $\implies$ epi(f) is a convex set. Suppose that epi(f) were not convex. Then, $\exists\ (x, t_x) \in \text{epi}(f), (y, t_y) \in \text{epi}(f), \lambda \in [0,1]$ such that

$$(\lambda x + (1-\lambda)y, \lambda t_x + (1-\lambda)t_y) \notin \text{epi}(f).$$

Thus, $f(\lambda x + (1-\lambda)y) > \lambda t_x + (1-\lambda)t_y \geq \lambda f(x) + (1-\lambda)f(y)$. So f is not convex. □

4.3.2 Quasiconvex functions

Recall that the α-sublevel set of a function $f : \mathbb{R}^n \to \mathbb{R}$ is the set

$$S_\alpha = \{x \in \mathbb{R}^n \mid f(x) \leq \alpha\}.$$

Definition 4.15. (Quasiconvex). A function f is quasiconvex if all of its sublevel sets ($S_\alpha\ \forall\ \alpha$) are convex sets.

Theorem 4.16. *f is convex $\implies$ f is quasiconvex. Note that the converse is not true (take $f(x) = x^3$, which is quasiconvex but not convex).*

Proof. Pick an arbitrary sublevel set $S_\alpha = \{x \mid f(x) \leq \alpha\}$. We need to show that if we take $x \in S_\alpha, y \in S_\alpha, \lambda \in [0,1]$, then $\lambda x + (1-\lambda)y \in S_\alpha$. Note that since $x \in S_\alpha$, $f(x) \leq \alpha$, and since $y \in S_\alpha$, we have $f(y) \leq \alpha$. Since f is convex,

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \leq \lambda\alpha + (1-\lambda)\alpha = \alpha.$$

Quasiconvex $\implies$ Convex does not hold. Take a function $f(x) = x^T Q x$. If f is not convex, then Q is not PSD, so there exists $\overline{x} \in \mathbb{R}^n$ such that $\overline{x}^T Q \overline{x} = -\beta$ for some $\beta > 0$. Consider $S_{-\beta} = \{x \in \mathbb{R}^n \mid f(x) \leq -\beta\}$. 0 is not in $S_{-\beta}$ since $f(0) = 0 \implies 0 \notin S_{-\beta}$. *** □

Example 4.17. Take x^2, which is convex and, thus, quasiconvex. We can examine any sublevel set $S_\alpha = \{x \in \mathbb{R}^n \mid f(x) \leq \alpha\}$. The corresponding interval on x is bounded and convex. What happens with x^3 though? All sublevel sets are convex, but x^3 is not convex.

Remark 4.18. One step further: Strictly convex $\implies$ Convex $\implies$ Quasiconvex. The two converses do not hold. Counterexamples are $f(x) = x$ and $f(x) = x^3$.

4.4 Convex optimization problems

Why do we pay special attention to convex optimization problems? In a convex problem, every local minimum is automatically a global minimum. (This is true even for the abstract convex optimization problem that is defined below.) In the unconstrained case, every stationary point (i.e., a point at which the gradient is zero) is automatically a global minimum.

Definition 4.19. (Convex optimization problem). Formally, a convex optimization problem is a problem of the form

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \ i = 1, \ldots, m \\ \text{s.t.} \quad & h_j(x) = 0, \ j = 1, \ldots, k, \end{aligned}$$

where $f, g_1, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}$ are convex functions and $h_1, \ldots, h_k : \mathbb{R}^n \to \mathbb{R}$ are affine functions.

Definition 4.20. (Abstract convex optimization problem).

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \Omega, \end{aligned}$$

where f is a convex function and Ω is a convex set.

A function that satisfies the first form satisfies the second form, but not vice versa. The first form is much closer to what is used in algorithms, while the second form is much closer to mathematical analysis. We define the first form because it is much closer to what we can actually solve efficiently; otherwise, many abstract and complex optimization problems can be formulated as optimization problems over a convex set.

Why does the first form imply the second form? Define

$$\Omega = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0, h_i(x) = 0\}.$$

If g_i are convex and h_i are affine, then Ω is a convex set. Why? Sublevel sets of g_i are convex, and intersections of convex sets are convex. Why do we include the set of affine functions? The only functions with equality guaranteed to give you convex sets are affine functions. (Think about a quadratic function: $x^2 = 2$ gives two points, which is not a convex set. But finding where some line equals a certain value gives either one point or the whole line.) So $\{x \in \mathbb{R}^n \mid a^T x + b = 0\}$ is a hyperplane, which is convex. Thus, *the entire feasible set* $\Omega = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0, h_i(x) = 0\}$ *is a convex set.*

The converse is not true: consider, for instance, $\Omega = \{x \in \mathbb{R} \mid x^3 \leq 0\}$. Then Ω is a convex set, but minimizing a convex function over Ω is not a convex optimization problem according to the first definition.

Note that we can reason about the convexity of sets based on the level sets of functions. Since the sublevel sets of a convex function must all be convex, if a function in question has a sublevel set that is not convex, then the function must not be convex.

Furthermore, keep in mind that MATLAB CVX only accepts convex optimization problems of the first form: `min(convex) s.t. convex <= 0, affine == 0`.

4.5 Local and global minima in the constrained setting

Consider a problem $\min f(x)$ s.t. $x \in \Omega$, where f is a convex function and Ω is a convex set. $\overline{x}$ is said to be

- **feasible** if $\overline{x} \in \Omega$,
- a **local min** if feasible and if $\exists\ \delta > 0$ s.t. $f(\overline{x}) \leq f(x)\ \forall\ x \in B_\delta(\overline{x}) \cap \Omega$,
- a **strict local min** if feasible and if $\exists\ \delta > 0$ s.t. $f(\overline{x}) < f(x)\ \forall\ x \in B_\delta(\overline{x}) \cap \Omega,\ x \neq \overline{x}$,
- a **global min** if feasible and if $f(\overline{x}) \leq f(x)\ \forall\ x \in \Omega$,
- a **strict global min** if feasible and if $f(\overline{x}) < f(x)\ \forall\ x \in \Omega,\ x \neq \overline{x}$.

Note that we have defined these terms before, but now we are adding to the definition the notion of Ω and intersections with Ω.

Theorem 4.21. *Consider a problem* $\min f(x)$ *s.t.* $x \in \Omega$*, where* f *is a convex function and* Ω *is a convex set. Then, every local min of the problem is also a global min.*

Proof. Suppose that $x \in \Omega$ is a local min but not a global min. Thus, $\exists\ y \in \Omega$ s.t. $f(y) < f(x)$. Because $x \in \Omega,\ y \in \Omega$, and Ω is convex, $\lambda x + (1-\lambda)y \in \Omega\ \forall\ \lambda \in [0,1]$. Thus, $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) < \lambda f(x) + (1-\lambda)f(x) = f(x)$. As $\lambda \to 1$, $\lambda x + (1-\lambda)y \to x$. Thus, there exist points arbitrarily close to x that are better than x, so x is not a local min (a contradiction). □

Theorem 4.22. *Consider a problem* $\min f(x)$ *s.t.* $x \in \Omega$*, where* f *is a strictly convex function and* Ω *is a convex set. Then, any local min of* f *is a unique global min, i.e., the optimal solution, if it exists, is unique.*

Proof. Suppose that if f is strictly convex, then the global min is not a unique global min, i.e. $\exists\ x, y$ that are both global min.

$$x \in \Omega,\ y \in \Omega,\ f(x) = f(y) \leq f(w)\ \forall\ w \in \Omega.$$

$$f\left(\frac{x+y}{2}\right) < \frac{1}{2}f(x) + \frac{1}{2}f(y) = \frac{1}{2}f(x) + \frac{1}{2}f(x) = f(x).$$

This is a contradiction. □

Question 4.23. So strict convexity implies a unique global min. But does convexity imply a unique global min? No: a simple counterexample is $f(x) = c$.

Question 4.24. If we have a unique global min, do we have to have strict convexity? No: a simple counterexample is $f(x) = |x|$.

4.6 First order and second order characterizations of convex functions

Theorem 4.25. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable. Then, the following are equivalent:*

1. *f is convex.*
2. *$f(y) \geq f(x) + \nabla^T f(x)(y - x)\ \forall\ x, y \in dom(f)$ (**first order characterization**).*
3. *$\nabla^2 f(x) \succeq 0\ \forall\ x \in dom(f)$ (**second order characterization**).*

Proof. We prove (1) $\to$ (2). If f is convex, then by definition, we have

$$f(\lambda y + (1-\lambda)x) \leq \lambda f(y) + (1-\lambda)f(x),\ \forall\ \lambda \in [0,1], x, y \in \text{dom}(f).$$

We can rewrite this:

$$\implies f(x + \lambda(y - x)) \leq f(x) + \lambda(f(y) - f(x))$$

$$\implies f(y) - f(x) \geq \frac{f(x + \lambda(y-x)) - f(x)}{\lambda}.$$

As $\lambda \to 0$, we get that

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

Now, we prove (2) $\to$ (1). Suppose that (2) holds. Take any $x, y \in \text{dom}(f)$, and let $z = \lambda x + (1-\lambda)y$. We have the following:

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

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

$$\implies \lambda f(x) + (1-\lambda)f(y) \geq f(z) + \nabla^T f(z)(\lambda x + (1-\lambda)y - z)$$

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

The intuition behind (2) is that $f(x) + \nabla^T f(x)(y - x)$ is the first order Taylor expansion around x. The first order Taylor expansion is the best linear approximation of f around x; the function is always above the first order Taylor expansion for any y. If the function is non-convex, this statement immediately fails (the first order Taylor expansion would sometimes go above the function).

The intuition behind (3), in one variable, is that the Hessian is the second derivative of f and $f''(x) \geq 0$. Recall that $f''(x)$ gives us the curvature of f around x, so f essentially "curves up" everywhere. In higher dimensions, what does it mean to have $\nabla^2 f(x) \succeq 0$? We must have that $y^T \nabla^2 f(x) y \geq 0\ \forall\ x, y$. □

Corollary 4.26. *Consider an unconstrained optimization problem $\min f(x)$ s.t. $x \in \mathbb{R}^n$. Suppose that f is convex and differentiable. Then, any point $\overline{x}$ satisfying $\nabla f(\overline{x}) = 0$ is a global min.*

Proof. By the first order characterization of convexity, we know that

$$f(y) \geq f(x) + \nabla^T f(x)(y - x),$$

which is true for any x, y. We can fix x to be $\overline{x}$ to write

$$f(y) \geq f(\overline{x}) + \nabla^T f(\overline{x})(y - \overline{x}) = f(\overline{x}) \ \forall \ y.$$

So $\overline{x}$ is a global min.

Note that $\nabla f(x) = 0$ is always a necessary condition for local optimality in an unconstrained problem, but for convex problems, this is not only necessary but also sufficient for local and *global* optimality. Also recall that another necessary condition for (unconstrained) local optimality of a point x was $\nabla^2 f(x) \succeq 0$, which is automatically passed by a convex function. □

Now, let's discuss characterizations of strict convexity.

Theorem 4.27. *(First order characterization of strict convexity).*

$$f \text{ is strictly convex } \iff f(y) > f(x) + \nabla^T f(x)(y - x) \ \forall \ x, y \in \Omega, \ x \neq y.$$

Theorem 4.28. *(Second order sufficient condition for strict convexity).*

$$\nabla^2 f(x) \succ 0 \ \forall \ x \in \Omega \implies f \text{ strictly convex on } \Omega.$$

Note that the converse does not hold: take $f(x) = x^4$, for $x \in \mathbb{R}$. f is strictly convex, but $f''(0) = 0$.

4.7 Quadratic functions

4.7.1 Convexity of quadratic functions

Given a quadratic function $f(x) = x^T Q x + b^T x + c$, what can we say about its convexity? Note that $\nabla^2 f(x) = 2Q$. Thus, we have the following:

- f is convex $\iff Q \succeq 0$.
- f is strictly convex $\iff Q \succ 0$.

Note that these results are independent of b, c.

Proof. We leave the proof as an exercise. □

4.7.2 Unconstrained quadratic minimization

Let's revisit the least squares problem, where we minimize the function $f(x)$:

$$\begin{aligned} \min_x \quad & x^T Q x + b^T x + c \\ \text{s.t.} \quad & x \in \mathbb{R}^n. \end{aligned}$$

Case 1: The objective is not convex, i.e. $Q \not\succeq 0$. Let λ be a negative eigenvalue and y be its associated eigenvector. Note that $y^TQy = y^T\lambda y = \lambda\|y\|^2 < 0$ and that

$$f(y) = y^TQy + b^Ty + c,$$

where $y^TQy < 0$.

$$f(\alpha y) = \alpha^2 y^TQy + \alpha b^Ty + c,$$

where $y^TQy < 0$, $b^Ty > 0$, and $c > 0$. Let $\alpha \to \infty \implies f(\alpha y) \to -\infty$. So the optimal value is unbounded below.

Case 2: $Q \succ 0 \implies f$ is strictly convex. $\nabla f(x) = 2Qx + b$, so

$$\nabla f(x) = 0 \implies x = -\frac{1}{2}Q^{-1}b \implies \text{ a unique global solution.}$$

Q^{-1} exists because all eigenvalues of Q are positive.

Case 3: $Q \succeq 0$ (but $Q \not\succ 0$) so f is convex. We ask a few key questions:

- Can the problem be unbounded below? Take $Q = 0$.
- Can the problem have a unique solution? Yes, if $Q \succ 0$.
- Can the problem have multiple solutions? Take the function $(x_1 - x_2)^2$. The global solutions are anything that satisfy $x_1 = x_2$.

So the optimal value may or may not be bounded, and there could be many optimal solutions.

4.7.3 Least squares, revisited

We return to the least squares problem, minimizing the function $f(x)$:

$$\begin{aligned} \min \quad & \|Ax - b\|_2^2 \\ \text{s.t.} \quad & x \in \mathbb{R}^n. \end{aligned}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^n$. We assume that the columns of A are linearly independent.

From our previous analysis of the least squares solution, we know that $x = (A^TA)^{-1}A^Tb$ is a strict local minimum, aided by the fact that $A^TA \succ 0$. However, we now know that since f is strictly convex, the solution is a strict *global* minimum.

4.8 Optimality conditions for convex optimization problems

We have already seen the optimality conditions for the *unconstrained* case: those optimality conditions were FONC, SONC, SOSC. Those involved locally optimal solutions, but now we discuss *globally* optimal solutions.

Theorem 4.29. *Consider the problem* $\min f(x)$ *s.t.* $x \in \Omega$. *Suppose that* $f : \mathbb{R}^n \to \mathbb{R}$ *is a convex and differentiable function and that* Ω *is a convex set. Then, a point* $\overline{x} \in \Omega$ *is globally optimal if and only if* $\overline{x} \in \Omega$ *and* $\nabla^T f(\overline{x})(y - \overline{x}) \geq 0 \ \forall \ y \in \Omega$.

The necessity of the condition holds independent of the convexity of f*, but convexity is used in establishing sufficiency.*

Proof. First, we prove sufficiency, i.e. that if $\nabla^T f(\overline{x})(y - \overline{x}) \geq 0 \ \forall \ y \in \Omega$ holds, then $\overline{x}$ is a global min. This is true because if f is convex, then by the first order characterization of convexity,

$$f(y) \geq f(x) + \nabla^T f(x)(y - x) \ \forall \ x, y \implies f(y) - f(\overline{x}) \geq \nabla^T f(\overline{x})(y - \overline{x}) \geq 0$$

$$\implies f(y) \geq f(\overline{x}) \ \forall \ y \in \Omega.$$

Now, let's prove necessity, i.e. that if $\overline{x}$ is a global min, then $\nabla^T f(\overline{x})(y - x) \geq 0 \ \forall \ y \in \Omega$. We proceed using proof by contradiction. Suppose that there exists a globally optimal solution $\overline{x}$ and a $y \in \Omega$ such that $\nabla^T f(\overline{x})(y - x) < 0$. Let $g(\alpha) = f(\overline{x} + \alpha(y - \overline{x}))$; functions like this come up a lot, so we outline the intuition behind g: note that $g(0) = f(\overline{x})$ and $g(1) = f(y)$. Observe that $\overline{x} + \alpha(y - \overline{x}) \in \Omega \ \forall \ \alpha \in [0, 1]$ since Ω is convex. $g'(\alpha) = (y - \overline{x})^T \nabla f(\overline{x} + \alpha(y - \overline{x}))$. Thus, $g'(0) = \nabla^T f(\overline{x})(y - \overline{x}) < 0$ by assumption. By definition of the derivative, there exists a $\delta > 0$ such that if $0 < \alpha \leq \delta$, then $g(\alpha) < g(0)$. Thus, $f(\overline{x} + \alpha(y - \overline{x})) < f(\overline{x}) \ \forall \ \alpha \in (0, \delta)$. Therefore, $\overline{x}$ is not a global min—a contradiction. □

4.9 Convexity preserving operations

4.9.1 Rule 1: Nonnegative weighted sums

The sum of convex functions is also convex. $f_1(x), f_2(x)$ are convex $\implies g(x) = f_1(x) + f_2(x)$ is also convex.

Proof. Let f_1, f_2 be convex functions, $w_1, w_2 \geq 0$, $x, y \in \mathbb{R}^n$, $\lambda \in [0, 1]$.

$$\begin{aligned} g(\lambda x + (1 - \lambda)y) &= f_1(\lambda x + (1 - \lambda)y) + f_2(\lambda x + (1 - \lambda)y) \\ &\leq \lambda f_1(x) + (1 - \lambda) f_1(y) + \lambda f_2(x) + (1 - \lambda) f_2(y) \\ &= \lambda(f_1(x) + f_2(x)) + (1 - \lambda)(f_1(y) + f_2(y)) \\ &= \lambda g(x) + (1 - \lambda) g(y). \end{aligned}$$

□

More generally, if $f_1(x), f_2(x), \ldots, f_m(x)$ are convex, and $w_1, w_2, \ldots, w_m \geq 0$ are scalars, then $g(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots w_m f_m(x)$ is convex.

Proof. Let $f_1, \ldots, f_n$ be convex functions, $w_1, \ldots, w_n \geq 0$, $x, y \in \mathbb{R}^n$, $\lambda \in [0, 1]$.

$$\begin{aligned} g(\lambda x + (1 - \lambda)y) &= w_1 f_1(\lambda x + (1 - \lambda)y) + \ldots + w_n f_n(\lambda x + (1 - \lambda)y) \\ &\leq w_1 \cdot (\lambda f_1(x) + (1 - \lambda) f_1(y)) + \ldots + w_n \cdot (\lambda f_n(x) + (1 - \lambda) f_n(y)) \\ &= \lambda(w_1 f_1(x) + \ldots + w_n f_n(x)) + (1 - \lambda)(w_1 f_1(x) + \ldots + w_n f_n(x)) \\ &= \lambda g(x) + (1 - \lambda) g(y). \end{aligned}$$

□

Question 4.30. If f_1 and f_2 are convex functions...

- Is $f_1(x) - f_2(x)$ convex? Take $f_1 = 0, f_2 = x^2$.
- Is $f_1(x) \cdot f_2(x)$ convex? Take $f_1 = x, f_2 = x^2$.
- Is $\frac{f_1(x)}{f_2(x)}$ convex? Take $f_1 = x^{3/2}, f_2 = x$.

4.9.2 Rule 2: Composition with an affine mapping

If $f : \mathbb{R}^n \to \mathbb{R}$ is convex, then for $A \in \mathbb{R}^{n \times m},\ b \in \mathbb{R}^n$, we must have that $g(x) = f(Ax + b)$ is also convex. If f is concave, then g is also concave. Note that this relates to the least squares problem: $\min_x \|Ax - b\|^2$.

Proof. Let $x, y \in \mathbb{R}^n$ and $\lambda \in [0, 1]$. Then

$$\begin{aligned} g(\lambda x + (1-\lambda)y) &= f(A(\lambda x + (1-\lambda)y) + b) \\ &= f(\lambda Ax + (1-\lambda)Ay + \lambda b + (1-\lambda)b) \\ &= f(\lambda(Ax + b) + (1-\lambda)(Ay + b)) \\ &\leq \lambda f(Ax + b) + (1-\lambda) f(Ay + b) \\ &= \lambda g(x) + (1-\lambda) g(y). \end{aligned}$$

The proof in the concave case is similar. □

Example 4.31. Is $f(x_1, x_2) = (x_1 - 3x_2)^4 + 5e^{3x_1 - x_2 - 5}$ convex? Yes. $x_1 - 3x_2$ is affine and thus convex. $(x_1 - 3x_2)^4$ is convex (we know t^4 for some univariate t is convex). Similarly, $e^{3x_1 - x_2 - 5}$ is also convex by Rule 2. Thus, by Rule 1, the sum is convex.

4.9.3 Rule 3: Pointwise maximum

$f_1(x), \ldots, f_m(x)$, $f_i : \mathbb{R}^n \to \mathbb{R}$, are convex $\implies g(x) = \max\{f_1(x), \ldots, f_m(x)\}$, $g(x) : \mathbb{R}^n \to \mathbb{R}$, with $\mathrm{dom}(g) = \mathrm{dom}(f_1) \cap \ldots \cap \mathrm{dom}(f_m)$, is convex.

Proof. Let f_j, $j \in \{1, \ldots, m\}$, be the function that achieves the maximum. Then

$$\begin{aligned} g(\lambda x + (1-\lambda)y) &= f_j(\lambda x + (1-\lambda)y) \\ &\leq \lambda f_j(x) + (1-\lambda) f_j(y) \\ &\leq \lambda \max\{f_1(x), \ldots, f_m(x)\} + (1-\lambda) \max\{f_1(x), \ldots, f_m(x)\} \\ &= \lambda g(x) + (1-\lambda) g(y). \end{aligned}$$

□

Remark 4.32. (Hinge loss). The hinge loss function $g(x) = \max(0, 1 - x)$ is convex.

Question 4.33. Is the pointwise minimum of a set of convex functions convex? No. We can look at two quadratic functions offset horizontally.

Question 4.34. What about the pointwise minimum of concave functions? Using a similar proof to the one above, the pointwise minimum of concave functions is concave.

Rule 3 usually appears in "min-max" problems. Imagine, for instance, some worst-case scenario event. In the context of the Fermat-Weber problem, imagine now that we want to minimize the maximum distance between myself and any of my loved ones, i.e., $\min \max \|y - x_i\|$, where y is my location and x_i is the location of my i-th loved one. $\max \|y - x_i\|$ is a pointwise maximum of several norms, which are convex, so the maximum is convex.

4.9.4 Rule 4: Restriction to a line

Suppose $f : \mathbb{R}^n \to \mathbb{R}$, is convex. Fix two points $\overline{x}, \overline{y} \in \mathbb{R}^n$. Then $g(\alpha) = f[\overline{x} + \alpha(\overline{y} - \overline{x})]$, $g : \mathbb{R} \to \mathbb{R}$, is convex. We can imagine sliding along a line through $\overline{x}$ and $\overline{y}$.

Proof. Rule 4 follows immediately from Rule 2 (composition with an affine mapping). $\overline{x} + \alpha(\overline{y} - \overline{x})$ is an affine expression in α since $\overline{x}$ and $\overline{y}$ are fixed. □

We will see many algorithms that work by iteratively minimizing a function over lines. It will be useful to recall that the restriction of a convex function to a line remains convex. It is important to keep in mind that these (univariate) subproblems are also convex.

5 Applications of convex optimization in machine learning and statistics

5.1 Least absolute shrinkage and selection operator (LASSO)

This simple idea was introduced in one of the most cited papers in statistics (Tibshirani '96). Suppose we want to fit a regression model to some data. We could start with an objective function

$$P(x) = c_0 + c_1 x + \ldots + c_{10} x^{10} + c_{11} \sin(x) + \ldots + c_{100} \sin(50x) + c_{101} e^x + c_{102} e^{\cos(x)} + \ldots + c_{200} e^{\sin(x)\cos(e^x)}.$$

We want to solve

$$\min_{c_0,\ldots,c_{200}} \sum_{i=1}^{m} (p(x_i) - y_i)^2,$$

where we have m data points, $(x_i, y_i),\ i = 1, \ldots, m$. This is a classic example of overfitting. We have given our objective function so much freedom that it will perfectly fit the given data points, but between data points or with out-of-sample data points, the model will do a terrible job. (Note that this relates to our least squares problem since the objective function can be expressed in the form $\|Ac - b\|^2$.)

We can modify our objective function to incorporate some measure that penalizes the number of terms used in our objective function:

$$\min_{c_0,\ldots,c_{200}} \sum_{i=1}^{m} (p(x_i) - y_i)^2 + \lambda \|c\|_0,$$

where $\lambda \geq 0$ is fixed and $\|c\|_0$ is the ℓ_0 **pseudonorm** and is equal to the number of nonzero entries of c. Note that the ℓ_0 pseudonorm is *not* convex. (For starters, the ℓ_0 pseudonorm is not a norm because it violates homogeneity: $\|\alpha x\|_0 \neq \alpha \|x\|_0$.)

A common approach in practice is to replace $\|c\|_0$ with $\|c\|_1 = \sum_{i=1}^{n} |c_i|$. Thus, the LASSO estimator can be expressed as the solution to the following optimization problem, for $\lambda \geq 0$:

$$\min_{c_0,\ldots,c_{200}} \sum_{i=1}^{m} (p(x_i) - y_i)^2 + \lambda \|c\|_1,$$

or for least squares,

$$\min_{c} \|Ac - b\|^2 + \lambda \|c\|_1.$$

This is an unconstrained convex optimization problem.

Note that if λ is large, then more effort goes into minimizing the second term. If λ is small, then more effort goes into minimizing the first term.

5.2 Support vector machines (SVMs)

SVMs are an example of **supervised learning**: given as input a labeled dataset $\{(x_i, y_i)\}_{i=1}^{m}$, where $x_i \in \mathbb{R}^n$ is a **feature vector** and $y_i \in \{-1, +1\}$ is the **label** of data point x_i, we want to design a **classifier** $f : \mathbb{R}^n \to \mathbb{R}$ that allows you to label an unobserved data point.

Example 5.1. (IBM research). IBM wanted to predict which employees were likely to resign, since costs of hiring a replacement were not insignificant, and offer those specific employees raises. The team generated feature vectors, perhaps including each employee's current salary, length since their last promotion, education level, age, nuumber of family members, etc. $x_i \in \mathbb{R}^n$, $i = 1, \ldots, m$, represented each employee i's feature vector with n features.

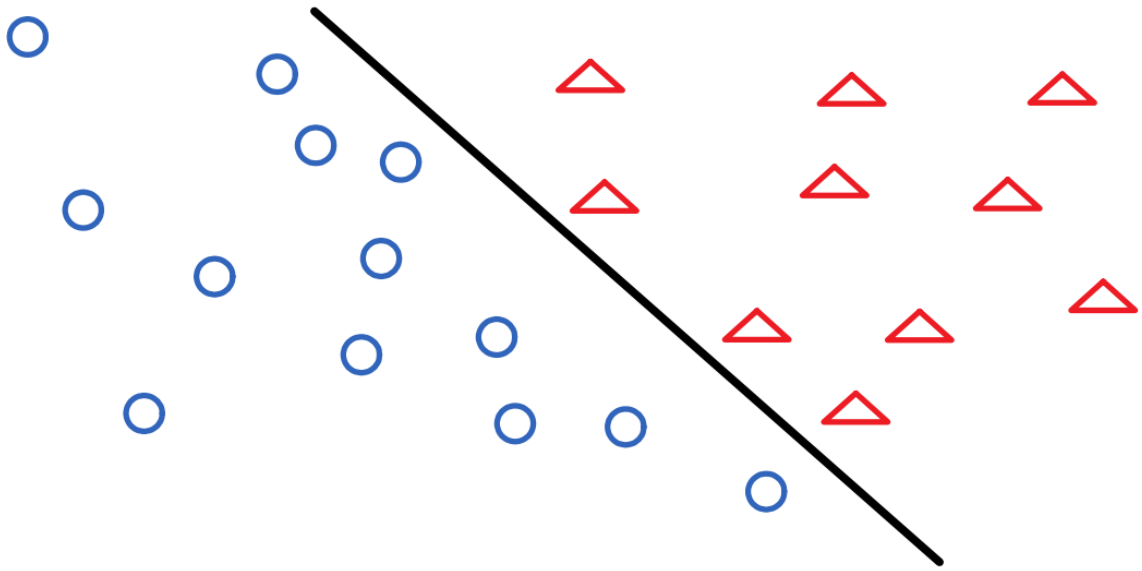
Example 5.2. (Insurance company). Will the new customer have an accident within one year? We can track, for instance, the following features of observations:

- Number of years since the customer got their license
- Number of previous accidents
- Number of traffic violations
- Age
- Education
- Gender.

These n features compose the feature vector $x_i \in \mathbb{R}^n$ for customer i (perhaps $n \approx 20 - 50$). What I have available is a **training set** of m previous customers (perhaps $m \approx 10^4$) with their measured feature vectors and labels $+1$ representing if they had an accident and -1 representing if they had no accident.

Example 5.3. (Spam classification).

We can visualize feature vectors with $n = 2$ in $\mathbb{R}^2$ (see the following illustration).



Usually, though, feature vectors will have much higher n. Blue circles could represent $y_i = +1$ and red triangles could represent $y_i = -1$; the output of our convex optimization is the parameters describing the line dividing the two classes of observations with label $+1$ and observations with label -1. We aim to find a function $f : \mathbb{R}^n \to \mathbb{R}$ such that

$$\begin{cases} f(x_i) < 0 \text{ if } y_i = -1, \\ f(x_i) > 0 \text{ if } y_i = +1. \end{cases}$$

Note that this is equivalent to finding a function that satisfies

$$y_i f(x_i) \geq 1, \ i = 1, \ldots, m.$$

A **linear classifier** takes the form $f(x) = a^T x - b$, $a \in \mathbb{R}^n$, $b \in \mathbb{R}$. Our goal is to find a and b that "best" separates the data into 2 classes. Note that we could have that the data is linearly separable or not linearly separable.

Case 1: Data is linearly separable.
When the data is linearly separable, there are infinite number of lines to pick; which affine function should we pick? Without loss of generality, we can modify a, b and find a, b such that

$$a^T x_i - b \geq +1 \text{ if } y_i = +1, \ i = 1, \ldots, m,$$
$$a^T x_i - b \leq -1 \text{ if } y_i = -1, \ i = 1, \ldots, m.$$

Here, we want a finite gap between the two classes of data. Note that the best affine function to pick is the line that maximizes the minimum distance to any data point. (We care most about points in the gray zone close to the boundary, but we want to keep the line as far away from this point as possible to minimize misclassification occurrences.)

Formally, in the linearly separable case, with $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, we have optimization problem (1):

$$\begin{aligned} \min_{a,b} \quad & \|a\| \\ \text{s.t.} \quad & y_i(a^T x_i - b) \geq 1, \ i = 1, \ldots, m. \end{aligned}$$

Note that this is a convex optimization problem and that the optimal solution is unique. (Why?) Also, note that we are minimizing $\|a\|$ since we want to *maximize* the distance between the hyperplane (our classifier) and any of our data points.

We can reformulate optimization problem (1) as optimization problem (2), with $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, $t \in \mathbb{R}$:

$$\begin{aligned} \max_{a,b,t} \quad & t \\ \text{s.t.} \quad & y_i(a^T x_i - b) \geq t, \ i = 1, \ldots, m, \\ & \|a\| \leq 1. \end{aligned}$$

This optimization problem is easier to interpret: we are maximizing the margin we have to help us be robust to noise. Note that we relax $\|a\| = 1$ to $\|a\| \leq 1$ to keep the constraint convex. Let's start analyzing these problems by making some claims.

Claim 5.4. *Optimization problems (1) and (2) are equivalent.*

Proof. We leave this as an exercise. □

Claim 5.5. *An optimal solution a^* to (2) satisfies $\|a^*\| = 1$.*

Proof. We leave this as an exercise. □

Claim 5.6. *The Euclidean distance between a point $v \in \mathbb{R}^n$ and a hyperplane $\{z \in \mathbb{R}^n \mid a^T z = b\}$ is given by*

$$\frac{|a^T v - b|}{\|a\|}.$$

Proof. We leave this as an exercise. □

How would we even detect if data is linearly separable or not? Data is linearly separable if and only if the "convex hulls" of the two sets (circles and triangles in the picture above) do not intersect.

Definition 5.7. (Convex hull). The smallest convex set that contains the set. Let S be a set consisting of s points $z_1, \ldots, z_k \subseteq \mathbb{R}^n$. The convex hull of S is defined as

$$\text{conv}(S) = \left\{ \sum_{i=1}^{k} \lambda_i s_i \mid s_i \in S,\ \lambda_i \geq 0,\ \sum_{i=1}^{k} \lambda_i = 1 \right\}.$$

Given two convex hulls for the two sets of data (circles and triangles), if they do not intersect, we can refer to the convex hulls as **support vectors** and have a **margin** between the two lines $a^{*T}x - b^* = -1$ and $a^{*T}x - b^* = +1$ of

$$\frac{2}{\|a^*\|},$$

where a^* is the optimal solution to convex optimization problem (1).

Now, how do we check if the convex hulls intersect? Let our decision variables be

$$\lambda_1, \ldots, \lambda_p \text{ for the set with label } -1,$$

$$\mu_1, \ldots, \mu_r \text{ for the set with label } +1.$$

$$\sum_{i=1}^{p} \lambda_i z_i = \sum_{i=1}^{r} \mu_i w_i,\ \lambda_i \geq 0,\ \mu_i \geq 0,\ \sum_{i=1}^{p} \lambda_i = 1,\ \sum_{i=1}^{r} \mu_i = 1.$$

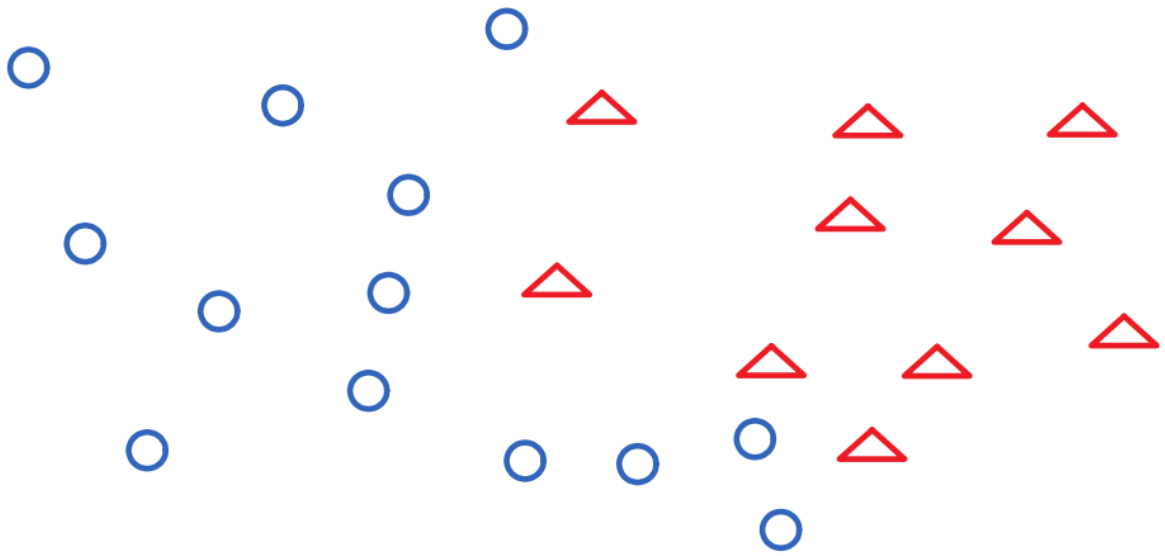
Can we solve this problem?

Case 2: Data is not linearly separable.
What if we cannot linearly separate the data? (See the following illustration in $\mathbb{R}^2$.)

We could minimize the number of points that get the wrong label. Alternatively, we could get a margin as large as possible (exactly as in the linearly separable case). We can relax our constraints from before, using a slack variable $\eta \in \mathbb{R}^m$, with $\eta_i \geq 0,\ i = 1, \ldots, m$:

$$a^T x_i - b_i \geq +1 - \eta_i \text{ if } y_i = +1,$$

$$a^T x_i - b_i \leq -1 + \eta_i \text{ if } y_i = -1.$$



What we really want to solve is the following optimization problem (3), for $\eta \in \mathbb{R}^m$,

$$\begin{array}{ll} \min\limits_{a,b,\eta} & \|\eta\|_0 \\ \text{s.t.} & y_i(a^T x_i - b) \geq 1 - \eta_i, \ i = 1, \ldots, m, \\ & \eta_i \geq 0, \ i = 1, \ldots, m, \end{array}$$

where the ℓ_0-norm $\|\eta\|_0$ is the number of non-zero elements of η.

Problem (3) exactly minimizes the number of misclassified points. Unfortunately, it is *not* a convex optimization problem. Instead, the following optimization problem (4) is a convex optimization problem and tends to yield a sparse solution:

$$\begin{array}{ll} \min\limits_{a,b,\eta} & \|\eta\|_1 \\ \text{s.t.} & y_i(a^T x_i - b) \geq t, \ i = 1, \ldots, m, \\ & \|a\| \leq 1. \end{array}$$

Let η^* be the optimal η. If $\eta_i^* = 0$, then the i-th data point is correctly classified. If $0 < \eta_i^* \leq 1$, then the i-th data point is correctly classified (it falls within our margin), though not robustly so. If $\eta_i^* > 1$, then the i-th data point is incorrectly classified.

We can also solve a modified optimization problem to balance the tradeoff between the number of misclassified points and the width of our margin, with $\gamma \geq 0$ fixed a priori:

$$\begin{array}{ll} \min\limits_{a,b,\eta} & \|a\| + \gamma\|\eta\|_1 \\ \text{s.t.} & y_i(a^T x_i - b) \geq t, \ i = 1, \ldots, m, \\ & \|a\| \leq 1. \end{array}$$

Using a larger γ means that we assign more importance to reducing the number of misclassified points, while using a smaller γ means that we assign more importance to having a large margin.

6 Unconstrained univariate minimization

In this section, we cover iterative methods for minimizing a function $f : \mathbb{R} \to \mathbb{R}$. Ideally, we want to minimize the number of function evaluations. Note that while a univariate optimization problem has limited use, we will apply one-dimensional line search in the future when we discuss multivariate optimization.

6.1 Bisection

Suppose that $f : \mathbb{R} \to \mathbb{R}$ is continuously differentiable. Assume that we know a (possibly large) interval $[a_0, b_0]$ in which the minimum of f lies and that the derivative of f changes sign only once in $[a_0, b_0]$. Furthermore, assume that there is only one local minimum in the interval. The bisection algorithm is as follows:

```
for k = 0:N
   if f'((x + y) / 2) = 0
      you are done;
   elseif f'((x + y) / 2) > 0
      a_{k+1} = a_k
      b_{k+1} = (a_k + b_k) / 2
   else
      b_{k+1} = b_k
      a_{k+1} = (a_k + b_k) / 2
   end
end
```

In each iteration, the length of the interval where the minimum lies is cut in half, i.e.,

$$|a_{k+1} - b_{k+1}| = \frac{1}{2}|a_k - b_k|.$$

Suppose that we are satisfied if we isolate the minimum within an interval of length ϵ. Then, this algorithm should take

$$N = \log_2 \frac{|a_0 - b_0|}{\epsilon}$$

number of steps.

We can also use bisection for root finding; in fact, that is what the algorithm is doing on f'. If f' is continuous, then bisection will find (a) root, and in each step, the root is sandwiched between our new endpoints.

Note that bisection is a safe and robust algorithm but is not as fast as the next two algorithms are. Since our intervals are halved in each step, bisection has "linear convergence," which we will formalize later.

In some applications, we won't know a priori an initial interval $[a_0, b_0]$. In this case, we can use the following strategy, assuming that $f(x) \to \infty$ as $|x| \to \infty$. To find a bracket containing the minimizer, it suffices to find three points $a < c < b$ such that $f(c) < f(a)$ and $f(c) < f(b)$.

```
pick x_0 < x_1 < x_2
if f(x_1) < f(x_0), f(x_1) < f(x_2)
```

```
   done
if f(x_0) > f(x_1) > f(x_2)
   pick x_3 > x_2 (e.g., such that (x_3 - x_2) = 2 (x_2 - x_1))
   if f(x_2) < f(x_3)
      done (bracket is [x_1, x_3])
   else
      continue (until f increases)
if f(x_0) < f(x_1) < f(x_2)
   ...similar process
```

A common use of bisection in optimization follows. Consider an optimization problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0. \end{aligned}$$

Suppose that we have a black box that can test for feasibility—it tells us whether the set $\{x \mid g_i(x) \leq 0\}$ is empty or not. How can we use the black box to solve our optimization problem? Note that our problem is equivalent to

$$\begin{aligned} \min \quad & \gamma \\ \text{s.t.} \quad & f(x) \leq \gamma, \\ & g_i(x) \leq 0. \end{aligned}$$

We can do bisection on γ. For each fixed γ, call the feasibility black box on the set $\{f(x) \leq \gamma, g_i(x) \leq 0\}$. If feasible, then decrease γ. If infeasible, increase γ. If we know an interval of length M where f^* lies, then we can get f^* with accuracy ϵ if we call our feasibility black box $\log_2 \frac{M}{\epsilon}$ times. Hence, in many cases, an efficient algorithm for feasibility testing directly gives an efficient algorithm for optimization.

6.2 Newton's method

Suppose now that we have access to f' and f''. Newton's method for minimization is based on the following iterations:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}.$$

We present two motivations for deriving this iterative algorithm:

1. Local quadratic approximation of f
2. Solving $f'(x) = 0$ by the "method of tangents."

6.2.1 Local quadratic approximation

Our motivation is to approximate f locally by a simpler function—for instance, a quadratic function q—that we know how to minimize. If f is quadratic itself, then our approximation is globally correct, and we terminate the algorithm in one step. If f is not quadratic, we move to the minimum of q and reapproximate f at that point with a new quadratic.

So at a point x_k, we wish to approximate f with a quadratic function

$$q(x) = ax^2 + bx + c,$$

and we want to match the following:

- $q(x_k) = f(x_k)$
- $q'(x_k) = f'(x_k)$
- $q''(x_k) = f''(x_k)$.

The q satisfying the three conditions is

$$q(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2,$$

which is just the second order Taylor expansion of f around x_k.

Assuming that $a > 0$, the minimum of q is achieved at $-\frac{b}{2a}$. The minimum is the new point we jump to:

$$x_{k+1} = -\frac{b}{2a} = x_k - \frac{f'(x_k)}{f''(x_k)}.$$

6.2.2 Root finding

Solving roots is actually the same thing as minimizing functions: we are, in effect, finding the roots of f' in hopes that they are local minima of f.

Let $g(x) = f'(x)$. At x_k, we approximate g with a line $y = mx + b$, the zero of which is our next point.

$$\begin{aligned} g(x_k) = g'(x_k)x_k + b &\implies b = g(x_k) - g'(x_k)x_k \\ &\implies 0 = g'(x_k)x_{k+1} + g(x_k) - g'(x_k)x_k \\ &\implies x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)} \\ &\implies x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \end{aligned}$$

While the Newton's method converges much faster than bisection (we will discuss quadratic convergence later), converge can be highly sensitive to initial conditions (iterations can move in the wrong direction, diverge, or converge very slowly if $g'(x^*) = 0$).

6.3 Secant method

The secant method is a simple modification of the Newton's method; we assume that we do not have access to f'' and, instead, approximate it with f':

$$f''(x_k) \approx \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}.$$

6.3.1 Local quadratic approximation

After substituting, we get

$$x_{k+1} = x_k - \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}} f'(x_k),$$

or equivalently,

$$x_{k+1} = \frac{f'(x_k)x_{k-1} - f'(x_{k-1})x_k}{f'(x_k) - f'(x_{k-1})}.$$

Note that we need two points to initialize the algorithm.

6.3.2 Root finding

The secant algorithm for root finding (solving $g(x) = f'(x) = 0$) is also straightforward as we can simply replace f' with g:

$$x_{k+1} = x_k - \frac{g(x_k) - g(x_{k-1})}{x_k - x_{k-1}} g(x_k),$$

or equivalently,

$$x_{k+1} = \frac{g(x_k)x_{k-1} - g(x_{k-1})x_k}{g(x_k) - g(x_{k-1})}.$$

7 Descent algorithms

We consider

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & x \in \mathbb{R}^n. \end{aligned}$$

How do we find a local optimum? How do we even find a stationary point? We cover some algorithms for this purpose. These will be iterative algorithms: we start at a point, jump to a new point with a hopefully lower objective value, and continue.

The general form of the iterations is

$$x_{k+1} = x_k + \alpha_k d_k.$$

$k = 0, 1, 2, \ldots$ indexes time, $x_k \in \mathbb{R}^n$ is the current point or iterate, $x_{k+1} \in \mathbb{R}^n$ is the next point or iterate, $d_k \in \mathbb{R}^n$ is the direction to move along at time k, and $\alpha_k \in \mathbb{R}, \alpha_k \geq 0$, is the step size at time k. The goal is to make the sequence $\{f(x_k)\}$ decrease as much as possible. We need to choose how to start, the descent direction d_k at each time, the step size α_k at each time, and when to stop.

7.1 Gradient descent methods

Note that throughout this discussion of gradient methods, we assume that f is at least C^1 (continuously differentiable). With gradient methods, the direction d_k to move along at step k is chosen based on information using $\nabla f(x_k)$. We discuss a couple reasons why $\nabla f(x_k)$ is a "good" vector to examine.

Lemma 7.1. *Suppose that you are at a point $x \in \mathbb{R}^n$ and locally examining the value of $f(x)$ in all directions around you. The vector $-\nabla f(x)$ is the direction with the maximum rate of decrease.*

Proof. Consider the function $g(\alpha) = f\left(x + \alpha \frac{d}{\|d\|}\right)$. $g'(0)$ gives the rate of decrease in direction d at point x:

$$g'(\alpha) = \frac{d^T}{\|d\|} \nabla f\left(x + \alpha \frac{d}{\|d\|}\right).$$

$$g'(0) = \frac{d^T}{\|d\|} \nabla f(x).$$

We claim, by Cauchy-Schwarz, that

$$-\frac{\|d\|}{\|d\|}\|\nabla f(x)\| \leq \frac{d^T \nabla f(x)}{\|d\|} \leq \frac{\|d\|}{\|d\|}\|\nabla f(x)\|$$

$$\implies -\|\nabla f(x)\| \leq g'(0) = \frac{d^T \nabla f(x)}{\|d\|} \leq \|\nabla f(x)\|.$$

We know that $\forall\ d$, $-\|\nabla f(x)\| \leq g'(0)$. The lower bound can be achieved if $d = -\nabla f(x)$:

$$\frac{-\nabla^T f(x) \nabla f(x)}{\|\nabla f(x)\|} = -\frac{\|\nabla f(x)\|^2}{\|\nabla f(x)\|} = -\|\nabla f(x)\|.$$

□

Definition 7.2. (Descent direction). For a given point $x \in \mathbb{R}^n$, a direction d is a descent direction for f at x if $\exists\ \overline{\alpha} > 0$ s.t. $f(x + \alpha d) < f(x)\ \forall\ \alpha \in (0, \overline{\alpha})$. Thus, there is a small enough (but nonzero) amount that you can move in direction d and be guaranteed to decrease the function value.

Lemma 7.3. *Consider a point $x \in \mathbb{R}^n$. Any direction $d \in \mathbb{R}^n$ that satisfies*

$$\nabla^T f(x) d = \langle \nabla f(x), d \rangle < 0$$

is a descent direction. (In particular, $-\nabla f(x)$ is a descent direction.)

Proof. Let's define $g(\alpha) = f(x + \alpha d)$. Taylor expand $g(\alpha)$ around $\alpha = 0$ to get

$$g(\alpha) = g(0) + g'(0)\alpha + o(\alpha).$$

So

$$f(x + \alpha d) - f(x) = d^T \nabla f(x) \alpha + o(\alpha).$$

Since $\lim_{\alpha \to 0} \frac{o(\alpha)}{\alpha} = 0$, there exists $\overline{\alpha} > 0$ such that $\frac{|o(\alpha)|}{\alpha} < |d^T \nabla f(x)|\ \forall\ \alpha \in (0, \overline{\alpha})$. Assuming that $d^T \nabla f(x) < 0$, this implies that $\forall\ \alpha \in (0, \overline{\alpha})$, we must have that $f(x + \alpha d) < f(x)$. Thus, d is a descent direction. □

Lemma 7.4. *Consider any positive definite matrix B. Then $-B\nabla f(x)$ is a descent direction at x (assuming $\nabla f(x) \neq 0$).*

Proof.

$$\langle \nabla f(x), -B\nabla f(x) \rangle = -\nabla^T f(x) B \nabla f(x) < 0.$$

By the previous lemma, $-B\nabla f(x)$ is a descent direction. □

This suggests a general paradigm for our descent algorithms:

$$x_{k+1} = x_k - \alpha_k B_k \nabla f(x_k),$$

where $B_k \succ 0,\ \forall\ k$.

7.1.1 Descent direction

How do we choose d_k? We can pick d_k to be one of many possible descent directions. All of the different well-known algorithms simply pick different B ($B \succ 0$) matrices. Typically, $\nabla f(x_k)$ appears in the construction of d_k. Some common choices of the descent direction include:

- **Steepest descent**: Pick $B_k = I,\ \forall\ k$. Thus, the direction that we choose is always $d_k = -\nabla f(x_k)$. Note that this choice is not always optimal as it might be too greedy.
- **Newton direction**: $B_k = (\nabla^2 f(x_k))^{-1}$, assuming that the Hessian is positive definite. Thus, $d_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$. Newton direction converges faster than steepest descent but is more expensive to construct.
- **Diagonally scaled steepest descent**: $B_k = \text{diag}(d_{1,k}, \ldots, d_{n,k}),\ d_{i,k} > 0$. For instance, we can take $d_{i,k} = \left(\frac{\partial^2 f(x_k)}{\partial x_i^2} \right)^{-1}$, i.e., diagonally approximate Newton.

- **Modified Newton direction**: Modified Newton direction addresses the tradeoff between construction costs and the rate of convergence. For instance, we could have:

$$B_k = \begin{pmatrix} \frac{\partial f}{\partial x_1 x_1}(x_k) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{\partial f}{\partial x_n x_n}(x_k) \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{\frac{\partial f}{\partial x_1 x_1}(x_k)} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \frac{1}{\frac{\partial f}{\partial x_n x_n}(x_k)} \end{pmatrix},$$

  or $B_k = (\nabla^2 f(x_0))^{-1}$, and update every M steps.

  Where does the Newton direction $-(\nabla^2 f(x_k))^{-1}\nabla f(x_k)$ come from? Recall how we updated from our previous discussion of Newton's method:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1}\nabla f(x_k).$$

  The second order approximation is

$$f(x) \approx f(x_k) + \nabla^T f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k) =: q(x).$$

  We want x_{k+1} to minimize $q(x)$. q is convex if $\nabla^2 f(x_k) \succeq 0$ (the Hessian of q is positive semidefinite). So f being convex implies that q is convex. Thus, it suffices to set $\nabla q(x) = 0$. The gradient is $\nabla q(x) = \nabla f(x_k) + \nabla^2 f(x_k)x - \nabla^2 f(x_k)x_k$:

$$\nabla q(x) = 0 \implies x = (\nabla^2 f(x_k))^{-1}(\nabla^2 f(x_k)x_k - \nabla f(x_k))$$

$$\implies x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1}\nabla f(x_k).$$

We now move back to the general form of descent algorithms: $x_{k+1} = x_k + \alpha_k d_k$.

7.1.2 Step size

How do we choose the step size α_k? Some options include:

- **Constant step size**: $\alpha_k = s > 0,\ \forall\ k$. While this is the simplest rule to implement, it may not converge if s is too large or maybe too slow if s is too small.
- **Diminishing step size**: $\{\alpha_k\}$ such that $\alpha_k \to 0$ as $k \to \infty$ and $\sum_{k=1}^{\infty} \alpha_k = \infty$. We could take $\alpha_k = \frac{1}{k}$, for instance. Descent is not guaranteed at each step but only later whne α_k becomes small.
- **Minimization rule (exact line search)**: Pick α_k that minimizes $f(x_k + \alpha d_k)$ over $\alpha \in [0, \infty)$. This can be expensive but is a univariate minimization problem—if f is convex, the one dimensional minimization problem is also convex. Thus, we can use the line search methods that we discussed previously.
- **Limited minimization**: Here, we pick α_k that minimizes $f(x_k + \alpha d_k)$ over $[0, s]$.
- **Successive step size reduction**: A well-known example is the Armijo rule, which we will cover later.

7.1.3 Stopping criteria

If $f(x)$ is convex, we stop when we reach $\nabla f(\overline{x}) = 0$ since $\overline{x}$ is a global min. Usually, we can stop when are within a certain threshold, such as $\epsilon = 0.001$. Common stopping criteria used in practice are:

- $\|\nabla f(x_k)\| \leq \epsilon$: Once $\nabla f(x_k) = 0$, our iterates stop moving—we have found a point satisfying the first order necessary condition for optimality, what we are aiming for.
- $|f(x_{k+1}) - f(x_k)| \leq \epsilon$: Improvements in the value of the function are saturating.
- $\|x_{k+1} - x_k\| \leq \epsilon$: Movement between iterates has become small.
- $\frac{|f(x_{k+1})-f(x_k)|}{\max\{1,|f(x_k)|\}} \leq \epsilon$: The added term removes dependence on the scale of f, and the max is taken to avoid dividing by small numbers.
- $\frac{\|x_{k+1}-x_k\|}{\max\{1,\|x_k\|\}} \leq \epsilon$.

7.1.4 Steepest descent with exact line search for quadratic functions

Our goal is to minimize

$$f(x) = \frac{1}{2}x^T Q x - b^T x, \; Q \succ 0.$$

We update using

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k).$$

We aim to choose $\alpha_k \geq 0$ to minimize $f(x_{k+1})$. Let

$$g(\alpha) := f(x_k - \alpha \nabla f(x_k)) = \frac{1}{2}(x_k - \alpha \nabla f(x_k))^T Q (x_k - \alpha \nabla f(x_k)) - b^T (x_k - \alpha \nabla f(x_k)).$$

Note that $g(\alpha)$ is quadratic in α and convex. We know that for $g(\alpha) = a\alpha^2 b\alpha + c$, $\arg\min g(\alpha) = -\frac{b}{2a}$, so

$$\alpha_k = \frac{\nabla^T f(x_k) \nabla f(x_k)}{\nabla^T f(x_k) Q \nabla f(x_k)}.$$

Thus, the overall algorithm is to update according to

$$x_{k+1} = x_k - \left(\frac{\nabla^T f(x_k) \nabla f(x_k)}{\nabla^T f(x_k) Q \nabla f(x_k)} \right) \nabla f(x_k),$$

where $\nabla f(x_k) = Q x_k - b$.

Theorem 7.5. *(The "zig-zag" theorem). Let $\{x_k\}$ be the sequence generated by the steepest descent algorithm. Then, for all k, $x_{k+1} - x_k$ is orthogonal to $x_{k+2} - x_{k+1}$.*

Proof. Our iterations are:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$
$$x_{k+2} = x_{k+1} - \alpha_{k+1} \nabla f(x_{k+1}).$$

So

$$\langle x_{k+1} - x_k, x_{k+2} - x_{k+1} \rangle = \alpha_k \alpha_{k+1} \langle \nabla f(x_k), \nabla f(x_{k+1}) \rangle.$$

Since α_k minimizes $\varphi_k(\alpha) := f(x_k - \alpha \nabla f(x_k))$, $\frac{\partial \varphi_k}{\partial \alpha}(\alpha_k) = 0$. Using the chain rule,

$$\frac{\partial \varphi_k}{\partial \alpha}(\alpha_k) = \langle -\nabla f(x_k), \nabla f(x_k - \alpha_k \nabla f(x_k)) \rangle = \langle -\nabla f(x_k), \nabla f(x_{k+1}) \rangle = 0.$$

Thus, $\langle x_{k+1} - x_k, x_{k+2} - x_{k+1} \rangle = 0$. □

7.1.5 Convergence

Definition 7.6. (Limit point). $x \in \mathbb{R}^n$ is a limit point of a sequence $\{x_k\}$ if there exists a subsequence of $\{x_k\}$ that converges to x.

Theorem 7.7. *Consider the sequence $\{x_k\}$ generated by any descent algorithm with $d_k = -B_k \nabla f(x_k)$ such that the eigenvalues of B_k are larger than some $m > 0$, for all k, and such that the step size is chosen according to the minimization rule or the limited minimization rule. Then, every limit point of $\{x_k\}$ is a stationary point.*

Remark 7.8. Note that a stationary point might not be a local minimum—it could be a saddle point or even a local maximum.

Theorem 7.9. *Consider a quadratic function $f(x) = \frac{1}{2} x^T Q x + b^T x + c$ with $Q \succ 0$. Let x^* be the minimizer of f.*

- *For the steepest descent algorithm with exact line search, $x_k \to x^*$ starting from any x_0, i.e. we have **global convergence**.*
- *For the steepest descent algorithm with a fixed step size, we have global convergence if and only if the step size α satisfies*

$$0 < \alpha < \frac{2}{\lambda_{\max}}(Q),$$

where $\lambda_{\max}(Q)$ denotes the maximum eigenvalue of Q.

Once we know that an iterative algorithm converges, we now ask: how fast does it converge?

Definition 7.10. (Rate of convergence). Let $\{x_k\}$ converge to x^*. We say that the converge is of **order** $p(\geq 1)$ and with **factor** $\gamma(> 0)$ if there exists k_0 such that $\forall\ k \geq k_0$,

$$\|x_{k+1} - x^*\| \leq \gamma \|x_k - x^*\|^p.$$

Note that the larger the order p, the faster the convergence, and for the same p, the smaller the γ, the faster the convergence. If $\{x_k\}$ converges with order p, then it also converges with order p' for any $p' \leq p$. If $\{x_k\}$ converges with order p and factor γ, then it also converges with order p and factor γ' for any $\gamma' \geq \gamma$. So we typically look for the largest p and the smallest γ for which the inequality holds.

- **Linear convergence**: $p = 1$ and $\gamma < 1$. This is "linear" for the following reason: for k that is large enough, we have $\|x_{k+i} - x^*\| \leq \gamma^i \|x_k - x^*\|$, so $\log \|x_{k+i} - x^*\| \leq i \log \gamma + \log \|x_k - x^*\|$. Thus, $-\log \|x_{k+1} - x^*\|$, which is a measure of the number of correct significant digits in x_{k+i}, grows linearly with i. Ex: $\|x_k - x^*\| \approx a^k,\ 0 < a < 1$.

- **Sublinear convergence**: $p = 1$ and $\gamma = 1$. Ex: $\|x_k - x^*\| \approx \frac{1}{k}$.
- **Superlinear convergence**: $p > 1$. This is slightly stronger than the usual definition of superlinear convergence.
- **Quadratic convergence**: $p = 2$. For quadratic convergence, the number of correct significant digits doubles in each iteration. Ex: $\|x_k - x^*\| \approx a^{2^k},\ 0 < a < 1$.

Remark 7.11. How do we find the order and factor of convergence? Recall that we want the largest p for which the inequality $\|x_{k+1} - x^*\| \leq \gamma \|x_k - x^*\|^p$ holds for large k. For that p, we want the smallest γ for which the inequality holds. A simple way to find p and γ is:

- Take the limit $\lim_{k\to\infty} \frac{\|x_{k+1}-x^*\|}{\|x_k-x^*\|^p}$.
- The order p is the largest number for which the limit is finite.
- Plug in the largest p from the previous step. The factor γ is the value of the limit.

Theorem 7.12. *(Convergence rate of steepest descent with exact line search for quadratic functions). Consider a quadratic function $f(x) = \frac{1}{2}x^TQx + b^Tx + c$ with $Q \succ 0$. Let m and M denote the smallest eigenvalue and largest eigenvalue of Q, respectively. Then, the sequence $\{f(x_k)\}$ generated by the steepest descent algorithm with exact line search converges to the unique global minimum of f with linear convergence (order of 1) and with factor $\left(\frac{M-m}{M+m}\right)^2$.*

Remark 7.13. $\kappa := \frac{M}{m}$ is called the **condition number** of the matrix Q. Note that $\kappa \geq 1$ and that $\left(\frac{M-m}{M+m}\right)^2 = \left(\frac{\kappa-1}{\kappa+1}\right)^2$. We want κ to be small for fast convergence.

Remark 7.14. What if the function f that we are minimizing is not quadratic? Denote the optimal solution by x^*. Locally, the function can be well approximated by a quadratic $\frac{1}{2}x^T\nabla^2 f(x^*)x$ (plus linear and constant terms). $\kappa(\nabla^2 f(x^*))$, the condition number of the Hessian at x^*, dictates the convergence rate.

7.2 Multivariate Newton's method

Newton's method, now generalized to multible dimensions, has a quadratic rate of convergence and is nothing but a descent method with a specific choice of a ddescent direction—one that iteratively adjusts itself to the local geometry of the function to be minimized. In practice, it converges much faster than gradient methods; for quadratic functions, for instance, while gradient methods can zigzag for a long time, Newton's method reaches the optimal solution in simply one step. However, we need access to second order information and need to solve a linear system of equations at every step of the algorithm.

7.2.1 Descent direction

Recall our general form of our descent methods:

$$x_{k+1} = x_k + \alpha_k d_k.$$

For now, we have

$$\alpha_k = 1 \ \forall\ k,$$

sometimes referred to as the pure Newton's method. Newton's method simply chooses

$$d_k = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

for the descent direction. Iteration is only well-defined when the Hessian at x_k is invertible. Recall that one motivation for Newton's method is minimizing a quadratic approximation of the function sequentially: the Taylor expansion of f around x_k is

$$f(x) \approx f(x_k) + \nabla^T f(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k) =: q(x).$$

q is lower bounded if and only if $\nabla^2 f(x_k) \succeq 0$. The first order necessary condition requires that $\nabla q(x^*) = 0$, so we aim to satisfy

$$\nabla f(x_k) + \nabla^2 f(x_k)x - \nabla^2 f(x_k)x_k = 0 \Longrightarrow x = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k).$$

If $\nabla^2 f(x_k) \succeq 0$, then q is convex, and our stationary point is globally optimal. Newton's method picks this point as the next iterate:

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k).$$

7.2.2 Convergence

Lemma 7.15. *Let $A \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$. Let $\|\cdot\|$ denote a vector norm and also its associated induced matrix norm. Then, $\|Ax\| \leq \|A\|\|x\|$.*

Proof. By definition, $\|A\| := \max_{\|y\|=1} \|Ay\|$. Suppose that there exists $x \in \mathbb{R}^n$ such that $\|Ax\| > \|A\|\|x\|$. Then,

$$\left\| A \frac{x}{\|x\|} \right\| = \frac{1}{\|x\|} \|Ax\| > \|A\|,$$

which contradicts the definition of the induced norm. □

Lemma 7.16. *Let $A : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ be a matrix valued function that is continuous at a point $u \in \mathbb{R}^n$. If $A(u)^{-1}$ exists, then there exists a scalar $\epsilon > 0$ such that $A(v)^{-1}$ also exists for all $v \in B(u, \epsilon)$.*

Remark 7.17. Notation: For $x^* \in \mathbb{R}^n$, $\epsilon \in \mathbb{R}_+$, let $B(x^*, \epsilon) := \{z \in \mathbb{R}^n \mid \|z - x^*\| \leq \epsilon\}$.

Theorem 7.18. *Suppose that $f \in C^3$ (three times continuously differentiable) and that $x^* \in \mathbb{R}^n$ is such that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is invertible. Then, there exists $\epsilon > 0$ such that iterations of Newton's method starting from any point $x_0 \in B(x^*, \epsilon)$ are well-defined and converge to x^*. Moreover, the convergence rate is quadratic.*

Remark 7.19. Note that this is simply a local statement—there is no guarantee that Newton iterations would converge if we start far away. Furthermore, there is no guarantee that our limit point is a local minimum—it could potentially even be a local maximum.

Proof. The Taylor expansion of ∇f around x_0 is

$$\nabla f(x) = \nabla f(x_0) + \nabla^2 f(x_0)(x - x_0) + o(\|x - x_0\|^2).$$

By Taylor's rule, there exist $\epsilon_1, c_1 > 0$ such that if $x_0, x \in B(x^*, \epsilon_1)$, then

$$\|\nabla f(x) - \nabla f(x_0) - \nabla^2 f(x_0)(x - x_0)\| \leq c_1 \|x - x_0\|^2.$$

Since $\nabla^2 f(x^*)$ is invertible, by the previous lemma, there exist $\epsilon_2, c_2 > 0$ such that $\forall\ x \in B(x^*, \epsilon_2)$, $\nabla^2 f(x)$ exists and $\|(\nabla^2 f(x))^{-1}\| \leq c_2$.

Let $\epsilon = \min\{\epsilon_1, \epsilon_2\}$. Then, $x_0, x \in B(x^*, \epsilon) \Longrightarrow$:

- $\|\nabla f(x) - \nabla f(x_0) - \nabla^2 f(x_0)(x - x_0)\| \leq c_1 \|x - x_0\|^2$.
- $\nabla^2 f(x)$ exists and $\|(\nabla^2 f(x))^{-1}\| \leq c_2$.

Suppose that $x_0 \in B(x^*, \epsilon)$. Substituting $x = x^*$ above and recalling that $\nabla f(x^*) = 0$, we get

$$\|\nabla^2 f(x_0)(x_0 - x^*) - \nabla f(x_0)\| \leq c_1 \|x^* - x_0\|^2.$$

Recall the Newton iterations:

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k).$$

Writing this for $k = 0$, subtracting x^* from both sides, and taking norms, we get

$$\begin{aligned} \|x_1 - x^*\| &= \|x_0 - x^* - (\nabla^2 f(x_0))^{-1} \nabla f(x_0)\| \\ &= \|\|(\nabla^2 f(x_0))^{-1}(\nabla^2 f(x_0)(x_0 - x^*) - \nabla f(x_0))\| \\ &\leq \|(\nabla^2 f(x_0))^{-1}\| \|\nabla^2 f(x_0)(x_0 - x^*) - \nabla f(x_0)\|. \end{aligned}$$

Thus,

$$\|x_1 - x^*\| \leq c_1 c_2 \|x_0 - x^*\|^2.$$

Let x_0 be such that

$$\|x_0 - x^*\| \leq \frac{\alpha}{c_1 c_2},$$

where $0 < \alpha < 1$.

$$\Longrightarrow \|x_1 - x^*\| \leq \alpha \|x_0 - x^*\|.$$

This imples that $x_1 \in B(x^*, \epsilon)$ and $\|x_1 - x^*\| \leq \frac{\alpha}{c_1 c_2}$. The same arguments apply to x_1, so we conclude

$$\|x_2 - x^*\| \leq c_1 c_2 \|x_1 - x^*\|^2,\ \|x_2 - x^*\| \leq \alpha \|x_1 - x^*\|.$$

By induction,

$$\|x_{k+1} - x^*\| \leq c_1 c_2 \|x_k - x^*\|^2,\ \|x_{k+1} - x^*\| \leq \alpha \|x_k - x^*\|,$$

which is quadratic convergence.

$$\Longrightarrow \|x_k - x^*\| \leq \alpha^k \|x_0 - x^*\|.$$

Since $\alpha < 1$, $\{x_k\} \to x^*$. □

7.2.3 Step size (Armijo rule)

Newton's method does not guarantee descent in every iteration; for instance, we might have $f(x_{k+1}) \geq f(x_k)$. But the Newton direction is a descent direction for convex functions, meaning that if we take a small enough step, we should get a decrease.

Recall that, for a given point $x \in \mathbb{R}^n$, a direction $d \in \mathbb{R}^n$ is called a descent direction if there exists $\overline{\alpha} > 0 (\alpha \in \mathbb{R})$ such that $f(x + \alpha d) < f(x),\ \forall\ \alpha \in (0, \overline{\alpha})$.

Theorem 7.20. *At any point $x \in \mathbb{R}^n$ where $\nabla f(x) \neq 0$ and $\nabla^2 f(x) \succ 0$, the Newton direction $-(\nabla^2 f(x))^{-1}\nabla f(x)$ is a descent direction.*

Proof. Define $h(\alpha) = f(x - \alpha(\nabla^2 f(x))^{-1}\nabla f(x))$. Then,

$$h'(\alpha) = (-(\nabla^2 f(x))^{-1}\nabla f(x))^T \nabla f(x - \alpha(\nabla^2 f(x))^{-1}\nabla f(x)),$$

$$h'(0) = -\nabla^T f(x)(\nabla^2 f(x))^{-1}\nabla f(x).$$

We have that $\nabla f(x) \neq 0$ and $\nabla^2 f(x) \succ 0 \implies (\nabla^2 f(x))^{-1} \succ 0$. Thus, $h'(0) < 0$. Thus, there exists $\overline{\alpha} > 0$ such that $h(\alpha) < h(0)\ \forall\ \alpha \in (0, \overline{\alpha})$. □

We now discuss a popular choice of the step size α_k for Newton's method, with

$$x_{k+1} = x_k - \alpha_k(\nabla^2 f(x_k))^{-1}\nabla f(x_k).$$

The Armijo rule is an **inexact line search method**—it does not find the exact minimum along a line but guarantees a sufficient and inexpensive decrease. The Armijo rule requires two parameters: $\epsilon \in (0, 1)$ and $\delta > 1$.

Suppose that we want to minimize a univariate function $h(\alpha)$ over $\alpha \geq 0$. (For us, $h(\alpha) = f(x_k + \alpha d_k)$, where x_k and d_k are fixed and d_k is the Newton direction.) Define an affine univariate function

$$\hat{h}(\alpha) = h(0) + \epsilon h'(0)\alpha.$$

The Armijo rule accepts a stepsize $\overline{\alpha}$ if:

- $h(\overline{\alpha}) \leq \hat{h}(\overline{\alpha})$ (ensures sufficient decrease)
- $h(\delta\overline{\alpha}) \geq \hat{h}(\delta\overline{\alpha})$ (ensures that the stepsize is not too small).

The Armijo backtracking algorithm is detailed below:

- Start with some initial step size α_0.
- At iteration j:
  - If $h(\alpha_j) \leq \hat{h}(\alpha_j)$, stop. Declare α_j as your step size.
  - If $h(\alpha_j) > \hat{h}(\alpha_j)$, let $\alpha_{j+1} = \frac{1}{\delta}\alpha_j$.

7.2.4 Levenberg-Marquardt modification

Note that if $\nabla^2 f(x) \not\succ 0$, then Newton's direction might not be a descent direction. If $\nabla^2 f(x)$ is singular, then Newton's direction won't even be well-defined. Thus, we examine the idea of making $\nabla^2 f(x)$ positive definite if it is not:

$$x_{k+1} = x_k - (\nabla^2 f(x_k) + \mu_k I)^{-1} \nabla f(x_k),\ \mu_k \geq 0.$$

If μ_k is large enough, then $-(\nabla^2 f(x_k) + \mu_k I)^{-1} \nabla f(x_k)$ will be a descent direction, and by choosing a small enough step size α_k, we can ensure descent. As $\mu_k \to 0$, we approach the ordinary Newton's method. As $\mu_k \to \infty$, we approach a pure gradient method with a small step size. In practice, we can start with a small value of μ_k, increasing it slowly until we observe descent ($f(x_{k+1}) < f(x_k)$).

Lemma 7.21. *Let $A \in \mathbb{R}^{n \times n}$ have eigenvalues $\lambda_1, \ldots, \lambda_n$, and let $\mu \in \mathbb{R}$. Then, the eigenvalues of $A + \mu I$ are $\lambda_1 + \mu, \ldots, \lambda_n + \mu$.*

Proof. Let λ_i be an eigenvalue of A corresponding to eigenvector v_i.

$$(A + \mu I)v_i = Av_i + \mu v_i = \lambda_i v_i + \mu v_i = (\lambda_i + \mu)v_i$$

$\implies \lambda_i + \mu$ is an eigenvalue of $A + \mu I$. □

7.2.5 Gauss-Newton method for nonlinear least squares

Let's say we have some function $f(a, \omega, \phi, b) = (\cdot)^2$ that is nonlinear in a, ω, ϕ, b. Our goal is to minimize f, which is a **nonlinear least squares problem**. More generally, we could have a list of functions $g_i(x) : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$ and aim to minimize $f(x) = \sum_{i=1}^m g_i^2(x)$. The Gauss-Newton method is an approximation of Newton's method for minimizing this function.

More formally, given (possibly nonlinear) functions $g_i(x) : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$ (typically $m \geq n$), we want to minimize

$$f(x) = \frac{1}{2} \sum_{i=1}^m g_i^2(x).$$

Define $g : \mathbb{R}^n \to \mathbb{R}^m$ as $g = \begin{pmatrix} g_1 \\ \vdots \\ g_m \end{pmatrix}$. Let $J(x) \in \mathbb{R}^{m \times n}$ be the Jacobian matrix of g, i.e., $J_{i,j}(x) = \frac{\partial g_i(x)}{\partial x_j}$. Then, the Gauss-Newton iteration is

$$x_{k+1} = x_k - (J(x_k)^T J(x_k))^{-1} J(x_k)^T g(x_k).$$

Note that this method only uses first order information. $-(J(x_k)^T J(x_k))^{-1} J(x_k)^T g(x_k)$ is a descent direction, since $J(x_k)^T g(x_k)$ is the gradient of f and $(J(x_k)^T J(x_k))^{-1} \succeq 0$. If $J(x_k)^T J(x_k) \not\succ 0$, we can apply the Levenberg-Marquardt modification.

The Newton iteration for minimizing f is therefore

$$x_{k+1} = x_k - \left(J(x_k)^T J(x_k) + \sum_{i=1}^m \nabla^2 g_i(x_k) g_i(x_k) \right)^{-1} J(x_k)^T g(x_k).$$

If g_i is linear for $i = 1, \ldots, m$, the problem is a linear least squares problem, Gauss-Newton equals Newton, and one iteration is enough to solve the problem globally.

Here's how we derive the Gauss-Newton method:

1. Replace $g(x)$ with its first order approximation $\tilde{g}(x, x_k)$ near the current iterate x_k.
2. Minimize $\frac{1}{2}\|\tilde{g}(x, x_k)\|^2$, which is now a convex quadratic function.

$$\frac{1}{2}\|\tilde{g}(x, x_k)\|^2 = \frac{1}{2}\left[\|g(x_k)\|^2 + 2(x - x_k)^T J^T(x_k)g(x_k) + (x - x_k)^T J^T(x_k)J(x_k)(x - x_k)\right]$$

$$\implies x^* = x_k - (J^T(x_k)J(x_k))^{-1} J^T(x_k)g(x_k).$$

7.3 Conjugate direction methods ***

We now discuss a class of descent methods, primarily developed for minimizing quadratic functions. Conjugate direction methods are, in some sense, an intermediate between gradient descent methods and Newton's method as they try to accelerate the convergence rate of steepest descent without paying the overhead of Newton's method. They have the following properties:

- They minimize a quadratic function in n variables in n steps.
- Evaluating and storing the Hessian is not required.
- We don't need to invert a matrix, unlike Newton's method.

Conjugate direction methods are also used for solving large-scale linear systems, particularly those defined by a positive definite matrix.

We are mostly concerned with minimizing

$$f(x) = \frac{1}{2}x^T Qx - b^T x, \; Q \succ 0.$$

7.3.1 Conjugate directions

Definition 7.22. (Q-conjugate). Let $Q \in \mathbb{R}^{n \times n}$ be a real symmetric matrix. We say that a set of non-zero vectors $d_1, \ldots, d_m \in \mathbb{R}^m$ are Q-conjugate if $d_i^T Q d_j = 0 \; \forall \; i, j, i \neq j$.

Lemma 7.23. *Let Q be an $n \times n$ symmetric positive definite matrix. If the directions $d_0, d_1, \ldots, d_k \in \mathbb{R}^n, \; k \leq n - 1$, are Q-conjugate, then they are linearly independent.*

Why $k \leq n - 1$? We cannot have more than n linearly independent vectors in $\mathbb{R}^n$. Thus, we cannot have more than n vectors that are Q-conjugate.

Proof. Suppose that the direcitons are linearly dependent, so there exist $c_0, \ldots, c_k$ that are not all zero such that $c_0 d_0 + c_1 d_1 + \ldots + c_k d_k = 0$. WLOG, assume that $c_0 \neq 0$. Multiplying both sides by $d_0^T Q$ gives us

$$c_0 d_0^T Q d_0 + c_1 d_0^T Q d_1 + \ldots + c_k d_0^T Q d_k = 0 \implies c_0 d_0^T Q d_0 = 0,$$

since most terms vanish due to Q-conjugacy. Since $d_0 \neq 0$ and $Q \succ 0$, we must have that $d_0^T Q d_0 > 0$—a contradiction. □

The **conjugate direction algorithm** is detailed below. Our input is an $n \times n$ matrix $Q \succ 0$, a vector $b \in \mathbb{R}^n$, and a set of n Q-conjugate directions $d_0, \ldots, d_{n-1}$. We pick an initial point $x_0 \in \mathbb{R}^n$. For $k = 0$ to $n-1$:

- Let $g_k := \nabla f(x_k) = Qx_k - b$.
- Let $\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}$.
- Let $x_{k+1} = x_k + \alpha_k d_k$.

We basically have n conjugate directions and minimize $f(x) = \frac{1}{2}x^T Q x - b^T x$ by doing exact line search iteratively along the n conjugate directions.

Lemma 7.24. *The step size α_k given in the conjugate direction algorithm gives the exact minimum along the direction d_k.*

Proof. We want to minimize $h(\alpha) = f(x_k + \alpha d_k)$, which is a univariate convex function. Thus, we can simply find a stationary point $\frac{d}{d\alpha}h(\alpha) = 0$. By the chain rule,

$$\frac{d}{d\alpha}h(\alpha) = d_k^T \nabla f(x_k + \alpha d_k) = d_k^T (Q(x_k + \alpha d_k) - b) = \alpha d_k^T Q d_k + d_k^T (Qx_k - b) = \alpha d_k^T Q d_k + d_k^T g_k.$$

$$h'(\alpha) = 0 \Longrightarrow \alpha^* = -\frac{d_k^T g_k}{d_k^T Q d_k}.$$

(Note: $d_k^T Q d_k > 0$.) □

Theorem 7.25. *For any starting point $x_0 \in \mathbb{R}^n$, the conjugate direction algorithm converges to the uinque minimum x^* of $f(x) = \frac{1}{2}x^T Q x - b^T x$ in n steps, i.e., $x_n = x^*$.*

Proof. Let x^* be the optimal solution. Let x_0 be any initial point in $\mathbb{R}^n$. The n-th iteration of the conjugate gradient algorithm produces

$$x_n = x_0 + \alpha_0 d_0 + \alpha_1 d_1 + \ldots + \alpha_{n-1} d_{n-1},$$

where $d_0, \ldots, d_{n-1}$ are our input Q-conjugate directions and, for $0 \leq k \leq n-1$,

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}, \ (g_k := \nabla f(x_k) = Qx_k - b).$$

The goal is to show that $x_n = x^*$.

$d_0, \ldots, d_{n-1}$ are linearly independent as they are Q-conjugate, so they span $\mathbb{R}^n$. In particular, we can write

$$x^* - x_0 = \beta_0 d_0 + \beta_1 d_1 + \ldots + \beta_{n-1} d_{n-1}$$

for some scalars $\beta_0, \ldots, \beta_{n-1}$. For $0 \leq k \leq n-1$, multiplying both sides by $d_k^T Q$ gives

$$d_k^T Q(x^* - x_0) = \beta_k d_k^T Q d_k.$$

Thus,

$$\beta_k = \frac{d_k^T Q(x^* - x_0)}{d_k^T Q d_k}.$$

We claim that $\beta_k = \alpha_k$ for $0 \leq k \leq n-1$. Once we prove this, we are done. We show that

$$d_k^T Q(x^* - x_0) = -d_k^T g_k = d_k^T Q(x^* - x_k).$$

Indeed, since $x^* - x_0 = x^* - x_k + x - k - x_0$ and $x_k - x_0 = \alpha_0 d_0 + \ldots + \alpha_{k-1} d_{k-1}$, multiplying both sides by $d_k^T Q$ gives

$$d_k^T Q(x_k - x_0) = 0 \Longrightarrow d_k^T Q(x^* - x_0) = d_k^T Q(x^* - x_k).$$

□

How do we compute the Q-conjugate directions from the matrix Q? We propose the conjugate Gram-Schmidt procedure.

7.3.2 Conjugate Gram-Schmidt algorithm

Theorem 7.26. *Let Q be an $n \times n$ positive definite matrix, and let $\{v_1, \ldots, v_{n-1}\}$ be a set of linearly independent vectors. Then, the vectors $\{d_0, \ldots, d_{n-1}\}$ constructed as follows are Q-conjugate and span the same space as $\{v_1, \ldots, v_{n-1}\}$:*

$$d_0 = v_0,$$

$$d_{i+1} = v_{i+1} - \sum_{m=0}^{i} \frac{v_{i+1}^T Q d_m}{d_m}, \ i = 0, \ldots, n-1.$$

Proof. Take $d_0 = v_0$. For some $i < n-1$, suppose that $d_0, \ldots, d_i$ are chosen in a way such that they are Q-conjugate and span $\{d_0, \ldots, d_i\} = \text{span}\{v_0, \ldots, v_i\}$. Pick d_{i+1} to be of the form

$$d_{i+1} = v_{i+1} + \sum_{m=0}^{i} c_{i+1,m} d_m,$$

where the coefficients $c_{i+1,m}$ are chosen in a way that d_{i+1} is Q-conjugate to $d_0, \ldots, d_i$. For any $0 \leq j \leq i$, we should have

$$d_j^T Q d_{i+1} = d_j^T Q v_{i+1} + d_j^T Q c_{i+1,j} d_j = 0,$$

where we use the Q-conjugacy of $d_0, \ldots, d_i$. Thus,

$$c_{i+1,j} = -\frac{d_j^T Q v_{i+1}}{d_j^T Q d_j}.$$

Note that $d_{i+1} \neq 0$. ($d_{i+1} = 0$ would imply that $v_{i+1} \in \text{span}\{d_0, \ldots, d_i\} \Longrightarrow v_{i+1} \in \text{span}\{v_0, \ldots, v_i\}$, a contradiction since $v_0, \ldots, v_{i+1}$ are assumed to be linearly independent.)

Finally, we show that $\text{span}\{d_0, \ldots, d_{i+1}\} = \text{span}\{v_0, \ldots, v_{i+1}\}$. $v_{i+1} \in \text{span}\{d_0, \ldots, d_{i+1}\}$ is clear from how we pick d_{i+1}, and $d_{i+1} \in \text{span}\{v_0, \ldots, v_{i+1}\}$ also holds. Thus, $\text{span}\{d_0, \ldots, d_{i+1}\} = \text{span}\{v_0, \ldots, v_{i+1}\}$. We have proven the theorem by induction. □

7.3.3 Conjugate gradient (CG) algorithm

We now discuss the conjugate gradient algorithm, which finds conjugate directions in each iteration and is more efficient than the Gram-Schmidt process, which requires a stack of all previous conjugate directions in memory.

Lemma 7.27. *Let Q be an $n \times n$ symmetric positive definite matrix, $f(x) = \frac{1}{2}x^TQx - b^Tx$, $\{d_0, \ldots, d_{n-1}\}$ a set of Q-conjugate directions, x_0 an arbitrary point in $\mathbb{R}^n$, and $\{x_0, \ldots, x_n\}$ the sequence of points generated by the conjugate directions algorithm. Let $g_k := \nabla f(x_k) = Qx_k - b$. Then,*

$$g_{k+1}^T d_k = 0,$$

for all $k = 0, \ldots, n-1$ and $i = 0, \ldots, k$.

Proof. □

Theorem 7.28. *Let Q be an $n \times n$ symmetric positive definite matrix, $f(x) = \frac{1}{2}x^TQx - b^Tx$, $\{d_0, \ldots, d_{n-1}\}$ a set of Q-conjugate directions, x_0 an arbitrary point in $\mathbb{R}^n$, and $\{x_0, \ldots, x_n\}$ the sequence of points generated by the conjugate directions algorithm. Let*

$$M_k = \{x \mid x = x_0 + v,\ v \in \mathit{span}\{d_0, \ldots, d_k\}\}.$$

Then, for $k = 0, \ldots, n-1$, x_{k+1} minimizes f over M_k.

Remark 7.29. Note that $M_{n-1} = \mathbb{R}^n$. So the algorithm indeed terminates with the optimal solution in n steps.

Proof. □

7.3.4 Solving linear systems

8 Linear programming (LP)

Linear programming (LP) is a subclass of convex optimization problems in which both the objective function and the constraints are affine functions. Linear programming is essentially about solving systems of linear inequalities. The **standard form** of a **linear program** is

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

where $c \in \mathbb{R}^n,\ b \in \mathbb{R}^m,\ A \in \mathbb{R}^{m \times n}$. Not all linear programs are in standard form, but they can all be rewritten in standard form using simple transformations.

Recall the largest independent set problem: given an undirected graph, find the largest collection of nodes among which no two share an edge.

$$\begin{aligned} \max \quad & x_1 + x_2 + \cdots + x_{12} \\ \text{s.t.} \quad & x_1 + x_2 \leq 1 \\ & x_1 + x_8 \leq 1 \\ & \vdots \\ & x_1 2 + x_8 \leq 1 \\ & x_i(1 - x_i) = 0,\ i = 1, \ldots, 12. \end{aligned}$$

We can write this problem as a linear program with integer constraints, so the problem can be called an **integer program** or a **linear integer program**. IPs are generally hard to solve, but we can obtain the **LP relaxation** of this problem by replacing the constraint $x_i \in \{0, 1\}$ with $x_i \in [0, 1]$:

$$\begin{aligned} \max \quad & x_1 + x_2 + \cdots + x_{12} \\ \text{s.t.} \quad & x_1 + x_2 \leq 1 \\ & x_1 + x_8 \leq 1 \\ & \vdots \\ & x_1 2 + x_8 \leq 1 \\ & 0 \leq x_i \leq 1,\ i = 1, \ldots, 12. \end{aligned}$$

Note that the optimal solution to the LP is an upper bound to the optimal solution to the IP.

An LP does not need to appear in standard form; there are several ways in which a linear program can differ from the standard form:

- It is a maximization problem instead of a minimization problem.
- Some constraints are inequalities instead of equalities ($a_i^T x \geq b_i$).
- Some variables are unrestricted in sign.

There are several possibilities for the solution of an LP:

- Infeasible vs. feasible.
- Unbounded vs. bounded.
- Inifinite vs. finite (including unique) number of optimal solutions.

8.1 Geometry of linear programming

The simplex algorithm, which we explore later, exploits the geometry of linear programming in a very fundamental way. We first prove some basic geometric results that are essential to the algorithm.

Recall that a hyperplane is a set $\{x \mid a^T x = b\}$, where $a \in \mathbb{R}^n$ is a nonzero vector. A half-space is a set $\{x \mid a^T x \geq b\}$, where $a \in \mathbb{R}^n$ is a nonzero vector.

Definition 8.1. (Polyhedron). The intersection of finitely many half-spaces is called a polyhedron. This is always a convex set, as half-spaces are convex, and intersections of convex sets are convex.

Definition 8.2. (Bounded set). A set $S \subset \mathbb{R}^n$ is bounded if there exists $K \in \mathbb{R}_+$ such that $\|x\| \leq K,\ \forall\ x \in S$.

Definition 8.3. (Polytope). A polytope is a bounded polyhedron.

Definition 8.4. (Extreme point). A point x is an extreme point of a convex set P if it cannot be written as a convex combination of two other points in P. In other words, there do not exist $y, z \in P,\ y \neq x,\ z \neq x,\ \lambda \in [0,1]$ such that $x = \lambda y + (1-\lambda)z$.

Alternatively, $x \in P$ is an extreme point if $x = \lambda y + (1-\lambda)z,\ y, z \in P,\ \lambda \in [0,1] \Longrightarrow x = y$ or $x = z$.

Remark 8.5. Extreme points are always on the boundary of a convex set, but not every point on the boundary is extreme.

Definition 8.6. (Tight or active or binding constraint). Consider a set of constraints

$$a_i^T x \geq b_i,\ i \in M_1,$$

$$a_i^T x \leq b_i,\ i \in M_2,$$

$$a_i^T x = b_i,\ i \in M_3.$$

Given a point $\overline{x}$, we say that a constraint i is tight (or active or binding) at $\overline{x}$ if $a_i^T \overline{x} = b_i$. Equality constraints are tight by definition.

Definition 8.7. (Linearly independent constraints). Two constraints are linearly independent if the corresponding a_i's are independent.

Definition 8.8. (Vertex). A point $x \in \mathbb{R}^n$ is a vertex of a polyhedron P if:

- It is feasible ($x \in P$).

- There exist n linearly independent constraints that are tight at x.

Remark 8.9. Note that the notion of an extreme point is defined geometrically, while the notion of a vertex is defined algebraically. The algebraic definition is more useful for algorithmic purposes and is crucial to the simplex algorithm. However, the geometric definition is used to prove the fundamental fact that an optimal solution to an LP can always be found at a vertex, which is crucial to correctness of the simplex algorithm.

Theorem 8.10. *Let $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ be a non-empty polyhedron with $A \in \mathbb{R}^{m \times n}$. Let $\overline{x} \in P$. Then,*

$$\overline{x} \text{ is an extreme point} \iff \overline{x} \text{ is a vertex.}$$

Proof. *** First, we prove that $\overline{x}$ is a vertex implies $\overline{x}$ is an extreme point. Let $\overline{x} \in P$ be a vertex. This implies that n linearly independent constraints are tight at $\overline{x}$. Let $\tilde{A} \in \mathbb{R}^{n \times n}$ with rows that are the rows of A associated with the tight constraints. Similarly, let $\tilde{b} \in \mathbb{R}^n$ be a vector with entries of b corresponding to the tight constraints. Thus, $\tilde{A}\overline{x} = \tilde{b}$. Suppose that we can write $\overline{x} = \lambda y + (1 - \lambda)z$ for some $y, z \in P$ and $\lambda \in [0, 1],\ y \neq \overline{x},\ z \neq \overline{x}$.

Next, we prove that $\overline{x}$ is an extreme point implies $\overline{x}$ is a vertex. Suppose that $\overline{x} \in P$ is not a vertex. Let $I = \{i \mid a_i^T \overline{x} = b_i\}$. Since $\overline{x}$ is not a vertex, there do not exist n linearly independent vectors $a_i, i \in I$. □

Corollary 8.11. *Given a finite set of linear inequalities, there can only be a finite number of extreme points.*

Proof. *** □

Definition 8.12. (Polyhedron containing a line). A polyhedron contains a line if there exist $x \in P$ and $d \in \mathbb{R}^n,\ d \neq 0$, such that

$$x + \lambda d \in P,\ \forall\ \lambda \in \mathbb{R}.$$

Theorem 8.13. *Consider a nonempty polyhedron P. The following statements are equivalent:*

- *P does not contain a line.*
- *P has at least one extreme point.*

Corollary 8.14. *Every bounded polyhedron (i.e., every polytope) has an extreme point.*

Theorem 8.15. *Consider the LP*

$$\begin{aligned} &min \quad c^T x \\ &s.t. \quad Ax \geq b\ (P). \end{aligned}$$

Suppose that P has at least one extreme point. Then, if there exists an optimal solution, there also exists an optimal solution that is at a vertex.

Proof. *** □

Remark 8.16. These theorems demonstrate that it is enough to examine only the extreme point (or vertices) when looking for an optimal solution. Thus, an algorithm for solving an LP is as follows. If there are m constraints in $\mathbb{R}^n$ (the polytope is $\{Ax \leq b\}$), then pick all possible subsets of n linearly independent constraints out of the m. Solve (in the worst case) $\binom{m}{n}$ systems of equations of the type $\tilde{A}x = \tilde{b}$, where $\tilde{A}, \tilde{b}$ are the restrictions of A and b to the subset of n constraints. (This can be done, for instance, using the conjugate gradient method from the previous lecture or by Gaussian elimination.) Check the feasibility of the solution, evaluate the objective function at each solution, and pick the best one. However, this algorithm, though correct, is quite inefficient as $\binom{m}{n}$ is exponential in n. For example, consider the constraints $\{-1 \leq x_i \leq 1\},\ i = 1, \ldots, n$. Though we only have $2n$ inequalities, we have 2^n extreme points. The simplex method is has a reduced number of vertices that we need to visit.

8.2 Simplex algorithm ***

Here's the main idea. Consider some generic LP, i.e.,

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b, \\ & x \geq 0. \end{aligned}$$

The simplex algorithm starts at a vertex and, while there is a better neighboring vertex, moves to it.

Definition 8.17. (Neighbors). Two vertices are neighbors if they share $n - 1$ tight constraints.

9 Semidefinite programming (SDP) ***

Semidefinite programming (SDP) in standard form takes the form

$$\begin{aligned} \min_{X \in S^{n\times n}} \quad & \mathrm{Tr}(CX) \\ \text{s.t.} \quad & \mathrm{Tr}(A_i X) = b_i,\ i = 1, \ldots, m, \\ & X \succeq 0, \end{aligned}$$

where $S^{n\times n}$ denotes the space of $n \times n$ real symmetric matrices. Our input data is $C \in S^{n\times n},\ A_i \in S^{n\times n}, i = 1, \ldots, m,\ b_i \in \mathbb{R}, i = 1, \ldots, m$. Thus, SDP is an optimization problem over the space of symmetric matrices. It has two types of constraints:

- Affine constraints in the entries of the decision matrix X.
- A constraint forcing some constraint to be positive semidefinite—this distinguishes SDP from LP.

Here is a summary of the importance of SDP:

- SDP is a very natural generalization of LP.
- It is still a convex optimization problem, in the geometric sense.
- We can solve SDPs efficiently (in polynomial time to arbitrary accuracy), usually using interior point methods.
- The expressive power of SDPs is much richer than of LPs.
- When it comes to nonconvex optimization, SDPs typically produce much stronger bounds and relaxations than LPs do.

Remark 9.1. Why use the trace notation? We can conveniently express affine constraints in the entries of a matrix:

$$\mathrm{Tr}(AX) = \sum_{i,j} A_{ij} X_{ij}\ (A, X \in S^{n\times n}).$$

Remark 9.2. A reminder on positive semidefinite matrices: let $X \in S^{n\times n}$, $X \succeq 0$. One feature is that $X = MM^T$ for some $n \times k$ matrix M, called a **Cholesky factorization**.

The feasible set of an SDP is called a **spectrahedron**. Note that every polyhedron is a spectrahedron (this is since every LP can be written as an SDP), but spectrahedra are richer geometric objects than polyhedra.

Example 9.3. (Elliptope). An example of a spectrahedron that is not a polyhedron.

$$\left\{ (x, y, z) \mid \begin{pmatrix} 1 & x & y \\ x & 1 & z \\ y & z & 1 \end{pmatrix} \succeq 0 \right\}.$$

Remark 9.4. Spectrahedra are always convex sets, as the set of positive semidefinite matrices is convex, affine constraints define a convex set, and the intersection of convex sets is convex. When we say that an SDP is a convex optimization problem, we mean this in the geometric sense: the objective is an affine function of the entries of the matrix and the feasible set is convex, but the feasible set is not written in the explict function form convex function ≤ 0, affine function $= 0$.

Remark 9.5. We can write an SDP as an infinite LP:

- Replace $X \succeq 0$ with linear constraints $y_i^T X y_i \geq 0 \; \forall \; y \in \mathbb{R}^n$.
- We can reduce this to be a countable infinity by only taking $y \in \mathbb{Z}^n$.

We can also write an SDP in standard functional form as a nonlinear program:

- Replace $X \succeq 0$ with $2^n - 1$ minor inequalities (Sylvester's criterion).

9.1 SDP relaxations for nonconvex optimization

9.1.1 SDP lower bounds for nonconvex polynomial minimization

10 Computational complexity theory ***

Computational complexity theory is a branch of mathematics that provides a formal framework for studying how efficiently one can solve problems on a computer. In this class, we will focus on studying decision problems, as decision problems are actually equivalent to optimization problems and search problems, and it is a bit cleaner to develop the theory for decision problems. The answer to a decision problem is simply YES or NO.

Note that we can differentiate between a problem—a (decision) problem is a general description of a problem to be answered with YES or NO—and a **problem instance**, which is defined by a finite input that needs to be specified for us to choose an answer. A decision problem has an infinite number of instances.

Remark 10.1. In general, we can think of input size as the total number of bits required to represent the input.

Definition 10.2. (Running time). Let $f, g : \mathbb{R}_+ \to \mathbb{R}_+$. We write:

- $f(n) = \mathcal{O}(g(n))$ if $\exists\ n_0, c > 0$ such that $f(n) \leq cg(n)\ \forall\ n \geq n_0$.
- $f(n) = \Omega(g(n))$ if $\exists\ n_0, c > 0$ such that $f(n) \geq cg(n)\ \forall\ n \geq n_0$.
- $f(n) = \Theta(g(n))$ if we have both $f(n) = \mathcal{O}(g(n))$ and $f(n) = \Omega(g(n))$.

Definition 10.3. (Polynomial-time algorithm). An algorithm whose running time as a function of input size is $\mathcal{O}(p(n))$ for some polynomial function p. Equivalently, it has a running time of $\mathcal{O}(n^k)$ for some positive integer k. This is the worst-case running time over all inputs of size n.

Definition 10.4. (Exponential-time algorithm). An algorithm whose running time as a function of input size is $\Omega(2^{cn})$ for some positive constant c. This is, again, the worst-case running time over all inputs of size n.

Remark 10.5. Some algorithms have a running time in between polynomial-time and exponential-time, e.g., $\mathcal{O}(n^{\log n})$, but these are also perceived as slow.

Definition 10.6. (Complexity class P). The class of all decision problems that admit a polynomial-time algorithm. This includes the decision problems ADDITION, MULTIPLICATION, LINEQ, LP, MAXFLOW, MINCUT, MATRIXPOS, SHORTEST PATH, SDP, PRIMES, ZEROSUMNASH, PENONPAPER, and others. (See notes for details.)

To prove that a problem is in P, we need to develop a polynomial-time algorithm from scratch, which can be far from trivial, or show via a reduction to a problem that is already known to be in P.

Definition 10.7. (Reduction). A reduces to B means: "if we could do B, then we could do A." A reduction from a decision problem A to a decision problem B is a general recipe for taking any instance of A and explicitly producing an instance of B such that the answer to the instance of A is YES if and only if the answer to the produced instance of B is YES.

Example 10.8. MAXFLOW reduces to LP (i.e., MAXFLOW $\rightarrow$ LP). Since we know how to solve LP in polynomial time, via interior point methods, we know how to solve MAXFLOW in polynomial time, so MAXFLOW is in P. This argument relies crucially on the fact that *the reduction is polynomial in length.* (Before we even solve the LP, we need to make sure the input size is not too big—the size must be polynomial in the size of the instance of the original problem.)

Example 10.9. MINCUT is in P, since MINCUT $\rightarrow$ MIN S-T CUT $\rightarrow$ MAXFLOW $\rightarrow$ LP $\Longrightarrow$ MINCUT $\rightarrow$ LP. Why? MIN S-T CUT $\rightarrow$ MAXFLOW since strong duality of linear programming implies that the minimum S-T cut of a graph is exactly equal to the maximum flow that can be sent from S to T. MINCUT $\rightarrow$ MIN S-T CUT since we can pick any node (say A), compute MIN S-T CUT from A to every other node, compute MIN S-T CUT from every other node to A, and take the minimum over all of these $2(|V| - 1)$ numbers (this holds since A must be in one of the cuts for the optimal solution); note that the reduction is polynomial in length.

Definition 10.10. (Complexity class NP (nondeterministic polynomial time)). A decision problem belongs to the class NP if the YES answer to any instance is easily verifiable; more precisely, every YES instance has a "certificate" of correctness that can be verified in polynomial time. This includes the decision problems in P, along with MAXCUT, TSP, STABLE SET, SAT, 3SAT, PARTITION, KNAPSACK, IP, COLORING, VERTEXCOVER, 3DMATCHING, and SUDOKU, which are NP-complete (defined later).

Remark 10.11. NP does *not* mean "not polynomial!" There are many easy problems in NP, such as ADDITION and LINEQ.

Remark 10.12. Membership in NP is shown by presenting an easily checkable certificate of the YES answer.

Remark 10.13. P $\subseteq$ NP. The polynomial-time algorithm itself is a certificate.

Remark 10.14. Note that for a given decision problem, it is not at all clear that a short certificate for the YES answer also implies a short certificate for the NO answer (e.g., TSP).

Definition 10.15. (NP-hard). A decision problem is said to be NP-hard if every problem in NP reduces to it via a polynomial-time reduction. This roughly means that the problem is "harder than all problems in NP."

NP-hardness is shown by a reduction from a problem that is already known to be NP-hard. NP-hard problems may not be in NP (or may not be known to be in NP).

Definition 10.16. (NP-complete). A decision problem that is NP-hard *and* in NP. This roughly means that the problem is among "the hardest problems in NP."

Example 10.17. (The satisfiability problem (SAT)). The input is a boolean formula in conjunctive normal form (CNF). The question is: is there a 0/1 assignment to the variables that satisfies the formula? SAT is NP-hard.

Example 10.18. (3SAT). The input is a boolean formula in conjunctive normal form (CNF), where each clause has exactly three literals. The question is: is there a 0/1 assignment to the variables that satisfies the formula? SAT is NP-hard, and there is a simple reduction from SAT to 3SAT, so 3SAT is NP-hard. 3SAT is clearly in NP, so 3SAT is NP-complete.

Definition 10.19. (Reductions cont.). This time, we use the reduction for a different purpose. A reduction from a decision problem A to a decision problem B tells us that if A is known to be hard, then B must also be hard.

Example 10.20. (3SAT $\rightarrow$ STABLE SET).

Example 10.21. (STABLE SET $\rightarrow$ 0/1 IP).

Example 10.22. (STABLE SET $\rightarrow$ Feasibility of Quadratic Equations).

Example 10.23. (3SAT $\rightarrow$ POLYPOS (degree 6)).

Question 10.24. Exercise: Prove NP-hardness of the previous three examples.

Definition 10.25. (KNAPSACK).

Definition 10.26. (PARTITION).

Definition 10.27. (POLYPOS—testing polynomial positivity).

11 ***