Data-Driven Optimization

DIWEN XU, University of Washington, USA

These notes are primarily based on the notes by Steven L. Brunton.

1 Optimization Overview

1.1 Convexity in Optimization

Definition 1.1 (An Optimization Problem).

$$\begin{aligned} \min_x \quad & f(x), \\ \text{Subject to} \quad & g_i(x) \le 0, \quad i = 1, \dots, m, \\ & h_j(x) = 0, \quad j = 1, \dots, p, \end{aligned}$$

where $f(x)$ is the objective function, and $g_i(x)$ and $h_j(x)$ are inequality and equality constraint equations over decision variable x. The inequality and equality constraints determine a feasible set S of admissible values of x over which to optimize f. When the objective function f and the feasible set S are both convex, then the problem is a convex optimization problem.

Definition 1.2 (Convex Sets). A set $C \subseteq \mathbb{R}^n$ is **convex** if and only if line segment connecting any two points $x, y \in C$ lies entirely in C.

$$\alpha x + (1-\alpha)y \in C \quad \text{for } \alpha \in [0,1].$$

Proposition 1.3 (Intersection of Convex Sets). *If C_1 and C_2 are convex, then $C = C_1 \cap C_2$ is convex.*

Proof. Let $x, y \in C$. Then $x, y \in C_1$ and $x, y \in C_2$. Since C_1 is convex, $\alpha x + (1-\alpha)y \in C_1$ for $\alpha \in [0,1]$. $\alpha x + (1-\alpha)y \in C_2$ for $\alpha \in [0,1]$. Thus $\alpha x + (1-\alpha)y \in C_1 \cap C_2 = C$ for $\alpha \in [0,1]$. □

Definition 1.4 (Convex Hull). The smallest convex set containing a set C is called its **convex hull**, denoted hull(C). The convex hull of a convex set is itself. Also, hull(C) can be viewed as the intersection of every convex set containing C.

Definition 1.5 (Affine Sets and Transformations). A set A is **affine** if for any $x, y \in A$, the point

$$\alpha x + (1-\alpha)y \in A \quad \text{for all real } \alpha \in \mathbb{R}.$$

Affine sets are automatically convex. Affine transformations have the form

$$f(x) = Ax + b,$$

and convex sets remain convex under affine transformations.

Definition 1.6 (Dual Spaces and Supporting Hyperplanes). A **supporting hyperplane** to a set $C \subset \mathbb{R}^n$ at a boundary point x_0 is a hyperplane

$$\{x : y^\top (x - x_0) = 0\}$$

where $y \in \mathbb{R}^n \setminus \{0\}$, such that

$$y^\top (x - x_0) \le 0 \quad \text{for all } x \in C.$$

Author's Contact Information: Diwen Xu, rwbyaloupeep@gmail.com, University of Washington, Seattle, Washington, USA.

C lies entirely in one closed half-space determined by that hyperplane. For a bounded convex body C containing the origin, its dual/polar set C^* is

$$C^* = \left\{ y \in \mathbb{R}^n : y^\top x \le 1 \text{ for all } x \in C \right\}.$$

Definition 1.7 (Convex Functions). A function $f : \mathbb{R}^n \to \mathbb{R}$ is **convex** if and only if for all x, y and $\alpha \in [0,1]$,

$$f(\alpha x + (1-\alpha)y) \le \alpha f(x) + (1-\alpha) f(y).$$

Equivalently, every secant line between $(x, f(x))$ and $(y, f(y))$ lies above the function on the segment between x and y.

- If f and g are convex, then $f + g$ is convex.
- If f is convex, then $-f$ is concave.
- If f is convex, then $x \mapsto f(Ax + b)$ is convex.
- If f and g are convex, then $h(x) = \max\{f(x), g(x)\}$ is convex.
- Composition $h(x) = f(g(x))$ is convex if
  - f is convex and non-decreasing and g is convex;
  - f is convex and non-increasing and g is concave.

Theorem 1.8 (Norms are Convex). *A norm $\|\cdot\|$ satisfies*

(1) $\|\alpha x\| = |\alpha| \, \|x\|$ *for all scalars* $\alpha \in \mathbb{R}$ *(homogeneity),*
(2) $\|x + y\| \le \|x\| + \|y\|$ *(triangle inequality),*
(3) $\|x\| \ge 0$ *and* $\|x\| = 0$ *iff* $x = 0$ *(non-negativity).*

$$\begin{aligned} \|\alpha x + (1-\alpha)y\| &\le \|\alpha x\| + \|(1-\alpha)y\| \\ &= \alpha \|x\| + (1-\alpha)\|y\| \\ &= \alpha f(x) + (1-\alpha) f(y). \end{aligned}$$

Definition 1.9 (Graph, Epigraph, Hypograph). For f defined on a domain D,

$$\begin{aligned} \text{graph}(f) &= \{(x, f(x)) : x \in D\}, \\ \text{epi}(f) &= \{(x, y) : x \in D,\ y \ge f(x)\}, \\ \text{hyp}(f) &= \{(x, y) : x \in D,\ y \le f(x)\}. \end{aligned}$$

A function is convex iff its epigraph is a convex set. A function is concave iff its hypograph is a convex set.

Theorem 1.10 (Jensen's Inequality). *For convex f, points $x_1, \dots, x_m$ and weights $\alpha_j \ge 0$ with $\sum_{j=1}^m \alpha_j = 1$,*

$$f(\alpha_1 x_1 + \cdots + \alpha_m x_m) \le \alpha_1 f(x_1) + \cdots + \alpha_m f(x_m).$$

If $p(x) \ge 0$ and $\int_S p(x)\, dx = 1$, then

$$f\left(\int_S p(x) x \, dx \right) \le \int_S p(x) f(x) \, dx,$$

so

$$f(\mathbb{E}[x]) \le \mathbb{E}[f(x)].$$

Theorem 1.11 (Local Minima are Global Minima). *If f is convex over a convex set C, then any local minimum is also a global minimum.*

Proof. Assume x is a local minimum. Then there exists an ϵ-ball around x,

$$B_\epsilon(x) = \{z : \|z - x\| < \epsilon\},$$

such that any feasible $z \in B_\epsilon(x)$ satisfies $f(z) \geq f(x)$. Assume for contradiction that there exists a feasible y with $f(y) < f(x)$. Because C is convex, the line segment between x and y is feasible

$$(1-\alpha)x + \alpha y \in C \quad \text{for } \alpha \in [0,1].$$

Choose

$$\alpha_1 = \frac{\epsilon}{\|y-x\|_2}, \quad \alpha_2 = \frac{\epsilon}{2\|y-x\|_2},$$

and define

$$z = (1-\alpha_2)x + \alpha_2 y.$$

Then z is feasible and $z \in B_\epsilon(x)$. By convexity of f and $f(y) < f(x)$, for any $\alpha \in [0,1]$,

$$\alpha f(y) + (1-\alpha) f(x) < f(x),$$

so $f(z) < f(x)$. This contradicts local optimality of x on $B_\epsilon(x)$. □

1.2 Gradients and the Geometry of Optimization

Definition 1.12 (Gradient). The gradient of a function $f(x)$ of an n-dimensional vector $x \in \mathbb{R}^n$ is

$$\nabla f = \begin{bmatrix} \partial f/\partial x_1 \\ \vdots \\ \partial f/\partial x_n \end{bmatrix}.$$

Theorem 1.13 (First-Order Condition for Convexity). *A function $f(x)$ is convex if and only if it lies above its first-order Taylor series approximation.*

$$f(x) \geq f(x_0) + \nabla f(x_0)^T (x - x_0).$$

Theorem 1.14 (Second-Order/Hessian Condition). *A function $f(x)$ is convex if and only if $\nabla^2 f(x) \geq 0$ everywhere.*

Theorem 1.15 (Tangent Planes as Supporting Hyperplanes). *The tangent plane may be defined by its tangent vector*

$$\vec{T} = \nabla(x, f(x)) = \begin{bmatrix} 1 \\ \nabla f(x) \end{bmatrix}.$$

A normal vector that is perpendicular to tangent vector is given by

$$\vec{n} = \begin{bmatrix} \nabla f(x) \\ -1 \end{bmatrix}.$$

For a convex function, the tangent hyperplane at any point of the graph of f is a supporting hyperplane of its epigraph. Moreover, the epigraph of f is the intersection of the half-spaces defined by all such tangent hyperplanes.

2 Gradient Based Optimization

2.1 Gradient Descent

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

where γ is the step size.

- **Fixed γ:** Fixed step schemes are simple, but may require tuning for adequate convergence.
- **Decaying γ:**

$$\gamma_k = \frac{\gamma_0}{1 + \beta k}, \quad \beta > 0.$$

- **Line search:**

$$\gamma_k = \arg\min_\gamma f(x_k - \gamma \nabla f(x_k)).$$

- **Adaptive γ:**
  - **Momentum methods:** Use physical intuition to imbue the step size with momentum, so that it is less prone to getting stuck in saddle points and locally flat regions.
  - **RMSProp:** Scale the step size with a moving average of past squared gradients, regularizing the step.
  - **Adaptive moment estimation (ADAM):** Combines momentum methods and RMSProp.

Definition 2.1 (Gradient Flow). $-\nabla f(x)$ is known as a gradient flow. It is a conservative, irrotational vector field, and it is incompressible if $\nabla^2 f = 0$.

Theorem 2.2 (Convergence of Gradient Descent). *If f is convex and Lipschitz continuous with Lipschitz constant L, then gradient descent with a step size $\gamma = 1/L$ will have the following convergence.*

$$f(x_k) - f(x^*) \leq \frac{L}{2k} \|x_0 - x^*\|^2.$$

A function is called μ-strongly convex if

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\mu}{2}\|y - x\|^2$$

for all $x, y \in \mathbb{R}^n$ and $\mu > 0$. In this case,

$$f(x_k) - f(x^*) \leq \frac{L}{2}\left(1 - \frac{\mu}{L}\right)^k \|x_0 - x^*\|^2.$$

2.2 Momentum Methods

2.2.1 Heavy-Ball Momentum Method.

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

$$\begin{cases} x_{k+1} = x_k + v_{k+1}, \\ v_{k+1} = \beta v_k - \gamma \nabla f(x_k). \end{cases}$$

2.2.2 Differential Equation Interpretation of Heavy-Ball Momentum Method.

$$m\ddot{x}(t) = -\nabla f(x(t)) - \delta \dot{x},$$

where m is the mass of the particle and δ is the friction coefficient.

$$m\frac{x_{k+1} - 2x_k + x_{k-1}}{\Delta t^2} \approx -\nabla f(x_k) - \delta \frac{x_{k+1} - x_k}{\Delta t}.$$

$$\implies (m + \delta\Delta t)x_{k+1} = -\Delta t^2 \nabla f(x_k) + (m + \delta\Delta t)x_k + m(x_k - x_{k-1}),$$

$$\implies x_{k+1} = x_k - \gamma \nabla f(x_k) + \beta(x_k - x_{k-1}),$$

where

$$\gamma = \frac{\Delta t^2}{m + \delta\Delta t}, \quad \beta = \frac{m}{m + \delta\Delta t}.$$

2.2.3 Nesterov's Accelerated Gradient (NAG) Algorithm.

$$\begin{aligned} x_{k+1} &= x_k - \gamma \nabla f\big(x_k + \beta(x_k - x_{k-1})\big) + \beta(x_k - x_{k-1}), \\ &= x_k - \gamma \nabla f(x_k + \beta v_k) + \beta v_k, \end{aligned}$$

$$\begin{cases} x_{k+1} = y_{k+1} - \gamma \nabla f(y_{k+1}), \\ y_{k+1} = x_k + \beta(x_k - x_{k-1}), \end{cases}$$

where $y_{k+1} = x_k + \beta v_k$ is the look ahead vector.

Theorem 2.3 (Convergence of Nesterov's Method). *If f is convex and f is Lipschitz continuous with constant L, then*

$$f(x_k) - f(x^*) \leq \frac{2L}{(k+1)^2} \|x_0 - x^*\|^2.$$

If f is μ-strongly convex, then

$$f(x_k) - f(x^*) \leq 2\left(1 - \sqrt{\frac{\mu}{L}}\right)^k \left(f(x_0) - f(x^*)\right).$$

2.2.4 AdaGrad and RMSProp.

$$\text{AdaGrad.} \begin{cases} x_{k+1} = x_k - \dfrac{\gamma}{\sqrt{v_{k+1}} + \epsilon} \nabla f(x_k), \\ v_{k+1} = v_k + (\nabla f(x_k))^2. \end{cases}$$

$$\text{RMSProp.} \begin{cases} x_{k+1} = x_k - \dfrac{\gamma}{\sqrt{v_{k+1}} + \epsilon} \nabla f(x_k), \\ v_{k+1} = \beta v_k + (1-\beta)(\nabla f(x_k))^2. \end{cases}$$

2.3 Stochastic Gradient Descent (SGD) and Adaptive Momentum Estimation (ADAM)

Stochastic gradient descent approximates the gradient on a subset of the data, called a *batch*, instead of the entire dataset.

$$\theta_{k+1} = \theta_k - \gamma \nabla_\theta L.$$

Adaptive momentum estimation combines key features of momentum methods and RMSProp.

$$x_{k+1} = x_k - \gamma \frac{\widehat{m}_{k+1}}{\sqrt{\widehat{v}_{k+1}} + \epsilon},$$

$$m_{k+1} = \beta_1 m_k + (1-\beta_1)\nabla f(x_k),$$
$$v_{k+1} = \beta_2 v_k + (1-\beta_2)\left(\nabla f(x_k)\right)^2.$$

$$\widehat{m}_k = \frac{m_k}{1-\beta_1^{k-1}}, \quad \widehat{v}_k = \frac{v_k}{1-\beta_2^{k-1}}.$$

2.4 Newton's Method

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

Theorem 2.4 (Convergence and Scaling). *If f is twice continuously differentiable, $\nabla^2 f$ is Lipschitz continuous near the minimizer x^*, and $\nabla^2 f(x^*)$ is positive definite, then the error $\epsilon_k = x_k - x^*$ satisfies*

$$\|\epsilon_{k+1}\| \leq \frac{1}{2}\|\epsilon_k\|^2.$$

Use the Wolfe conditions for the line search, which moves in the new gradient direction d until $\nabla_d f < \tau$, where τ is a threshold that indicates when a sufficiently flat region has been found.

(1) We take steps of size Δx and size $2\Delta x$ until we find a valid point where either (i) $f_2 > f_1$ or (ii) $\nabla f_2 > 0$. If these conditions are not met, then we double the step size and repeat.
(2) Once a region containing a minimum has been identified, we zoom in to the minimum with a bisection method. Using spline interpolation if there are multiple points in the bisection search.

2.5 Quasi-Newton Methods and BFGS

2.5.1 Approaches to Approximate A.

- **SR1:** $\text{rank}(A_k - A_{k-1}) = 1$, $A_k = A_{k-1} + uv^T$, u,v $\in \mathbb{R}^n$.
- **PSB:** $\min \|A_k - A_{k-1}\|_F$.
- **DFP:** $\min \|W(A_k - A_{k-1})W\|_F$.
- **BFGS:** $\min \|W(B_k - B_{k-1})W\|_F$, $B = A^{-1}$.

2.5.2 Basic Derivations and Formulas to Update A and B.

$$A_0 = B_0 = I.$$

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

Secant Equation. $Ap = q$, $Bq = p$.

2.5.3 Rank-1 Update.

$$(A_k + uv^T)p = q,$$
$$u = \frac{1}{v^T p}(q - A_k p),$$
$$v = q - A_k p,$$
$$A_{k+1} = A_k + \frac{1}{v^T p} vv^T.$$
$$v = p - B_k q,$$
$$B_{k+1} = B_k + \frac{1}{v^T q} vv^T.$$

2.5.4 Rank-2 Update.

$$\text{Broyden's. } A_{k+1} = A_k + \frac{1}{p^T A_k q}(p - A_k q)\, p^T A_k,$$
$$B_{k+1} = B_k + \frac{1}{p^T p}(q - B_k p)p^T.$$

$$\text{BFGS. } A_{k+1} = \left(I - \frac{pq^T}{q^T p}\right) A_k \left(I - \frac{qp^T}{q^T p}\right) + \frac{pp^T}{q^T p},$$
$$B_{k+1} = B_k + \frac{qq^T}{q^T p} - \frac{B_k p (B_k p)^T}{p^T B_k p}.$$

$$\text{DFP. } A_{k+1} = A_k + \frac{pp^T}{p^T q} - \frac{A_k q q^T A_k}{q^T A_k q},$$
$$B_{k+1} = \left(I - \frac{qp^T}{q^T p}\right) B_k \left(I - \frac{pq^T}{q^T p}\right) + \frac{qq^T}{q^T p}.$$

2.5.5 Inverse Formulas for Rank-1 Matrix Updates.

Sherman Morrison formula. $(A + uv^T)^{-1} = A^{-1} - \dfrac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1} u}$.

2.5.6 Inverse Formulas for Rank-k Matrix Updates.

Woodbury matrix identities. $(A+Uv^T)^{-1} = A^{-1} - A^{-1}U\left(I + v^T A^{-1} U\right)^{-1} v^T A^{-1}$.

2.6 Gradient Vector Calculus Identities

$$\nabla(c^T x) = \nabla(x^T c) = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \end{bmatrix} = c.$$

$$\nabla(Ax) = \begin{bmatrix} \frac{\partial}{\partial x_1}(a_{1j}x_j) & \frac{\partial}{\partial x_2}(a_{1j}x_j) & \cdots \\ \frac{\partial}{\partial x_1}(a_{2j}x_j) & \frac{\partial}{\partial x_2}(a_{2j}x_j) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = A.$$

$$\nabla\left(x^T A x\right) = \left(A + A^T\right) x.$$

3 Linear Programming

3.1 Polytopes and High Dimensional Geometry

$$\begin{aligned} \min_x \; & c^T x, \\ \text{Subject to } \; & Ax \le b, \\ & x \ge 0. \end{aligned}$$

$$\begin{aligned} \min_x \; & c^T x, \\ \text{Subject to } \; & Ax + s = b, \\ & x, s \ge 0, \end{aligned}$$

where $c \in \mathbb{R}^n$ is the vector of linear weights for the objective function, and $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ define the constraint equations.

- Vertices are always connected by one-dimensional edges.
- The intersection of 2 edges is a vertex.
- In n-dimensions, a single inequality constraint defines a half-space bounded by an $n-1$-dimensional surface.
- The intersection of $n+1$ inequality constraints will bound a polytope in n-dimensions.
- In n-dimensions, the vertices of a polytope are given by the exact equality of n inequality constraints.

3.2 Dantzig's Simplex Method

Dantzig's simplex method relies on two major properties.

- A local extremum will occur on a vertex, assuming that the feasible set is a bounded polytope.
- The local extremum is a global extremum because linear programs are convex optimization problems.

The process of going from vertex to vertex is called *pivoting*. The most commonly used pivoting technique is *Dantzig's pivot rule*.

(1) Choose an active constraint to relax (loosen).
Pick the direction where the objective function increases the most, and if this variable is currently tight, then loosen it. The direction of fastest increase of the objective function is given by the gradient ∇f.
(2) Choose an inactive constraint to tighten.
Once we select a tight variable to loosen, we must choose another loose variable to tighten. This defines the next vertex that we walk to. Generally, we pick among the slack variables, finding the inequality that will be violated first when increasing the newly loosened variable.
(3) Recenter the problem at the new vertex (pivot update).
Recenter by taking the inequality that will be violated first and swapping the position of the loosened and tightened variables.

3.3 Simplex Tableau Method

(1) Tight Variable to Loosen (column): Pick the variable with the most negative coefficient (in a maximization problem) in the objective row.
(2) Loose Variable to Tighten (row): The minimum ratio test: pick the row with the smallest ratio RHS/(pivot coefficient).
(3) Row Reduction about Pivot: Perform row operations so that the pivot entry becomes 1, and all other entries in that pivot column become 0.

There are some exceptions and edge cases that require attention.

(1) Unbounded Problems. If during the minimum ratio test all pivot coefficients are less than zero, or no feasible pivot is found, the objective can increase indefinitely, indicating an unbounded linear program.
(2) Degeneracy. Sometimes multiple constraints become active at once, or the ratio test yields a tie. This can cause cycling, where we pivot among the same vertices. Solvers use anti-cycling rules, like Bland's rule, to break these ties.
(3) Phase I and Artificial Variables. We begin by adding artificial variables to each constraint that lacks a straightforward slack variable. Phase I involves solving an auxiliary LP where the objective is to minimize the sum of all artificial variables, subject to the same constraints. If the Phase I optimum is zero, a feasible solution to the original problem exists, and the artificial variables can be made inactive. If the minimum is greater than zero, then no feasible solution exists.

3.4 Duality in Linear Programming

Generally, for the primal LP

$$\begin{aligned} \max \; & c^\top x, \\ \text{subject to} \; & x \ge 0, \\ & Ax \le b, \end{aligned}$$

the dual is given by

$$\begin{aligned} \min \; & b^\top y, \\ \text{subject to} \; & y \ge 0, \\ & A^\top y \ge c. \end{aligned}$$

- **Weak duality:** For any feasible primal x and any feasible dual y, we have
$$c^\top x \le b^\top y.$$
- **Strong duality:** Under mild conditions, such as the existence of an optimal basic feasible solution in the primal, the maximum primal objective and minimum dual objective coincide at optimality.

4 Least-Squares Regression

4.1 Least-Squares Regression

$$Ax = b, \quad b = b_{\text{true}} + \epsilon.$$

$$\|Ax-b\|_2^2 = (Ax-b)^T(Ax-b) = x^TA^TAx - x^TA^Tb - b^TAx + b^Tb.$$

$$\nabla\|Ax-b\|_2^2 = 2A^TAx - A^Tb - A^Tb = 0$$
$$\implies A^TAx = A^Tb \implies x = (A^TA)^{-1}A^Tb.$$

The matrix $A^\dagger = (A^TA)^{-1}A^T$ is known as the Moore–Penrose pseudo-inverse. However, this is only valid when (A^TA) is invertible. The matrix A^TA is invertible when A has n linearly independent columns and $m \geq n$. Thus, this formula is only applicable for the over-determined case. Given the SVD of a real matrix A,

$$A = U\Sigma V^T,$$

then the Moore–Penrose pseudo-inverse may be written as

$$A^\dagger \triangleq V\Sigma^{-1}U^T \quad \implies \quad x = V\Sigma^{-1}U^Tb.$$

Least-squares regression is sensitive to large outliers in the data. To address the robustness of least squares to outliers, it may be possible to instead minimize the sum of absolute values of the errors,

$$\|Ax-b\|_1,$$

known as the least absolute deviation (LAD) regression. Least-squares regression is an estimator for the mean of a distribution, while least-absolute-deviation regression estimates the median.

The solution space to a linear system is determined by the following four fundamental subspaces of A:

- The column space, $\mathrm{col}(A)$, also known as the range.
- The orthogonal complement to $\mathrm{col}(A)$ is $\ker(A^T)$.
- The row space, $\mathrm{row}(A)$. $\mathrm{row}(A) = \mathrm{col}(A^T)$.
- The kernel space, $\ker(A)$, is the orthogonal complement to $\mathrm{row}(A)$, and is also known as the null space.

If $b \in \mathrm{col}(A)$ and if $\dim(\ker(A)) \neq 0$, then there are infinitely many solutions x. If $b \notin \mathrm{col}(A)$, then there are no solutions, and the system of equations is called inconsistent.

$$\mathrm{col}(A) \oplus \ker(A^T) = \mathbb{R}^n,$$
$$\mathrm{col}(A^T) \oplus \ker(A) = \mathbb{R}^n.$$

In the over-determined case when no solution exists, we would often like to find the solution x that minimizes the sum-squared error $\|Ax-b\|_2^2$, the so-called least-squares solution. In the under-determined case when infinitely many solutions exist, we may like to find the solution x with minimum norm $\|x\|_2$ so that $Ax = b$, the so-called minimum-norm solution.

4.2 Singular Value Decomposition

$$A = U\Sigma V^T.$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are unitary matrices with orthonormal columns, and $\Sigma \in \mathbb{R}^{m\times n}$ is a matrix with real, non-negative entries on the diagonal and zeros off the diagonal. The columns of U are called left singular vectors of A and the columns of V are right singular vectors. The diagonal elements of Σ are called singular values and they are ordered from largest to smallest

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0.$$

The above can easily be generalized for complex valued matrices $A \in \mathbb{C}^{m\times n}$, in which case we use the complex conjugate transpose * instead of the real transpose ^{T}. When $m \geq n$, the matrix Σ has at most n non-zero elements on the diagonal. If there are linearly dependent columns of A, then there will be even less non-zero singular values.

$$A = U\Sigma V^T = \begin{bmatrix}\widehat{U} & \widehat{U}_\perp\end{bmatrix}\begin{bmatrix}\widehat{\Sigma}\\ 0\end{bmatrix}V^T = \widehat{U}\,\widehat{\Sigma}\,V^T.$$

The rank of A is equal to the number of non-zero singular values. The SVD can also be used to obtain an optimal rank-r approximation of $A = \widetilde{U}\,\widetilde{\Sigma}\,\widetilde{V}^T$, where the rank $r < n$ is chosen to retain the largest singular values. In this case, the matrices $\widetilde{U}$ and $\widetilde{V}$ contain the first r columns of U and V and $\widetilde{\Sigma}$ contains the first $r \times r$ block of Σ.

- The column space of A is the same as the column space of $\widehat{U}$.
- The orthogonal complement to the column space of A is given by the column space of $\widehat{U}_\perp$.
- The row space of A is spanned by the columns of $\widehat{V}$.
- The kernel space of A is given by $\mathrm{col}(\widehat{V}_\perp)$.

$$A^\dagger \triangleq \widehat{V}\,\widehat{\Sigma}^{-1}\widehat{U}^T \quad \implies \quad A^\dagger A = \widehat{V}\,\widehat{V}^T.$$
$$A^\dagger Ax^* = A^\dagger b \quad \implies \quad x^* = \widehat{V}\,\widehat{\Sigma}^{-1}\widehat{U}^Tb.$$
$$Ax^* = \widehat{U}\,\widehat{\Sigma}\,\widehat{V}^T\,\widehat{V}\,\widehat{\Sigma}^{-1}\widehat{U}^Tb = \widehat{U}\,\widehat{U}^Tb.$$

Theorem 4.1 (Eckart–Young). *Optimal rank-r approximation to X, in a least-squares sense, is given by the rank-r SVD truncation $\widetilde{X}$.*

$$\arg\min_{\mathrm{rank}(\widetilde{X})=r} \|X-\widetilde{X}\|_F = \widetilde{U}\,\widetilde{\Sigma}\,\widetilde{V}^T.$$

$$X - \widetilde{X} = \sum_{k=r+1}^{m}\sigma_k u_k v_k^T, \quad \|X-\widetilde{X}\|_F^2 = \sum_{k=r+1}^{m}\sigma_k^2, \quad \|X-\widetilde{X}\|_2 = \sigma_{r+1}.$$

4.3 Minimum Norm Regression

$$\min_x \quad \|x\|_2^2$$
$$\text{s.t.} \quad Ax = b.$$

$$L(x,\lambda) = x^Tx + \lambda^T(Ax-b).$$

$$\nabla_x L = 2x + A^T\lambda = 0 \quad \implies \quad Ax = -\frac{1}{2}AA^T\lambda,$$

$$\nabla_\lambda L = Ax - b = 0 \quad \implies \quad \lambda = -2\,(AA^T)^{-1}b,$$

$$x = -\frac{1}{2}A^T\lambda = A^T(AA^T)^{-1}b.$$

This is the Moore–Penrose right pseudo-inverse.

4.4 Condition Number and Regularization

4.4.1 *Condition Number.*

$$\kappa(A) = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}.$$

$$A(x+\epsilon_x) = b + \epsilon_b, \quad \frac{\|\epsilon_x\|}{\|x\|} = \frac{\sigma_{\max}}{\sigma_{\min}}\frac{\|\epsilon_b\|}{\|b\|}.$$

4.4.2 Ridge Regression. Tikhonov regularization, also known as ridge regression, addresses the poor condition number of the least-squares regression when the columns of A are very nearly parallel and the condition number of the pseudo-inverse is large. Any noise on b will be amplified by this ill-conditioned pseudo-inverse. This has the effect of some entries of x being large and opposite, leading to high variance in the resulting solution. The solution is to regularize the least-squares problem with an additional loss term $\|x\|_2^2$.

$$x = \arg\min_{x'} \ \|Ax' - b\|_2^2 + \alpha\|x'\|_2^2,$$

where α is a hyper-parameter that weights this regularizing term.

$$x = \left(A^T A + \alpha I\right)^{-1} A^T b.$$

$$(A^T A + \alpha I)^{-1} = (V\Sigma^2 V^T + \alpha I)^{-1} = \left(V(\Sigma^2 + \alpha I)V^T\right)^{-1}.$$

This leads to a fundamental property in many regularized optimization problems: the bias-variance tradeoff.

4.4.3 Sparse Regression. The least absolute shrinkage and selection operator (LASSO) is an ℓ_1 penalized regression technique that balances model complexity with descriptive capability. Least-squares regression will tend to result in a vector x that has nonzero coefficients for all entries, indicating that all columns of A must be used to predict b. However, we often believe that the statistical model should be simpler, indicating that x may be sparse.

$$x = \arg\min_{x'} \ \|Ax' - b\|_2^2 + \lambda\|x'\|_1.$$

A regression model is more interpretable if it has fewer terms that bear on the outcome, motivating yet another perspective on sparsity. The elastic net combines the ridge and LASSO regularization terms

$$x = \arg\min_{x'} \ \|Ax' - b\|_2^2 + \lambda\|x\|_1 + \alpha\|x\|_2^2.$$

4.5 Quadratic Programming

$$\min_x \ \frac{1}{2}x^T Q x + c^T x$$
$$\text{Subject to } \ Ax \le b$$

where Q is a symmetric matrix. If Q is positive semi-definite, then this quadratic program is convex and any local minimum is a global minimum. For quadratic programming with active constraints the optimum value can occur on an edge, a face, or a vertex. There are several approaches to solve quadratic programs.

- **Active set methods:** Guess which constraints are active and then solve a smaller problem with only active constraints.
- **Interior point methods (IPM):** Scale well to large systems when we have active linear constraints like $Ax \le b$.
- **Sequential quadratic programming (SQP):** Good for medium-scale systems with possibly nonlinear constraints.

4.6 Nonlinear Least-Squares: Gauss–Newton Methods

$$L = \frac{1}{2}\sum_{j=1}^{m} \left\|y_j - N(x_j;\theta)\right\|_2^2 = \frac{1}{2}\sum_{j=1}^{m} f_j(\theta)^2 = \frac{1}{2}\left\|f(\theta)\right\|_2^2.$$

$$\nabla_\theta L = \sum_{j=1}^{m} f_j(\theta)\nabla f_j(\theta) = J^T f, \quad J = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} \nabla f_1^T \\ \nabla f_2^T \\ \vdots \end{bmatrix}.$$

$$H = \nabla_\theta^2 L = J^T J + \sum_{j=1}^{m} f_j(\theta)\,\nabla^2 f_j(\theta).$$

Because $f_j(\theta)$ are the residual errors, assume that these are small, and approximate the Hessian just using the first-order terms.

$$H \approx J^T J.$$

Therefore, the full Gauss–Newton iteration is

$$\theta_{k+1} = \theta_k + s_k,$$
$$J_k^T J_k\, s_k = -J_k^T f(\theta_k).$$

Gauss–Newton methods are efficient for well-conditioned problems, but have known failure modes when the Hessian approximation is ill-conditioned. The Levenberg–Marquardt (LM) algorithm combines Gauss–Newton with gradient descent to improve robustness for poorly conditioned or ill-posed problems. LM also helps when the initial guess is far from the solution and residuals are large, making the Hessian approximation in Gauss–Newton inaccurate.

$$\left(J^T J + \alpha I\right)\Delta\theta = -J^T f.$$

The damping parameter α blends between Gauss–Newton ($\alpha = 0$) and gradient descent ($\alpha \to \infty$). When the initial guess is far from the solution, a large α is used, and the gradient-descent behavior stabilizes the updates. As the iteration approaches the optimal solution, LM automatically shifts to the faster Gauss–Newton method.

$$\rho = \frac{\Delta L_{\text{actual}}}{\Delta L_{\text{predicted}}}.$$
$$\Delta L_{\text{actual}} = L(\theta_k) - L(\theta_{k+1}),$$
$$\Delta L_{\text{predicted}} = -(J^T f)^T\,\Delta\theta - \frac{1}{2}\Delta\theta^T\left(J^T J + \alpha I\right)\Delta\theta.$$

The parameter α is initialized with a small positive value (e.g., 10^{-3}).

- If $\rho > 0.75$, reduce α (the model is trustworthy).
- If $\rho < 0.25$, increase α (the model is unreliable).

$$\alpha = \max(\alpha\nu,\ \epsilon\|\Delta\theta\|),$$

where $\epsilon > 0$ ensures sufficient regularization.

4.7 The Conjugate Gradient Method

The conjugate gradient (CG) method is an iterative algorithm for solving large, sparse systems of linear equations

$$Ax = b,$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite (SPD), $x \in \mathbb{R}^n$ is the unknown vector, and $b \in \mathbb{R}^n$ is the right-hand side.

$$p_i^T A p_j = 0 \quad \forall\, i \ne j, \quad x = \sum_j \alpha_j p_j.$$

$$Ax = \sum_j \alpha_j A p_j = b \ \Rightarrow\ \sum_j \alpha_j p_k^T A p_j = \alpha_k\, p_k^T A p_k = p_k^T b.$$

$$x_{k+1} = x_k + \alpha_k p_k = x_k + \frac{p_k^T b}{p_k^T A p_k}\, p_k.$$

$$p_0 = r_0 = b - Ax_0.$$

$$r_k = b - Ax_k, \quad p_k = r_k - \sum_{j=0}^{k-1} \frac{p_j^T A r_k}{p_j^T A p_j} p_j.$$

At each iteration, CG works within the k-dim Krylov subspace

$$\mathcal{K}_k = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0\},$$

where $r_0 = b - Ax_0$ is the initial residual.
A typical convergence estimate is

$$\|x_k - x^*\|_A \leq C \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k.$$

For poorly conditioned systems (large $\kappa(A)$), convergence slows, and preconditioning is commonly used

$$M^{-1}Ax = M^{-1}b,$$

where M approximates A while being easy to invert.
The Generalized Minimal Residual method (GMRES) applies to any square matrix and directly minimizes the two-norm of the residual $\|r\|_2$ by building an orthonormal basis for the Krylov space. GMRES typically has higher cost per iteration and its storage requirements grow with each iteration, but it is often more robust and can be a good alternative to direct methods when they are too expensive.