DSA5103 Optimization Problem for Data Modelling

AY2024/25 Sem2 By Zhao Peiduo

Lecture 1

Nonlinear Programming A general **nonlinear programming problem (NLP)** is to minimize/maximize a function $f(x)$, subject to equality constraints $g_i(x) = 0$, $i \in [m]$, and inequality constraints $h_j(x) \le 0$, $j \in [p]$. Here, f, g_i, and h_j are functions of the variable $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$. The term definitions are as follows:

- f: **Objective function**
- $g_i(x) = 0$: **Equality constraints**
- $h_j(x) \le 0$: **Inequality constraints**

It suffices to discuss minimization problems since minimizing $f(x)$ is equivalent to maximizing $-f(x)$.

Feasible Set
$$S = \{x \in \mathbb{R}^n \mid g_1(x) = 0, \ldots, g_m(x) = 0,\ h_1(x) \le 0, \ldots, h_p(x) \le 0\}.$$

A point in the feasible set is a **feasible solution** or **feasible point** wbere all constraints are satisfied; otherwise, it is an **infeasible solution** or **infeasible point**. When there is no constraint, $S = \mathbb{R}^n$, we say the NLP is **unconstrained**.

Local and Global Minimizer Let S be the feasible set. Define $B_\epsilon(y) = \{x \in \mathbb{R}^n \mid \|x - y\| < \epsilon\}$ to be the open ball with center y and radius ϵ. Here, $\|x\| = \sqrt{x_1^2 + \cdots + x_n^2}$.

1. A point $x^* \in S$ is said to be a **local minimizer** of f if there exists $\epsilon > 0$ such that
$$f(x^*) \le f(x) \quad \forall x \in S \cap B_\epsilon(x^*).$$
2. A point $x^* \in S$ is said to be a **global minimizer** of f if
$$f(x^*) \le f(x) \quad \forall x \in S.$$

Interior point Let $S \subseteq \mathbb{R}^n$ be a nonempty set. An point $x \in S$ is called an **interior point** of S if
$$\exists \epsilon > 0 \quad s.t. \quad B_\epsilon(x) \subseteq S.$$

Gradient Vector Let $S \subseteq \mathbb{R}^n$ be a nonempty set. Suppose $f : S \to \mathbb{R}$, and x is an interior point of S such that f is differentiable at x. Then the **gradient vector** of f at x is
$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}.$$

Hessian Matrix Let $S \subseteq \mathbb{R}^n$ be a nonempty set. Suppose $f : S \to \mathbb{R}$, and x is an interior point of S such that f has second-order partial derivatives at x. Then the **Hessian** of f at x is the $n \times n$ matrix:
$$H_f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(x) & \frac{\partial^2 f}{\partial x_n \partial x_2}(x) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{bmatrix}.$$

- The ij-entry of $H_f(x)$ is $\frac{\partial^2 f}{\partial x_i \partial x_j}(x)$.
- In general, $H_f(x)$ is not symmetric. However, if f has continuous second-order derivatives, then the Hessian matrix is symmetric since ∂x_i and ∂x_j are interchangeable.

Positive (Semi)Definite Let A be a real $n \times n$ matrix.

1. A is said to be **positive semidefinite** if $x^T Ax \ge 0$, $\forall x \in \mathbb{R}^n$.
2. A is said to be **positive definite** if $x^T Ax > 0$, $\forall x \ne 0$.
3. A is said to be **negative semidefinite** if $-A$ is positive (semi)definite.
4. A is said to be **negative definite** if $-A$ is positive definite.
5. A is said to be **indefinite** if A is neither positive nor negative semidefinite.

Eigenvalue Test Theorem Let A be a real symmetric $n \times n$ matrix.

1. A is **positive semidefinite** iff every eigenvalue of A is nonnegative.
2. A is **positive definite** iff every eigenvalue of A is positive.
3. A is **negative semidefinite** iff every eigenvalue of A is nonpositive.
4. A is **negative definite** iff every eigenvalue of A is negative.
5. A is **indefinite** iff it has both a positive eigenvalue and a negative eigenvalue.

Proof for: A is positive semidefinite iff every eigenvalue of A is nonnegative
(Forward) Suppose A is positive semidefinite, show that its eigenvalues are nonnegative. By definition, a Hermitian matrix A is positive semidefinite if for all nonzero vectors $x \in \mathbb{C}^n$:
$$x^* Ax \ge 0$$

Let λ be an eigenvalue of A with corresponding eigenvector x such tha $Ax = \lambda x$. Taking the inner product of both sides with x, we obtain:
$$x^* Av = v^*(\lambda x) = \lambda(x^* x)$$

Since $v^* v$ (the squared norm of v) is always positive for nonzero v, the above equation implies $\lambda \ge 0$

(Backward) Since A is Hermitian, it has an orthonormal basis of eigenvectors $\{q_1, q_2, \ldots, q_n\}$ with corresponding real eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$.
For any vector x, we can express it in terms of the eigenvectors as:
$$x = \sum_{i=1}^n c_i q_i$$
for some scalars c_i, and compute the quadratic form:
$$x^* Ax = \left(\sum_{i=1}^n c_i^* q_i^*\right) A \left(\sum_{j=1}^n c_j q_j\right)$$
Expanding the expression using the orthonormality of the eigenvectors:
$$x^* Ax = \sum_{i=1}^n \lambda_i |c_i|^2$$
Since we are given that all eigenvalues $\lambda_i \ge 0$, and the squared magnitudes $|c_i|^2$ are nonnegative, it follows that:
$$x^* Ax \ge 0 \quad \forall x \ne 0$$
Thus, A is positive semidefinite.

Necessary and Sufficient Conditions
Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is nonlinear and differentiable. A point x^* is called a **stationary point** of f if $\nabla f(x^*) = 0$.
Necessary condition: Confine our search for global minimizers within the set of stationary points
If x^* is a local minimizer of f, then

1. x^* is a stationary point, i.e., $\nabla f(x^*) = 0$
2. The Hessian $H_f(x^*)$ is positive semidefinite

Sufficient condition: Verify that a point is indeed a local minimizer
If the following conditions hold, then x^* is a local minimizer of f.

1. x^* is a stationary point, i.e., $\nabla f(x^*) = 0$
2. The Hessian $H_f(x^*)$ is positive definite,

Convex set A set $D \in \mathbb{R}^n$ is said to be a **convex** set if for any two points x and y in D, the line segment joining x and y also lies in D. That is,
$$x, y \in D \Rightarrow \lambda x + (1 - \lambda)y \in D \quad \forall \lambda \in [0, 1].$$

Strictly convex function
Let $D \subseteq \mathbb{R}^n$ be a convex set. Consider a function $f : D \to \mathbb{R}$.

1. The function f is said to be **convex** if $f(\lambda x + (1 - \lambda)y) \le \lambda f(x) + (1 - \lambda)f(y), \quad \forall x, y \in D,\ \lambda \in [0, 1]$.
2. The function f is said to be **strictly convex** if $f(\lambda x + (1 - \lambda)y) < \lambda f(x) + (1 - \lambda)f(y)$. for all distinct $x, y \in D$, $\lambda \in (0, 1)$.

For a convex f It holds that

1. any local minimizer is a global minimizer.
2. if f is strictly convex, then the global minimizer is unique.

Test for convexity of a differentiable function
Suppose that f has continuous second partial derivatives on an open convex set D in $\mathbb{R}^n$.

1. The function f is convex on D **iff** the Hessian matrix $H_f(x)$ is positive semidefinite at each $x \in D$.
2. If $H_f(x)$ is positive definite at each $x \in D$, then f is strictly convex on D.
3. If $H_f(\hat{x})$ is indefinite at some point $\hat{x} \in D$, then f is not a convex nor a concave function on D.

Eigenvalue Decomposition: The eigenvalue decomposition of $A \in \mathbb{S}^n$ is given by:
$$A = Q\Lambda Q^T = [Q_{\cdot 1} \quad \cdots \quad Q_{\cdot n}] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} [Q_{\cdot 1} \quad \cdots \quad Q_{\cdot n}]^T$$
where Q is an orthogonal matrix whose **columns** are eigenvectors of A, Λ is a diagonal matrix with eigenvalues of A on the diagonal.
Change of bases using eigenvectors Denote the ith column of orthogonal matrix Q as q_i. Change the bases to $\{q_1, q_2\}$. With new bases,

- For any vector x, $x = Q(Q^T x)$, so its representation becomes
$$\tilde{x} = Q^T x = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix}$$
- Since $y = Ax = Q\Sigma Q^T x$, the representation of y is
$$\tilde{y} = \Sigma \tilde{x} = \begin{bmatrix} \lambda_1 \tilde{x}_1 \\ \lambda_2 \tilde{x}_2 \end{bmatrix}$$

Hence, the linear transformation results in a scaling of λ along the eigenvector associated with λ.

Statistical Properties Let $x_1, \dots, x_n \in \mathbb{R}^p$ be n observations of a random variable x.

- Mean vector: $\mu = \bar{x} = \frac{1}{n}\sum_{i=1}^n x_i \in \mathbb{R}^p$
- (Sample/Empirical) Covariance matrix: $\Sigma = \frac{1}{n-1}\sum_{i=1}^n (x_i - \mu)(x_i - \mu)^T \in \mathbb{R}^{p \times p}$ (Covariance matrices are symmetric and positive semidefinite)
- Standard deviation (for $p = 1$): $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^n (x_i - \mu)^2}$

PCA

- PCA is often used to **reduce the dimensionality** of large data sets while preserving as much information as possible.
- PCA allows us to identify the **principal directions in which the data varies**.

Let $x_1, \dots, x_n \in \mathbb{R}^p$ be n observations of a random variable x and

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}.$$

The mean vectors of x_i and $Q^T x_i$ (for $i = 1, \dots, n$) are, respectively,

$$\mu = \frac{1}{n}\sum_{i=1}^n x_i, \quad \hat{\mu} = \frac{1}{n}\sum_{i=1}^n Q^T x_i = Q^T \mu.$$

Consequently, the associated covariance matrices are, respectively,

$$\Sigma = \frac{1}{n-1}\sum_{i=1}^n (x_i - \mu)(x_i - \mu)^T,$$

$$\hat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^n (Q^T x_i - Q^T \mu)(Q^T x_i - Q^T \mu)^T = Q^T \Sigma Q.$$

Optimization problem of PCA

$$\max_{Q \in \mathbb{R}^{p \times k},\, Q^T Q = I} \operatorname{trace}(Q^T \Sigma Q).$$

Let the eigenvalue decomposition of Σ be

$$\Sigma = \begin{bmatrix} q_1 & \cdots & q_p \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_p \end{bmatrix} \begin{bmatrix} q_1 & \cdots & q_p \end{bmatrix}^T,$$

where

$$\lambda_1 \geq \cdots \geq \lambda_p \geq 0.$$

Then

$$Q = \begin{bmatrix} q_1 & \cdots & q_k \end{bmatrix}.$$

Standard PCA workflow

1. Make sure the data X are rows = observations and columns = variables.
2. Standardize the columns of X.
3. Run $[Q, X_{\text{new}}, d, \text{tsquared}, \text{explained}] = \text{pca}(X)$.
4. Using the variance% in "explained", choose k (usually 1, 2, or 3) components for visual analysis.
    - For example, if $d = (1.9087, 0.0913)$, explained$= (95.4, 4.6)$, one may choose $k = 1$ as the first principal component carries 95.4% of the information.
    - For example, if $d = (2.9108, 0.9212, 0.1474, 0.0206)$, explained$= (72.8, 23.0, 3.7, 0.5)$, one may choose $k = 2$ as the first two principal components carry 95.8% of the information.
5. Plot $X_{\text{new}}(:, 1), \dots, X_{\text{new}}(:, k)$ on a k-dimensional plot.

Lecture 2

Gradient Descent Method Given $x_0 \in \mathbb{R}^n$, for $k = 0, 1, 2, \dots$ do:

$$r_k = A x_k - b,$$
$$\alpha_k = \frac{(r_k, r_k)}{(A r_k, r_k)},$$
$$x_{k+1} = x_k - \alpha_k r_k.$$

Gradient Descent Method Example: Ax = b where A is Symmetric Positive Definite
Let

$$f(x) = \|x - x_\star\|_A^2 = (A(x - x_\star), (x - x_\star)) = (x - x_\star)^T A (x - x_\star),$$

where $x_\star$ is the solution of

$$Ax = b.$$

It is obvious that

$$f(x) = 0 \quad \text{if and only if} \quad x = x_\star.$$

Denote

$$x = x_0 + \delta_0.$$

Then,

$$\begin{aligned} f(x) &= f(x_0) + (A\delta_0, \delta_0) + 2\delta_0^T (A x_0 - b) \\ &= f(x_0) + \delta_0^T A \delta_0 + 2\delta_0^T r_0, \end{aligned}$$

where

$$r_0 = A x_0 - b.$$

It is clear that

$$f(x) \leq f(x_0)$$

only if

$$\delta_0^T r_0 \leq 0,$$

in particular,

$$-r_0 = b - A x_0$$

is the negative of the gradient direction $-\nabla f$ at the point x_0.
The negative of the gradient direction is locally the direction that yields the fastest rate of decrease for f. Hence, we can choose

$$\delta_0 = -\alpha_0 r_0,$$

so that

$$\begin{aligned} f(x) &= f(x_0) + \alpha_0^2 (A r_0, r_0) - 2\alpha_0 r_0^T r_0 \\ &= f(x_0) + \alpha_0^2 r_0^T A r_0 - 2\alpha_0 r_0^T r_0 \leq f(x_0), \end{aligned}$$

provided

$$\alpha_0 \geq 0.$$

It is obvious, we have

$$f(x) \leq f(x_0), \quad \forall 0 \leq \alpha \leq \frac{2(r_0, r_0)}{(A r_0, r_0)}.$$

The optimal α shall satisfy

$$f(x) = \min_{\alpha_0 \in \mathbb{R}} f(x_0) + \alpha_0^2 (A r_0, r_0) - 2\alpha_0 r_0^T r_0,$$

i.e.,

$$\alpha_0 = \frac{(r_0, r_0)}{(A r_0, r_0)} \geq 0.$$

Therefore, we conclude

$$\text{If} \quad x = x_0 - \alpha_0 r_0, \quad \text{then} \quad f(x) \leq f(x_0).$$

Kantorovich Inequality Let B be any Symmetric Positive Definite real matrix and $\lambda_{\max}$ and $\lambda_{\min}$ its largest and smallest eigenvalues. Then,

$$\frac{(Bx, x)(B^{-1}x, x)}{(x, x)^2} \leq \frac{(\lambda_{\max} + \lambda_{\min})^2}{4\lambda_{\max}\lambda_{\min}}, \quad \forall x \neq 0.$$

Kantorovich Inequality Proof
Clearly, it is equivalent to show that the result is true for any unit vector x. Since B is symmetric, we have

$$B = Q^T D Q,$$

where Q is orthogonal and

$$D = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix},$$

and

$$\lambda_{\max} = \lambda_1 \geq \cdots \geq \lambda_n = \lambda_{\min} > 0.$$

We have

$$(Bx, x)(B^{-1}x, x) = (DQx, Qx)(D^{-1}Qx, Qx).$$

Setting

$$y = Qx = [y_1 \quad \cdots \quad y_n]^T, \quad \beta_i = y_i^2.$$

Note that $\sum_{i=1}^n \beta_i = 1$, and

$$\lambda = (Dy, y) = \sum_{i=1}^n \beta_i \lambda_i$$

is a convex combination of the eigenvalues λ_i, $i = 1, \cdots, n$, and furthermore, the following relation holds,

$$(Bx, x)(B^{-1}x, x) = \lambda \psi(y),$$

with

$$\psi(y) = (D^{-1}y, y) = \sum_{i=1}^n \beta_i \frac{1}{\lambda_i}.$$

Noting that

$$\psi(y) \le \frac{1}{\lambda_1} + \frac{1}{\lambda_n} - \frac{\lambda}{\lambda_1 \lambda_n}, \quad (\text{since } \sum_{i=1}^n \beta_i = 1, \text{proved later})$$

therefore,

$$(Bx, x)(B^{-1}x, x) = \lambda \psi(y) \le \lambda \left(\frac{1}{\lambda_1} + \frac{1}{\lambda_n} - \frac{\lambda}{\lambda_1 \lambda_n} \right).$$

The maximum of the right-hand side is reached for

$$\lambda = \frac{\lambda_1 + \lambda_n}{2}$$

yielding

$$(Bx, x)(B^{-1}x, x) = \lambda \psi(y) \le \lambda \left(\frac{1}{\lambda_1} + \frac{1}{\lambda_n} - \frac{\lambda}{\lambda_1 \lambda_n} \right)$$

$$\le \frac{\lambda_1 + \lambda_n}{4} \left(\frac{1}{\lambda_1} + \frac{1}{\lambda_n} \right)$$

Proof for $\psi(y) \le \frac{1}{\lambda_1} + \frac{1}{\lambda_n} - \frac{\lambda}{\lambda_1 \lambda_n}$
Since

$$0 < \lambda_n \le \cdots \le \lambda_i \le \cdots \le \lambda_1, \quad i = 1, \ldots, n,$$

we have for any $i = 1, \ldots, n$ that

$$\lambda_1 \ge \lambda_i > 0, \quad \lambda_i - \lambda_n \ge 0, \quad i = 1, \ldots, n,$$

which gives

$$\lambda_1(\lambda_i - \lambda_n) \ge \lambda_i(\lambda_1 - \lambda_n),$$

i.e.,

$$\lambda_1 \lambda_n \le \lambda_i(\lambda_1 + \lambda_n - \lambda_i),$$

and

$$\frac{1}{\lambda_i} \le \frac{\lambda_1 + \lambda_n - \lambda_i}{\lambda_1 \lambda_n}.$$

Note that

$$\beta_i \ge 0, \quad \sum_{i=1}^n \beta_i = 1,$$

we get

$$\beta_i \frac{1}{\lambda_i} \le \beta_i \frac{\lambda_1 + \lambda_n - \lambda_i}{\lambda_1 \lambda_n},$$

and so,

$$\sum_{i=1}^n \beta_i \frac{1}{\lambda_i} \le \sum_{i=1}^n \beta_i \frac{\lambda_1 + \lambda_n - \lambda_i}{\lambda_1 \lambda_n}$$

$$= \frac{1}{\lambda_1} + \frac{1}{\lambda_n} - \frac{\sum_{i=1}^n \beta_i \lambda_i}{\lambda_1 \lambda_n}.$$

This lemma helps to establish the following result regarding the convergence rate of the method.
Theorem Let A be a Symmetric Positive Definite matrix. Then, the A-norms of the error vectors

$$d_k = x_\star - x_k = -A^{-1} r_k$$

generated by the Gradient Descent Algorithm satisfy the relation

$$\|d_{k+1}\|_A \le \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} \|d_k\|_A,$$

and so,

$$\lim_{k \to \infty} \|d_k\|_A = 0,$$

which gives

$$\lim_{k \to \infty} d_k = 0,$$

i.e., the algorithm converges for any initial guess x_0.
Proof First, we have

$$\|d_k\|_A^2 = (Ad_k, d_k) = (-r_k, d_k) = (r_k, A^{-1} r_k).$$

Then we have

$$\|d_{k+1}\|_A^2 = (Ad_{k+1}, d_{k+1}) = (-r_{k+1}, d_{k+1})$$

and by simple substitution,

$$d_{k+1} = d_k + \alpha_k r_k,$$

$$\|d_{k+1}\|_A^2 = (-r_{k+1}, d_k + \alpha_k r_k),$$

$$= (-r_{k+1}, d_k) - \alpha(r_{k+1}, r_k),$$

$$= (-r_{k+1}, d_k),$$

since

$$(r_{k+1}, r_k) = 0.$$

Thus,

$$\|d_{k+1}\|_A^2 = (-r_{k+1}, d_k),$$

$$= (-r_k + \alpha_k A r_k, d_k),$$

$$= (-r_k, d_k) + \alpha_k (A r_k, d_k),$$

$$= (r_k, A^{-1} r_k) - \alpha_k (A r_k, A^{-1} r_k),$$

$$= (r_k, A^{-1} r_k) - \frac{(r_k, r_k)^2}{(A r_k, r_k)},$$

$$= \|d_k\|_A^2 \left(1 - \frac{(r_k, r_k)}{(A r_k, r_k)} \times \frac{(r_k, r_k)}{(r_k, A^{-1} r_k)} \right).$$

The result follows by applying the Kantorovich inequality.

Unconstrained problem
To minimize a **differentiable** function f

$$\min_{x \in \mathbb{R}^n} f(x)$$

Recall that a global minimizer is a local minimizer, and a local minimizer is a stationary point.

- We may try to find stationary points x, i.e., $\nabla f(x) = 0$ for solving an unconstrained problem.
- When it is difficult to solve $\nabla f(x) = 0$, we look for an approximate solution via iterative methods.

A general algorithmic framework
Choose $x^{(0)}$ and repeat

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}, \quad k = 0, 1, 2, \ldots$$

until some stopping criteria is satisfied.

- $x^{(0)}$ initial guess of the solution.
- $\alpha_k > 0$ is called the step length/step size/learning rate.
- $p^{(k)}$ is a search direction.

Descent Direction
The search direction $p^{(k)}$ should be a descent direction at $x^{(k)}$.

- We say $p^{(k)}$ is a descent direction at $x^{(k)}$ if

$$\nabla f(x^{(k)})^T p^{(k)} < 0$$

- The function value f can be reduced along this descent direction with "appropriate" step length

$$\exists \delta > 0 \quad \text{such that} \quad f(x^{(k)} + \alpha_k p^{(k)}) < f(x^{(k)}) \quad \forall \alpha_k \in (0, \delta)$$

Algorithm 1 Steepest Descent Method

```
1: Initialization: Choose initial point $x^{(0)}$, tolerance $\epsilon > 0$, set $k \leftarrow 0$.
2: while $\|\nabla f(x^{(k)})\| > \epsilon$ do
3:     Find the step length $\alpha_k$ (e.g., by a certain line search rule).
4:     Update the solution:
           $x^{(k+1)} = x^{(k)} - \alpha_k \nabla f(x^{(k)})$
5:     Increment $k \leftarrow k + 1$.
6: end while
7: Output: $x^{(k)}$ (approximate solution)
```

One may choose to use a constant step length (say $\alpha_k = 0.1$), or find it via line search rules:

- Exact line search
- Backtracking line search

Exact line search
Exact line search tries to find α_k by solving the one-dimensional problem:

$$\min_{\alpha>0} \quad \varphi(\alpha) := f(x^{(k)} + \alpha p^{(k)})$$

- In general, exact line search is the most difficult part of the steepest descent method.
- If f is a simple function, it may be possible to obtain an analytical solution for α_k by solving $\varphi'(\alpha) = 0$.

Contour plot A contour is a fixed height $f(x_1, x_2) = c$.

Algorithm 2 Steepest Descent Method with Exact Line Search

```
1: Initialization: Choose initial point $x^{(0)}$, tolerance $\epsilon > 0$, set $k \leftarrow 0$.
2: while $\|\nabla f(x^{(k)})\| > \epsilon$ do
3:     Compute search direction: $p^{(k)} = -\nabla f(x^{(k)})$.
4:     Find optimal step length:
           $\alpha_k = \arg\min_{\alpha>0} f(x^{(k)} + \alpha p^{(k)})$
5:     Update the solution:
           $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$
6:     Increment $k \leftarrow k + 1$.
7: end while
8: Output: $x^{(k)}$ (approximate solution)
```

Properties of steepest descent method with exact line search
Let $\{x^{(k)}\}$ be the sequence generated by the steepest descent method with exact line search.

- **Monotonic decreasing property**:

$$f(x^{(k+1)}) < f(x^{(k)}) \quad \text{if } \nabla f(x^{(k)}) \neq 0.$$

- Suppose f is a *coercive* function with continuous first-order derivatives on $\mathbb{R}^n$. Then some subsequence of $\{x^{(k)}\}$ converges.
  The limit of any convergent subsequence of $\{x^{(k)}\}$ is a stationary point of f.

Backtracking Line Search
Backtracking line search starts with a relatively large step length and iteratively shrinks it (i.e., "backtracking") until the Armijo condition holds.

Algorithm 3 Backtracking Line Search

```
1: Choose $\bar{\alpha} > 0$, $\rho \in (0,1)$, $c_1 \in (0,1)$; Set $\alpha \leftarrow \bar{\alpha}$.
2: repeat
3:     Until
           $f(x^{(k)} + \alpha p^{(k)}) \leq f(x^{(k)}) + c_1 \alpha \nabla f(x^{(k)})^T p^{(k)}$
                                                        ▷ Armijo Condition
4:     $\alpha \leftarrow \rho\alpha$
5: until Armijo condition holds
6: return $\alpha_k = \alpha$
```

Notes:

- $p^{(k)}$ is a descent direction:

$$\nabla f(x^{(k)})^T p^{(k)} < 0$$

- The Armijo condition:

$$f(x^{(k)} + \alpha p^{(k)}) \leq f(x^{(k)}) + c_1 \alpha \nabla f(x^{(k)})^T p^{(k)}$$

  ensures a reasonable amount of decrease in the objective function.
- Example parameter choices:

$$\bar{\alpha} = 1, \quad \rho = 0.9, \quad c_1 = 10^{-4}$$

Algorithm 4 Steepest Descent Method with Backtracking Line Search

```
1: Choose $x^{(0)}$, $\epsilon > 0$, $\bar{\alpha} > 0$, $\rho \in (0,1)$, $c_1 \in (0,1)$; Set $k \leftarrow 0$.
2: while $\|\nabla f(x^{(k)})\| > \epsilon$ do
3:     Compute search direction: $p^{(k)} = -\nabla f(x^{(k)})$.
4:     Set $\alpha \leftarrow \bar{\alpha}$.
5:     repeat
6:         Until Armijo condition holds:
               $f(x^{(k)} + \alpha p^{(k)}) \leq f(x^{(k)}) + c_1 \alpha \nabla f(x^{(k)})^T p^{(k)}$
7:         $\alpha \leftarrow \rho\alpha$.
8:     until Armijo condition holds
9:     $\alpha_k \leftarrow \alpha$.
10:    Update the solution:
           $x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}$
11:    Increment $k \leftarrow k + 1$.
12: end while
13: return $x^{(k)}$.
```

Steepest Descent Method for Multivariate Linear Regression

Algorithm 5 Steepest Descent for Multivariate Linear Regression

```
1: Choose $\beta_0^{(0)}, \beta^{(0)} = (\beta_1^{(0)}, \dots, \beta_p^{(0)})^T$ and $\epsilon > 0$; Set $k \leftarrow 0$.
2: while $\|\nabla L(\beta_0^{(k)}, \beta^{(k)})\| > \epsilon$ do
3:     Determine step length $\alpha_k$.
4:     Update parameters:
           $\beta_0^{(k+1)} = \beta_0^{(k)} - \alpha_k \sum_{i=1}^{n} ((\beta^{(k)})^T x_i + \beta_0^{(k)} - y_i)$
5:     for $j = 1, 2, \dots, p$ do
           $\beta_j^{(k+1)} = \beta_j^{(k)} - \alpha_k \sum_{i=1}^{n} ((\beta^{(k)})^T x_i + \beta_0^{(k)} - y_i) x_{ij}$
6:     end for
7:     Increment $k \leftarrow k + 1$.
8: end while
9: return $\beta_0^{(k)}, \beta^{(k)} = (\beta_1^{(k)}, \dots, \beta_p^{(k)})^T$.
```

Normal Equation

$$\min_{\beta_0, \beta_1, \dots, \beta_p} L(\beta_0, \beta_1, \dots, \beta_p) = \frac{1}{2} \sum_{i=1}^{n} (\beta^T x_i + \beta_0 - y_i)^2$$

$$\hat{X}^T \hat{X} \hat{\beta} = \hat{X}^T Y$$

How to solve

$$\hat{X}^T \hat{X} \hat{\beta} = \hat{X}^T Y$$

Case 1. When $\hat{X}^T \hat{X}$ is invertible, the normal equation implies that

$$\hat{\beta} = (\hat{X}^T \hat{X})^{-1} \hat{X}^T Y$$

is the **unique** solution of linear regression.
This often happens when we face an **over-determined system** — number of training examples n is much larger than number of features p.
We have many training samples to fit but do not have enough degree of freedom.

Case 2. When $\hat{X}^T \hat{X}$ is not invertible, the normal equation will have infinite number of solutions.
$\hat{X}^T \hat{X}$ is not invertible when we face an **under-determined problem** — $n < p$.
We have too many degrees of freedom and do not have enough training samples.
We can apply any method for solving a linear system (e.g., Gaussian elimination) to obtain a solution.

Lecture 3

Classification Binary classification:

- Email: spam/not spam
- Student: fail/pass

We usually assign:

$$\text{label} \begin{cases} 0, & \text{normal state/negative class, e.g., not spam} \\ 1, & \text{abnormal state/positive class, e.g., spam} \end{cases}$$

However, the label assignment can be arbitrary:

$$0 = \text{not spam}, 1 = \text{spam} \quad \text{or} \quad 0 = \text{spam}, 1 = \text{not spam}$$

Data: $x_i \in \mathbb{R}^p, y_i \in \{0, 1\}, i = 1, 2, \dots, n$.

Multi-class classification:

- Iris flower (3 species: Setosa, Versicolor, Virginica)
- Optical character recognition

Data: $x_i \in \mathbb{R}^p, y_i \in \{1, \dots, K\}, i = 1, 2, \dots, n$.

Linear Regression vs. Logistic Regression

Linear Regression

- Data $x_i, y_i \in \mathbb{R}$
- Fit: $f(x) = \beta^T x + \beta_0 = \hat{\beta}^T \hat{x}$, where $\hat{\beta} = [\beta_0; \beta]$, $\hat{x} = [1; x]$

Logistic Regression

- Data $x_i, y_i \in \{0, 1\}$
- Fit:

$$f(x) = g(\hat{\beta}^T \hat{x})$$

where

$$g(z) = \frac{1}{1 + e^{-z}} \quad \text{(logistic function)}$$

so,

$$f(x) = g(\hat{\beta}^T \hat{x}) = \frac{1}{1 + e^{-(\beta^T x + \beta_0)}}$$

Logistic Regression Decision Rule

$$f(x) = g(\hat{\beta}^T \hat{x}), \quad g(z) = \frac{1}{1 + e^{-z}}$$

$$f(x) = p(y = 1 | x; \hat{\beta})$$

Predict $y = 1$ (class 1) if:

$$f(x) \geq 0.5 \quad \text{i.e.,} \quad \hat{\beta}^T \hat{x} \geq 0$$

Predict $y = 0$ (class 0) if:

$$f(x) < 0.5 \quad \text{i.e.,} \quad \hat{\beta}^T \hat{x} < 0$$

Decision Boundary

The set of all $x \in \mathbb{R}^p$ such that:

$$\beta_0 + \beta^T x = 0$$

is called the decision boundary between classes 0 and 1.

The logistic regression has a linear decision boundary; it is:

- a point when $p = 1$
- a line when $p = 2$
- a plane when $p = 3$
- in general a $(p - 1)$-dimensional subspace

Maximum Likelihood Estimation

- Data $(x_i, y_i), i = 1, 2, \dots, n, x_i \in \mathbb{R}^p, y_i \in \{0, 1\}$.
- The likelihood of a single training example (x_i, y_i) is:

$$\text{probability}(x_i \in \text{class } y_i) = \begin{cases} p(y_i = 1 | x_i; \hat{\beta}) = f(x_i), & \text{if } y_i = 1 \\ p(y_i = 0 | x_i; \hat{\beta}) = 1 - f(x_i), & \text{if } y_i = 0 \end{cases}$$

$$= f(x_i)^{y_i} [1 - f(x_i)]^{1 - y_i}$$

- Assuming independence of training samples, the likelihood is:

$$\prod_{i=1}^{n} f(x_i)^{y_i} [1 - f(x_i)]^{1 - y_i}$$

- Want to find $\hat{\beta}$ to maximize the log-likelihood:

$$L(\hat{\beta}) = -\log \left(\prod_{i=1}^{n} f(x_i)^{y_i} [1 - f(x_i)]^{1 - y_i} \right)$$

$$= -\sum_{i=1}^{n} \left(y_i \log f(x_i) + (1 - y_i) \log(1 - f(x_i)) \right)$$

For a single training example (x_i, y_i), the cost is:

$$-y_i \log f(x_i) - (1 - y_i) \log(1 - f(x_i))$$

$$= \begin{cases} -\log f(x_i), & \text{if } y_i = 1 \\ -\log(1 - f(x_i)), & \text{if } y_i = 0 \end{cases}$$

Simplifying the Cost Function

$$\log \left(\frac{f(x_i)}{1 - f(x_i)} \right) = \log \left(\frac{\frac{1}{1 + e^{-\hat{\beta}^T \hat{x}_i}}}{1 - \frac{1}{1 + e^{-\hat{\beta}^T \hat{x}_i}}} \right)$$

$$= \log(e^{\hat{\beta}^T \hat{x}_i}) = \hat{\beta}^T \hat{x}_i$$

$$\log(1 - f(x_i)) = \log \left(1 - \frac{1}{1 + e^{-\hat{\beta}^T \hat{x}_i}} \right)$$

$$= \log \left(\frac{1 + e^{\hat{\beta}^T \hat{x}_i} - 1}{1 + e^{\hat{\beta}^T \hat{x}_i}} \right)$$

$$= -\log \left(1 + e^{\hat{\beta}^T \hat{x}_i} \right)$$

Gradient of the cost function

- **Cost function**

$$L(\beta_0, \beta_1, \dots, \beta_p) = \sum_{i=1}^{n} \log(1 + e^{\beta_0 + \beta^T x_i}) - y_i(\beta_0 + \beta^T x_i)$$

$$\beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$

- **Calculate**

$$\frac{\partial}{\partial \beta_0} L = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i \right) = \sum_{i=1}^{n} (f(x_i) - y_i)$$

$$\frac{\partial}{\partial \beta_1} L = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i \right) x_{i1} = \sum_{i=1}^{n} (f(x_i) - y_i) x_{i1}$$

$$\frac{\partial}{\partial \beta_2} L = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i \right) x_{i2} = \sum_{i=1}^{n} (f(x_i) - y_i) x_{i2}$$

$$\vdots$$

$$\frac{\partial}{\partial \beta_p} L = \sum_{i=1}^{n} \left(\frac{1}{1 + e^{-(\beta_0 + \beta^T x_i)}} - y_i \right) x_{ip} = \sum_{i=1}^{n} (f(x_i) - y_i) x_{ip}$$

—

Solution may not exist

The solution (global minimizer) of the minimization problem

$$\min_{\beta_0, \beta_1, \dots, \beta_p} \sum_{i=1}^{n} \log(1 + e^{\beta_0 + \beta^T x_i}) - y_i(\beta_0 + \beta^T x_i)$$

may not exist. (Regularization will help solve this issue)

Example. $n = 1, x_1 = -1, y_1 = 0$. Then the cost function

$$L(\beta_0, \beta_1) = \log(1 + e^{\beta_0 - \beta_1})$$

We can see that $\min L = 0$. However, this value cannot be attained.

—

Multi-class classification: one-vs-rest

Idea: transfer multi-class classification to multiple binary classification problems

Data: $x_i \in \mathbb{R}^p, y_i \in \{1, \dots, K\}, i = 1, 2, \dots, n$.

For each $k \in \{1, 2, \dots, K\}$

1. Construct a new label $\tilde{y}_i = 1$ if $y_i = k$ and $\tilde{y}_i = 0$ otherwise
2. Learn a binary classifier f_k with data $x_i, \tilde{y}_i$

Multi-class classifier predicts class k where k achieves the maximal value

$$\max_{k \in \{1, 2, \dots, K\}} f_k(x)$$

Overfitting

- **Underfitting:** a model is too simple and does not adequately capture the underlying structure of the data
- **Overfitting:** a model is too complicated and contains more parameters that can be justified by the data; it does not generalize well from training data to test data
- **Good fit:** a model adequately learns the training data and generalizes well to test data

Ridge regularization
In linear/logistic regression, over-fitting occurs frequently. Regularization will make the model simpler and works well for most of the regression/classification problems.

- Ridge regularization:

$$\lambda \|\beta\|^2 = \lambda \sum_{j=1}^{p} \beta_j^2$$

λ: regularization parameter, $\|\beta\|^2$: regularizer

- It is differentiable. It forces β_j's to be small
- Extreme case: suppose λ is a huge number, it will push all β_j's to be zero and the model will be naive

—
Ridge regularized problems

- Logistic regression + ridge regularization (Gradient methods can be used, a solution exists)

$$\min_{\beta_0,\beta_1,\ldots,\beta_p} \sum_{i=1}^{n} \log(1 + e^{\beta_0 + \beta^T x_i}) - y_i(\beta_0 + \beta^T x_i) + \lambda \sum_{j=1}^{p} \beta_j^2$$

- Linear regression + ridge regularization (Apply either normal equation or gradient methods)

$$\min_{\beta_0,\beta_1,\ldots,\beta_p} \frac{1}{2} \sum_{i=1}^{n} (\beta^T x_i + \beta_0 - y_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

—
Lasso regularization

- Lasso (Least Absolute Shrinkage and Selection Operator) regularization:

$$\lambda \|\beta\|_1 = \lambda \sum_{j=1}^{p} |\beta_j|$$

- It is non-differentiable. It forces some β_j's to be exactly zero
- It can be used for feature selection (model selection). It selects important features (removing non-informative or redundant features)
- When λ is larger, less features will be selected