Mathematics Essentials

Ke Chen

Department of Computer Science, The University of Manchester

Ke.Chen@manchester.ac.uk

Outline

Linear Algebra

Vector, Vector Space, Matrix, Eigen Analysis, Linear Transformation and Matrix Decomposition

Random Vector

Random Variable, Random Vector and Multivariate Statistics

Matrix Calculus

Vector/Matrix Calculus Rules and Illustrative Example

Optimisation Basics

Optimality Condition, Constraint Optimisation and Illustrative Example

Machine Learning Principle

Learning Model, Loss/Utility Function and Learning Algorithm

Linear Algebra: Notation

- Vector notation: column versus row vector of d elements

$$\text{column vector}: \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}, \quad \text{row vector}: \boldsymbol{x}^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_d \end{bmatrix}.$$

- Matrix notation: rectangle matrix $(d \times n)$ versus square matrix $(d \times d)$

$$A = \left(A_{ij}\right)_{d\times n} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{d1} & A_{d2} & \cdots & A_{dn} \end{bmatrix}, \quad B = \left(B_{ij}\right)_{d\times d} = \begin{bmatrix} B_{11} & \cdots & B_{1d} \\ B_{21} & \cdots & B_{2d} \\ \vdots & \ddots & \vdots \\ B_{d1} & \cdots & B_{dd} \end{bmatrix}.$$

$$A^T = \left(A_{ji}\right)_{n\times d} \qquad B^T = \left(B_{ji}\right)_{d\times d}$$

- Inner (dot) product of two vectors

$$< \boldsymbol{x}, \boldsymbol{y} >= \boldsymbol{x}^T \boldsymbol{y} = \sum_{i=1}^{d} x_i y_i.$$

- Magnitude of a vector

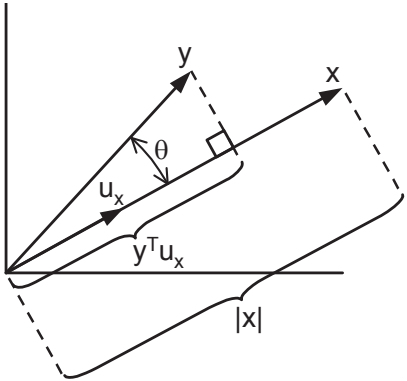
$$||\boldsymbol{x}|| = \sqrt{\boldsymbol{x}^T \boldsymbol{x}} = (\sum_{i=1}^{d} x_i x_i)^{\frac{1}{2}} = (\sum_{i=1}^{d} x_i^2)^{\frac{1}{2}}.$$

- Angle between two vectors

$$\cos\theta = \frac{\boldsymbol{x}^T \boldsymbol{y}}{||\boldsymbol{x}||||\boldsymbol{y}||} = \left(\frac{\boldsymbol{x}}{||\boldsymbol{x}||}\right)^T \left(\frac{\boldsymbol{y}}{||\boldsymbol{y}||}\right).$$

- Orthogonal projection of $\boldsymbol{y}$ onto $\boldsymbol{x}$

$$\boldsymbol{y} \mapsto \boldsymbol{x} : \ (\boldsymbol{y}^T \boldsymbol{x})\boldsymbol{u}_x \ \text{ where } \boldsymbol{u}_x = \frac{\boldsymbol{x}}{||\boldsymbol{x}||} \ (||\boldsymbol{u}_x|| = 1).$$



- Orthogonal and Orthonormal vectors

  If $\boldsymbol{x}$ is orthogonal to $\boldsymbol{y}$, then $\boldsymbol{x}^T\boldsymbol{y} = 0$ ($\because \cos 90^0 = 0$).

  If $\boldsymbol{x}$ is orthonormal to $\boldsymbol{y}$, then $\boldsymbol{x}^T\boldsymbol{y} = 0$ and $||\boldsymbol{x}|| = ||\boldsymbol{y}|| = 1$.

- Outer product of two vectors

$$\boldsymbol{x} \otimes \boldsymbol{y} = \boldsymbol{x}\boldsymbol{y}^T = (x_i y_j)_{d\times n}, \quad \text{where } \boldsymbol{x} = (x_i)_{d\times 1} \text{ and } \boldsymbol{y} = (y_j)_{n\times 1}.$$

- Linearly Dependent versus Linearly Independent

  If a set of vectors, $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, are linearly dependent, then there exists a set of coefficients, $c_1, c_2, \cdots, c_n$, such that

$$c_1\boldsymbol{x}_1 + c_2\boldsymbol{x}_2 + \cdots + c_n\boldsymbol{x}_n = \boldsymbol{0}, \ \exists c_k \neq 0, \ k = 1, \cdots, n.$$

  If a set of vectors, $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$, are linearly independent, then there exists a set of coefficients, $c_1, c_2, \cdots, c_n$, such that

$$c_1\boldsymbol{x}_1 + c_2\boldsymbol{x}_2 + \cdots + c_n\boldsymbol{x}_n = \boldsymbol{0}, \ \forall c_k = 0, \ k = 1, \cdots, n.$$

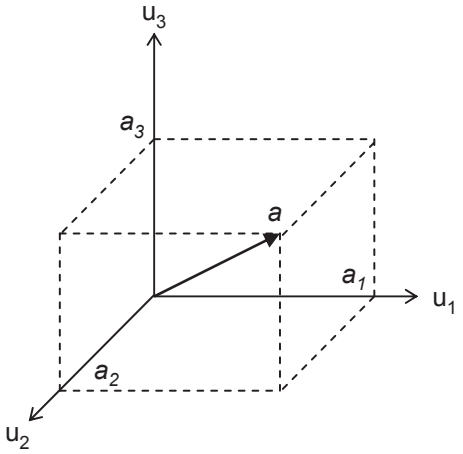
- A d-dimensional vector space, $\mathbb{R}^d$, is the space where all the d-dimensional vectors, $\forall \boldsymbol{a} \in \mathbb{R}^d$, reside.
- A vector space is represented by a set of basis vectors, $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_d$, such that $\forall \boldsymbol{a} \in \mathbb{R}^d$,

$$\boldsymbol{a} = a_1\boldsymbol{u}_1 + a_2\boldsymbol{u}_2 + \cdots + a_d\boldsymbol{u}_d.$$

  The coefficients, $a_1, a_2, \cdots, a_d$, are the components of $\boldsymbol{a}$ on the basis.
- Basis Formation
  - Necessary-sufficient condition: $\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_d$ must be linearly independent.
  - Orthogonal versus Orthonormal basis

$$\boldsymbol{u}_i^T\boldsymbol{u}_j = \begin{cases} \neq 0, & i = j \\ 0, & i \neq j \end{cases}, \quad \boldsymbol{u}_i^T\boldsymbol{u}_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

- Orthonormal bases lead to the Cartesian coordinate system, a point representation of vector space.

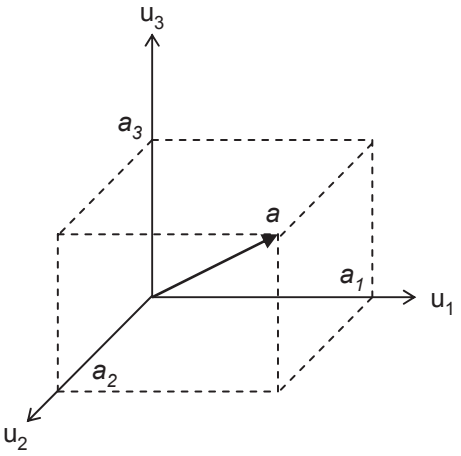


Example: 3-D vector space ($d = 3$)

- Orthonormal basis vector:
  $\boldsymbol{u}_1 = [1\ 0\ 0]^T$, $\boldsymbol{u}_2 = [0\ 1\ 0]^T$, and $\boldsymbol{u}_3 = [0\ 0\ 1]^T$.
- We can denote any 3-D vector with $\boldsymbol{a} = [a_1\ a_2\ a_3]^T$ for $\boldsymbol{a} \in \mathbb{R}^3$ or $a_i \in \mathbb{R}$, $i = 1, 2, 3$.
- To use the orthonormal bases to represent any vector in 3-D, we simply conduct the orthogonal projection onto all 3 basis vectors to form the coefficients:

$$\begin{aligned} \boldsymbol{a} &= [\boldsymbol{a}^T\boldsymbol{u}_1]\boldsymbol{u}_1 + [\boldsymbol{a}^T\boldsymbol{u}_2]\boldsymbol{u}_2 + [\boldsymbol{a}^T\boldsymbol{u}_3]\boldsymbol{u}_3 \\ &= a_1\begin{bmatrix}1\\0\\0\end{bmatrix} + a_2\begin{bmatrix}0\\1\\0\end{bmatrix} + a_3\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix} \end{aligned}$$

- The coefficients, (a_1, a_2, a_3), form a point representation of vector, $\boldsymbol{a}$, in the Cartesian coordinate system.



- Matrix Product: $C = AB, A = (A_{ij})_{d\times n}, B = (B_{ij})_{n\times k}, C = (C_{ij})_{d\times k}, C_{ij} = \sum_{p=1}^{n} A_{ip}B_{pj}$.

$$C = AB = \begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1n} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{d1} & A_{d2} & A_{d3} & \cdots & A_{dn} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1k} \\ B_{21} & B_{22} & \cdots & B_{2k} \\ B_{31} & B_{32} & \cdots & B_{3k} \\ \vdots & \ddots & \vdots & \vdots \\ B_{n1} & B_{n2} & \cdots & B_{nk} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & \cdots & C_{1k} \\ C_{21} & C_{22} & C_{23} & \cdots & C_{2k} \\ C_{31} & C_{32} & C_{33} & \cdots & C_{3k} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ C_{d1} & C_{d2} & C_{d3} & \cdots & C_{dk} \end{bmatrix}$$

Property: $XY \neq YX, \quad XYZ = (XY)Z = X(YZ)$.

- Hadamard Product: $C = A \odot B, A = (A_{ij})_{d\times n}, B = (B_{ij})_{d\times n}, C = (C_{ij})_{d\times n}, C_{ij} = A_{ij}B_{ij}$.

$$C = A \odot B = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{d1} & A_{d2} & \cdots & A_{dn} \end{bmatrix} \odot \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1n} \\ B_{21} & B_{22} & \cdots & B_{2n} \\ \vdots & \ddots & \vdots & \vdots \\ B_{d1} & B_{n2} & \cdots & B_{dn} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1n} \\ C_{21} & C_{22} & \cdots & C_{2n} \\ \vdots & \ddots & \vdots & \vdots \\ C_{d1} & C_{d2} & \cdots & C_{dn} \end{bmatrix}$$

Linear Algebra: Square Matrix

- Determinant of $A = (A_{ij})_{d\times d}$ is defined by $|A| = \sum_{k=1}^{d} A_{ik}|A_{ik}|(-1)^{k+i}$, $|A| = |A^T|$ where A_{ik} is the minor matrix formed by removing the ith row and the kth column of A.
- Trace of $A = (A_{ij})_{d\times d}$ is the sum of its diagonal elements; i.e., $\mathrm{tr}(A) = \sum_{k=1}^{d} A_{kk}$.
- Rank of $A = (A_{ij})_{d\times d}$ is the number of linearly independent rows/columns, $\mathrm{rank}(A)$.
- Non-singular $A = (A_{ij})_{d\times d}$ is the square matrix of $\mathrm{rank}(A) = d$. In this case, $|A| \neq 0$.
- Orthonormal $A = (A_{ij})_{d\times d}$ is the square matrix of the property: $A^T A = AA^T = I_d$ where I_d is the identity matrix, a matrix of 1/0 in diagonal/off-diagonal elements.
- Inverse of $A = (A_{ij})_{d\times d}$ is the square matrix A^{-1} of the property: $A^{-1}A = AA^{-1} = I_d$. The inverse of $A = (A_{ij})_{d\times d}$ exists if and only if it is a non-singular square matrix.
- Semi-definite matrix $A = (A_{ij})_{d\times d}$ is a square matrix of the property: $\forall \boldsymbol{x} \in \mathbb{R}^d$, $\boldsymbol{x}^T A \boldsymbol{x} \geq 0$ (positive semi-definite); $\boldsymbol{x}^T A \boldsymbol{x} \leq 0$ (negative semi-definite)

Note: Geometric meaning of matrix properties is important for ML (see relevant Wikipedia articles for details).

- Given a square matrix, $A = (A_{ij})_{d \times d}$, the eigen analysis would find an eigenvector, $\boldsymbol{v}$, and its corresponding eigenvalue, λ, such that

$$A\boldsymbol{v} = \lambda \boldsymbol{v} \iff \begin{cases} \boldsymbol{v} \text{ is an eigenvector} \\ \lambda \text{ is the corresponding eigenvalue} \end{cases}$$

  The process of obtaining eigenvalues and eigenvectors is named Eigen Analysis.

- Property
  - If $A = (A_{ij})_{d \times d}$ is non-singular, all d eigenvalues are non-zero, $\lambda_i \neq 0$ $(i = 1, \cdots, d)$, and

$$|A| = \prod_{i=1}^{d} \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_d.$$

  - If $A = (A_{ij})_{d \times d}$ is real and symmetric, $A = A^T$,
    (i) all eigenvalues are real;
    (ii) The eigenvectors associated with distinct eigenvalues are orthogonal.

Note: Geometric meaning of eigenvectors/eigenvalues is very important for unsupervised representation learning.

- Linear Transformation is a linear mapping, $P = (P_{ij})_{p \times d}$, from a vector space, $\boldsymbol{x} \in \mathbb{R}^d$, onto another vector space, $\boldsymbol{y} \in \mathbb{R}^p$; i.e. $\boldsymbol{y} = P\boldsymbol{x}$; i.e., $y_i = \sum_{k=1}^{d} p_{ij} x_j, \ i = 1, \cdots, p$.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \cdots & P_{1d} \\ P_{21} & P_{22} & P_{23} & \cdots & P_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{p1} & P_{p2} & P_{p3} & \cdots & P_{pd} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}.$$

- In the ML context, dimensionality of two vector spaces should be different.
  - For low-dimensional representation (via dimension reduction), $p < d$.
  - For over-complete (sparse) representation, $p > d$.
- All linear representation learning models aim at learning a projection matrix, P, for feature extraction to generate a new representation, $\boldsymbol{y}$, for a raw data point, $\boldsymbol{x}$.

Spectral Decomposition is a powerful tool for matrix decomposition, a foundation of linear algebra in computers.

- Given a real symmetric matrix, $A = \left(A_{ij}\right)_{d\times d}$ where $A_{ij} = A_{ji}$, it can be decomposed by product of matrices consisting of its eigenvectors and eigenvalues:

$$A_{d\times d} = V_{d\times d}\Sigma_{d\times d}V_{d\times d}^T.$$

  - V is an orthogonal matrix, $V^TV = I$, where column i is the ith eigenvector of A.
  - Σ is a diagonal matrix where $\Sigma_{ii} = \lambda_i$ and $\Sigma_{ij} = 0$ if $i \neq j$ where λ_i is the ith eigenvalue of A.

- Example:

$$\begin{bmatrix} 1 & 3 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Singular Value Decomposition (SVD) is yet another powerful tool for matrix decomposition, a foundation of linear algebra in computers.

- Given a matrix of any form, $X_{d\times N}$, SVD decomposes it into the following form:

$$X_{d\times N} = U_{d\times d}\Sigma_{d\times N}V^T_{N\times N}.$$

  - U is an orthogonal matrix, $U^TU = I_{d\times d}$, where column i is the ith eigenvector of XX^T.
  - V is an orthogonal matrix, $V^TV = I_{N\times N}$, where column i is the ith eigenvector of X^TX.
  - Σ is a "diagonal" matrix where $\Sigma_{ii} = \sqrt{\lambda_i}$, $\Sigma_{ii} \geq \Sigma_{jj}$ if $i > j$ and λ_i is the ith eigenvalue shared by XX^T and X^TX.

- Example:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Random Vector

- Random variable is a variable of which values depend on outcomes of a random phenomenon.
- Formally, it is a measurable function defined on a probability space that maps from the sample space to the real numbers; probability for discrete variables or density for continuous variables.
- Random vector refers to multi-dimensional generalisation of the concept of random variable: a vector of which elements are random variables.
- In general, a random vector can be described with measures similar to those defined for scalar random variables.

For a random vector of d random variables, $\mathcal{X} = (\mathcal{X}_1, \mathcal{X}_2, \cdots, \mathcal{X}_d)$, a sample of $\mathcal{X}$ that has N points is $X = (X_{ij})_{d \times N}$. Similar to scalar random variables, their statistics can be measured by

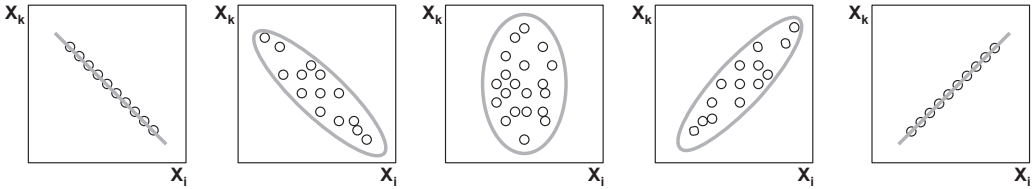
- Mean Vector (empirical)

$$\boldsymbol{m} = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_d \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{n=1}^{N} X_{1n} \\ \frac{1}{N}\sum_{n=1}^{N} X_{2n} \\ \vdots \\ \frac{1}{N}\sum_{n=1}^{N} X_{dn} \end{bmatrix} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n, \quad \text{where } \boldsymbol{x}_n = \begin{bmatrix} X_{1n} \\ X_{2n} \\ \vdots \\ X_{dn} \end{bmatrix}, \ n = 1, 2, \cdots, N.$$

- Covariance Matrix (empirical)

$$S_{d \times d} = \begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1d} \\ C_{21} & C_{22} & \cdots & C_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ C_{d1} & C_{d2} & \cdots & C_{dd} \end{bmatrix}, \quad C_{ij} = \begin{cases} \frac{1}{N-1}\sum_{n=1}^{N}(X_{in} - m_i)(X_{jn} - m_j) & i \neq j \\ \frac{1}{N-1}\sum_{n=1}^{N}(X_{in} - m_i)^2 \Longleftrightarrow \Sigma_i^2 & i = j \end{cases}$$

- The covariance matrix characterises the tendency of any pair of variables in a random vector to vary together or co-vary.
- covariance elements can be expressed by $C_{ik} = \rho_{ik}\Sigma_i\Sigma_k$, where ρ_{ik} is the correlation coefficient. Thus, the covariance has several important properties as follows:
  - If $C_{ik} > 0$, $\mathcal{X}_i$ and $\mathcal{X}_k$ tend to increase together.
  - If $C_{ik} < 0$, $\mathcal{X}_i$ tends to decrease when $\mathcal{X}_k$ tend to increase.
  - If $C_{ik} = 0$, $\mathcal{X}_i$ and $\mathcal{X}_k$ are uncorrelated.
  - $-1 \leq \rho_{ik} \leq 1$, hence $|C_{ik}| \leq \Sigma_i\Sigma_k$



- **Matrix derivatives: 3-step procedure**
  - Step 1: know the dimension of dependent (function)/independent variables
  - Step 2: convert vector/matrix into element-wise calculation
  - Step 3: put back into the vector/matrix form

	Scalar	Vector	Matrix
Scalar	$\frac{\mathrm{d}y}{\mathrm{d}x}$	$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}x} = \left[\frac{\partial y_i}{\partial x}\right]$	$\frac{\mathrm{d}\mathbf{Y}}{\mathrm{d}x} = \left[\frac{\partial y_{ij}}{\partial x}\right]$
Vector	$\frac{\mathrm{d}y}{\mathrm{d}\mathbf{x}} = \left[\frac{\partial y}{\partial x_j}\right]$	$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\mathbf{x}} = \left[\frac{\partial y_i}{\partial x_j}\right]$	
Matrix	$\frac{\mathrm{d}y}{\mathrm{d}\mathbf{X}} = \left[\frac{\partial y}{\partial x_{ji}}\right]$		

- **Derivative Rule**
  Assume that $f(x)$ and $g(x)$ are differentiable functions (the derivative exists) and c is a constant. The following rules are applicable to scalar, vector and matrix variables.
  - Derivative with respect to constant or irrelevant independent variables

$$\frac{d}{dx}(c) = 0, \quad \frac{d}{dx}(cf(x)) = c\frac{df(x)}{dx}.$$

  - Addition and Substraction Rule

$$\frac{d}{dx}(f(x) \pm g(x)) = \frac{df(x)}{dx} \pm \frac{dg(x)}{dx}.$$

  - Product and Quotient Rule

$$\frac{d}{dx}(f(x)g(x)) = g(x)\frac{df(x)}{dx} + f(x)\frac{dg(x)}{dx}, \quad \frac{d}{dx}\left(\frac{f(x)}{g(x)}\right) = \frac{g(x)\frac{df(x)}{dx} - f(x)\frac{dg(x)}{dx}}{g^2(x)}.$$

  - Chain Rule

$$\frac{d}{dx}\Big(f\big[g(x)\big]\Big) = \frac{df(x)}{dg(x)}\frac{dg(x)}{dx}.$$

- **Illustrative Example**

$$\text{If } \boldsymbol{c} \in \mathbb{R}^d,\ \boldsymbol{x} \in \mathbb{R}^d \text{ and } f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x},\quad \frac{df(\boldsymbol{x})}{d\boldsymbol{x}} = ?$$

  - Step 1: Dimension

    We know $\boldsymbol{c}$ and $\boldsymbol{x}$ are d-dimensional vectors and $\boldsymbol{c}^T\boldsymbol{x}$ is scalar. Therefore, we face a situation: $\boldsymbol{x}$ is a vector and $y = f(\boldsymbol{x})$ is a scalar.

  - Step 2: Element-wise calculation

$$y = \boldsymbol{c}^T\boldsymbol{x} = \sum_{j=1}^{d} c_j x_j = c_1 x_1 + \cdots + c_j x_j + \cdots + c_d x_d,\ \frac{dy}{dx_j} = c_j \quad j = 1, \cdots, d.$$

  - Step 3: Vector form

$$\frac{dy}{d\boldsymbol{x}} = \left[\frac{dy}{dx_j}\right] \Longrightarrow \frac{df(\boldsymbol{x})}{d\boldsymbol{x}} = \left[c_j\right]_{d\times 1} = \boldsymbol{c}.$$

- **Optimality Condition**

  For a differentiable objective function $f(\boldsymbol{x})$ and $\boldsymbol{x} \in \mathbb{R}^d$, the optimum point $\boldsymbol{x}^* \in \mathbb{R}^d$ must satisfy the following conditions:

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*) = \left.\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^*} = \mathbf{0} \iff \left.\frac{\partial f(\boldsymbol{x})}{\partial x_j}\right|_{x_j=x_j^*} = 0,\ j = 1, \cdots, d.$$

- **Constrained Optimisation**

  A constrained optimisation problem often has the following form:

$$\textbf{minimise}\ \ f(\boldsymbol{x}) \quad \textbf{subject to}\ \ c_i(\boldsymbol{x}) = 0, \quad i = 1, 2, \cdots, m$$

  where $f(\boldsymbol{x})$ and $c_i(\boldsymbol{x})$ are objective and constraint functions, respectively.

- **Lagrangian Multipliers: A solution to constrained optimisation**
  - Key idea: convert a constrained optimization problem into an unconstrained counterpart
  - Introducing m "new" variables, Lagrangian multipliers $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_m)^T$, to constraints, $\boldsymbol{c}(\boldsymbol{x}) = (c_1(\boldsymbol{x}), \cdots, c_m(\boldsymbol{x}))^T$, to form a unconstraint objective function:

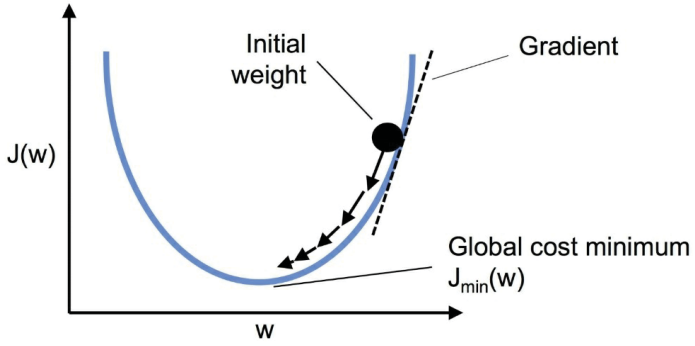
$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{c}(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i c_i(\boldsymbol{x}).$$

  - The optimality condition for the unconstrained objective function, $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$, at the optimum point, $(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ is

$$\left.\frac{\partial \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}^*} = \mathbf{0} \quad \textbf{and} \quad \left.\frac{\partial \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \boldsymbol{\lambda}}\right|_{\boldsymbol{\lambda}=\boldsymbol{\lambda}^*} = \mathbf{0}.$$

That is, $$\left.\frac{\partial \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial x_j}\right|_{x_j=x_j^*} = 0 \quad \textbf{and} \quad \left.\frac{\partial \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial \lambda_i}\right|_{\lambda_i=\lambda_i^*} = 0, \quad j = 1, \cdots, d; \; i = 1, \cdots, m.$$

- **Gradient-based Local Search**
  - A generic optimisation technique to find out a local optimum of any differentiable functions
  - It lays the foundation for deep learning algorithms and other optimisation problems in ML.
  - Given a loss/utility function, $J(\boldsymbol{w})$, gradient descent/ascent for minimum/maximum:
    1. Initialisation: set $\boldsymbol{w} = \boldsymbol{w}_0$ randomly (or with a priori knowledge)
    2. Update rule: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t \mp \eta \nabla_{\boldsymbol{w}} J(\boldsymbol{w})\big|_{\boldsymbol{w}=\boldsymbol{w}_t}$, $0 < \eta \leq 1$.
    3. Repeat (2) until a stopping condition is met.



Illustrative Example: Derivation of SVM Learning Algorithm

- Training dataset: $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ and $y \in \{+1, -1\}$.
- Convert the Margin into the objective (loss) function

$$\text{Margin} = \frac{2}{||\boldsymbol{w}||} \implies \mathcal{L}_M(\boldsymbol{w}) = \frac{1}{2}||\boldsymbol{w}||^2 = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}.$$

- Constraints required for all the training examples

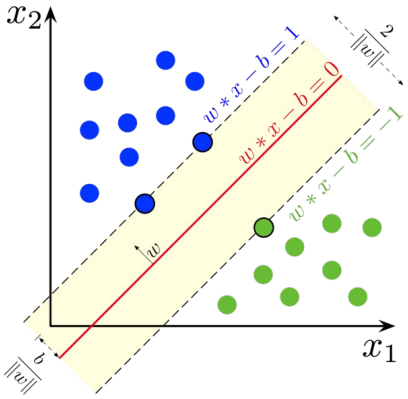
  For $y_i = +1$, $\boldsymbol{w}^T\boldsymbol{x}_i - b \geq 1$.

  For $y_i = -1$, $\boldsymbol{w}^T\boldsymbol{x}_i - b \leq -1$.

  subject to $y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) \geq 1, \quad i = 1, 2, \cdots, N.$

- Construct the unconstrained loss function with Lagrangian multipliers, $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_N)^T$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}; \mathcal{D}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{N} \lambda_i [y_i(\boldsymbol{w}^T\boldsymbol{x}_i - b) - 1].$$



Illustrative Example: Derivation of SVM Learning Algorithm

- To find out optimal parameters, $\boldsymbol{w}$ and b, applying the optimality condition to the loss function:

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}; \mathcal{D})}{\partial \boldsymbol{w}} = \mathbf{0} \implies \boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \boldsymbol{x}_i. \tag{1}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}; \mathcal{D})}{\partial b} = 0 \implies \sum_{i=1}^{N} \lambda_i y_i = 0. \tag{2}$$

- Dual loss function obtained by inserting Eqs.(1) and (2) into the original loss function

$$\mathcal{L}_{dual}(\boldsymbol{\lambda}; \mathcal{D}) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j.$$

- Apply the quadratic programming technique to $L_{dual}(\boldsymbol{\lambda}; \mathcal{D})$ to find optimal parameters, $\boldsymbol{\lambda}^*$. The corresponding $\boldsymbol{x}_i$ is a support vector (SV) if $\lambda_i \neq 0$ and not SV otherwise.
- As the inner product, $\boldsymbol{x}_i^T \boldsymbol{x}_j$, is only required in $L_{dual}(\boldsymbol{\lambda}; \mathcal{D})$, it can be extended to kernel SVM: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j)$, where $\phi(\cdot)$ is a non-linear mapping function and $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is its corresponding kernel function.

Where does a machine learning algorithm come from?

Learning Model

A mechanism that encodes "knowledge" for problem solving, which is often a parametric model. Learning is going to find out "appropriate" model parameters.

Cost/Loss (Utility) Function

A function defined as the performance criterion for a specific learning goal based on a given parametric (learning) model and training data to judge how well the parameters of this model are set based on a training dataset.

Learning Algorithm

The algorithm comes from an optimisation process that minimises (maximises) the cost/loss (utility) function with respect to model parameters, which derives a learning rule/algorithm for finding out "appropriate" model parameters with a given training dataset.

Fact: For the same task, there may be different loss functions, which result in different learning algorithms. Applying different optimisation techniques to the same loss function also leads to different learning algorithms.

Reference

If you want to deepen your understanding and learn something beyond this lecture, you can self-study the optional references below.

[Goodfellow et al., 2016] Goodfellow I., Bengio Y., and Courville A. (2016): *Deep Learning*, MIT Press. (Chapters 2-4)

[Barber, 2012] Barber D. (2012): *Bayesian Reasoning and Machine Learning*, Cambridge University Press. (Appendix)

[Deisenroth et al., 2020] Deisenroth M.P., Faisal A.A., and Ong C.S. (2020): *Mathematics for Machine Learning*, Cambridge University Press. (Chapters 2-7)