MSE 546: Advanced Machine Learning

Sirisha Rambhatla

University of Waterloo

Lecture

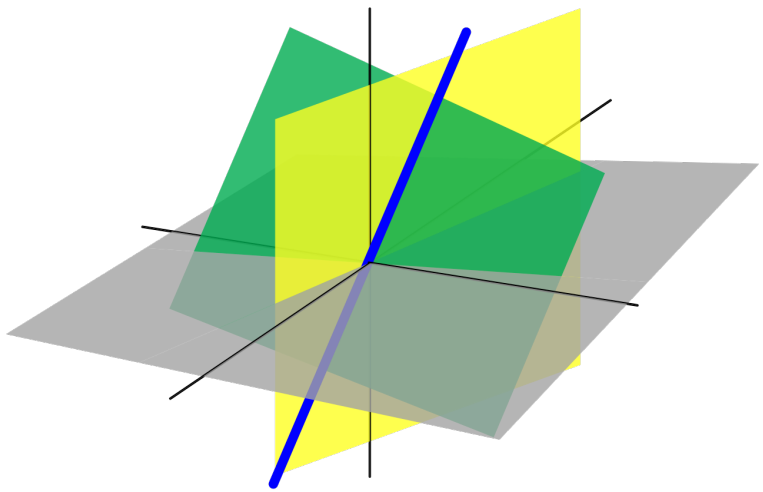
Math Background: Linear Algebra and Optimization

Outline

Outline

Ease of notation and much more!

The properties of data samples (vectors) in higher dimensions lead to key insights, e.g. dimensionality reduction. It also makes our lives easier in analysis.



For $\boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^n$ inner-product is represented as

$$\sum_{i=1}^{n} w_i x_i$$

Scalar Representation

$$\boldsymbol{w}^\top \boldsymbol{x}$$

Vector Representation

Linear Algebra

Linear algebra is the study of matrices and vectors, and other generalizations such as tensors.

- vector
- matrices
- tensors

Definition and Notation: Vectors

Vector: a vector $\boldsymbol{x} \in \mathbb{R}^n$ is a list of n numbers, usually written as a column vector:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

- a vector of all ones is denoted by $\mathbf{1}$
- a vector of all zeros is denoted by $\mathbf{0}$
- a unit vector $\boldsymbol{e}_i$ is a vector of all 0's, except 1 at the i-th entry aka **one-hot** vector

Definition and Notation: Matrices

Matrix: A matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with m rows and n columns is a 2d array of numbers, arranged as follows:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

- if $m = n$ the matrix is said to be **square**
- $\mathbf{A}_{ij}/A_{ij}$ or $\boldsymbol{A}_{i,j}/A_{i,j}$ to denotes the entry in i-th row and j-th column
- $\mathbf{A}_{i,:}$ to denote the i-th row and $\mathbf{A}_{:,j}$ to denote the j-th column.
- convention: all vectors as column vectors by default
- we use bold upper case letters $\mathbf{A}$ to denote matrices, bold lower case letters $\boldsymbol{a}$ to denote vectors, and non-bold letters a to denote scalars.

Definition and Notation

Matrices continued ...

We can view a matrix as a set of columns stacked along the horizontal axis:

$$\mathbf{A} = \begin{bmatrix} | & | & & | \\ \mathbf{A}_{:,1} & \mathbf{A}_{:,2} & \cdots & \mathbf{A}_{:,n} \\ | & | & & | \end{bmatrix} \qquad \mathbf{A} = [\mathbf{A}_{:,1}, \mathbf{A}_{:,2}, \ldots, \mathbf{A}_{:,n}]$$

also view a matrix as a set of rows stacked along the vertical axis

$$\mathbf{A} = \begin{bmatrix} — & \mathbf{A}_{1,:}^{\mathsf{T}} & — \\ — & \mathbf{A}_{2,:}^{\mathsf{T}} & — \\ & \vdots & \\ — & \mathbf{A}_{m,:}^{\mathsf{T}} & — \end{bmatrix} \qquad \mathbf{A} = [\mathbf{A}_{1,:}; \mathbf{A}_{2,:}; \ldots; \mathbf{A}_{m,:}]$$

Definition and Notation

Transpose of a Matrix

The transpose of a matrix results from "flipping" the rows and columns. Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, its transpose, written $\mathbf{A}^\top \in \mathbb{R}^{m \times n}$, is defined as $(\mathbf{A}^\top)_{ij} = (\mathbf{A})_{ji}$

$$(\mathbf{A}^\mathsf{T})^\mathsf{T} = \mathbf{A}$$
$$(\mathbf{AB})^\mathsf{T} = \mathbf{B}^\mathsf{T}\mathbf{A}^\mathsf{T}$$
$$(\mathbf{A} + \mathbf{B})^\mathsf{T} = \mathbf{A}^\mathsf{T} + \mathbf{B}^\mathsf{T}$$

- If for a square matrix $\mathbf{A} = \mathbf{A}^\top$, it is called **symmetric**

Vector Spaces

Vector Spaces A vector space is a collection of such vectors, which can be added together, and scaled by scalars (1-dimensional numbers), in order to create new points. These operations are defined to operate element-wise.

Vector Addition and Scaling For a vector $\boldsymbol{x} \in \mathbb{R}^n$

- **Addition:** $\boldsymbol{x} + \boldsymbol{y} = (x_1 + y_1, \ldots, x_n + y_n)$
- **Scaling:** $c\boldsymbol{x} = (cx_1, ..., cx_n)$, where $c \in \mathbb{R}$

Linear independence, spans and basis sets

For a set of vectors $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$:

Linear Independence: If none of these vectors can be represented as a linear combination of the remaining vectors.

$$\boldsymbol{x}_j \text{ cannot be represented as } \sum_{i=1, i\neq j}^{n-1} \alpha_i \boldsymbol{x}_i$$

Span: Span of a set of vectors $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ is defined as the set of all vectors that can be expressed as a linear combination of $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$

$$\text{span}(\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}) \triangleq \left\{ \boldsymbol{v} : \boldsymbol{v} = \sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i, \quad \alpha_i \in \mathbb{R} \right\}$$

Basis $\mathcal{B}$ is a set of linearly independent vectors that span the whole space, i.e., $\text{span}(\mathcal{B}) = \mathbb{R}^n$, e.g. standard basis uses the coordinate vectors $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n$.

Range and Null Space of a matrix

Range Space aka Column Space Viewing $\mathbf{A} \in \mathbb{R}^{m \times n}$ as a set of n vectors in $\mathbb{R}^m$. The range of this matrix is the span of the columns of $\mathbf{A}$, i.e.

$$\mathrm{range}(\mathbf{A}) \triangleq \{\boldsymbol{v} \in \mathbb{R}^m : \boldsymbol{v} = \mathbf{A}\boldsymbol{x}, \boldsymbol{x} \in \mathbb{R}^n\}$$

Null Space The set of all vectors $\boldsymbol{x} \in \mathbb{R}^n$ that get mapped to the null vector when multiplied by $\mathbf{A}$, i.e.,

$$\mathrm{nullspace}(\mathbf{A}) \triangleq \{\boldsymbol{x} \in \mathbb{R}^n : \mathbf{A}\boldsymbol{x} = \mathbf{0}\}$$

The span of the rows of $\mathbf{A}$ is the complement to the nullspace of $\mathbf{A}$

Vector and Matrix Norms

Vector Norms A norm of a vector $\|\boldsymbol{x}\|$ is, informally, a measure of the "length" of the vector. More formally, a **norm** is any function $f : \mathbb{R}^n \to \mathbb{R}$ that satisfies four properties:

- For all $\boldsymbol{x} \in \mathbb{R}^n$, $f(\boldsymbol{x}) \geq 0$ (**non-negativity**).
- $f(\boldsymbol{x}) = 0$ if and only if $\boldsymbol{x} = 0$ (**definiteness**)
- For all $x \in \mathbb{R}^n$, $t \in \mathbb{R}$, $f(t\boldsymbol{x}) = |t|f(\boldsymbol{x})$ (**homogeneity**)
- For all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, $f(\boldsymbol{x} + \boldsymbol{y}) \leq f(\boldsymbol{x}) + f(\boldsymbol{y})$ (**triangle inequality**)

Common Examples

- **p-norm** $\|\boldsymbol{x}\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$
- **2-norm aka Euclidean norm** $\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2)}$ also $\|\boldsymbol{x}\|_2^2 = \boldsymbol{x}^\top \boldsymbol{x}$
- **1-norm** $\|\boldsymbol{x}\|_1 = \sum_{i=1}^n |x_i|$
- **Max-norm** $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$

Vector and Matrix Norms

Matrix Norms

- **Frobenius Norm** If we think of a matrix as a vector, we can define the matrix norm in terms of a vector norm, $\|\mathbf{A}\| = \|\text{vec}(\mathbf{A})\|$. If the vector norm is the 2-norm, the corresponding matrix norm is the Frobenius norm:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2} = \sqrt{tr(\mathbf{A}^\top \mathbf{A})} = \|\text{vec}(\mathbf{A})\|_2$$

Here $tr(\cdot)$ denotes the trace of the matrix.

Some Key Properties of Matrices

Rank of a Matrix The **column rank** of a matrix $\mathbf{A}$ is the dimension of the space spanned by its columns, and the **row rank** is the dimension of the space spanned by its rows.

- For $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\text{rank}(\mathbf{A}) \leq \min(m, n)$. If $\text{rank}(\mathbf{A}) = \min(m, n)$, then $\mathbf{A}$ is said to be **full rank**, otherwise it is called **rank deficient**.
- For $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^\mathsf{T}) = \text{rank}(\mathbf{A}^\mathsf{T}\mathbf{A}) = \text{rank}(\mathbf{A}\mathbf{A}^\mathsf{T})$.
- For $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times p}$, $\text{rank}(\mathbf{AB}) \leq \min(\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B}))$.
- For $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$, $\text{rank}(\mathbf{A} + \mathbf{B}) \leq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B})$.

A square matrix is invertible iff* it is full rank.

* "iff" is not a typo, it means "if and only if" referring to necessary and sufficient conditions

Special Matrices

Diagonal Matrix A diagonal matrix is a matrix where all non-diagonal elements are 0. This is typically denoted by $\mathbf{D} = \text{diag}(d_1, d_2, \ldots, d_n)$, with

$$\mathbf{D} = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix}$$

Identity matrix is a square matrix with $\mathbf{I} = \text{diag}(1, 1, \ldots, 1) \in \mathbb{R}^{n \times n}$

Positive Definite Matrices A symmetric matrix (square) is positive definite iff all of its eigenvalues are positive. These are denoted by $\mathbf{A} \succ 0$. We also have **positive semi-definite** matrices where $\mathbf{A} \succcurlyeq 0$. Similarly we have negative definite ($\mathbf{A} \prec 0$) and negative semi-definite ($\mathbf{A} \preccurlyeq 0$) matrices. An example is a Gram matrix $\mathbf{G} = \mathbf{A}^\top \mathbf{A}$ which is always positive semi-definite.

Special Matrices

Orthogonal Matrices

- Two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ are **orthogonal** if $\boldsymbol{x}^\top \boldsymbol{y} = 0$.
- A vector $\boldsymbol{x} \in \mathbb{R}^n$ is **normalized** if $\|\boldsymbol{x}\|_2 = 1$
- A set of vectors that is pairwise orthogonal and normalized is called orthonormal.
- A square matrix $\mathbf{U} \in \mathbb{R}^{n \times n}$ is **orthogonal** (aka **orthonormal**) if all its columns are **orthonormal**.
- Note the difference in meaning of the term **orthogonal** for vectors and matrices.
- If the entries of a matrix $\mathbf{U}$ are complex valued, we use the term **unitary** instead of orthogonal. And for these matrices

$$\mathbf{U}^\top \mathbf{U} = \mathbf{I} = \mathbf{U}\mathbf{U}^\top$$

and

$$\mathbf{U}^\top = \mathbf{U}^{-1}$$

Vector-Vector Product

For a matrix $\boldsymbol{A}$ and $\boldsymbol{x}$, their product
In terms of inner product

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

In terms of outer product:

$$\boldsymbol{x}\boldsymbol{y}^{\mathsf{T}} \in \mathbb{R}^{m \times n} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix}$$

Matrix-Vector Product

For a matrix $\boldsymbol{A}$ and $\boldsymbol{x}$, their product
In terms of inner product

$$\boldsymbol{y} = \mathbf{A}\boldsymbol{x} = \begin{bmatrix} — & \boldsymbol{a}_1^\top & — \\ — & \boldsymbol{a}_2^\top & — \\ & \vdots & \\ — & \boldsymbol{a}_m^\top & — \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{a}_1^\top \boldsymbol{x} \\ \boldsymbol{a}_2^\top \boldsymbol{x} \\ \vdots \\ \boldsymbol{a}_m^\top \boldsymbol{x} \end{bmatrix}$$

In terms of outer product:

$$\boldsymbol{y} = \mathbf{A}\boldsymbol{x} = \begin{bmatrix} | & | & & | \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} | \\ \boldsymbol{a}_1 \\ | \end{bmatrix} x_1 + \begin{bmatrix} | \\ \boldsymbol{a}_2 \\ | \end{bmatrix} x_2 + \ldots + \begin{bmatrix} | \\ \boldsymbol{a}_n \\ | \end{bmatrix} x_n$$

Matrix-Matrix Product

For two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$, their product $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times p}$ give by

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$

In terms of inner product

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} — & \boldsymbol{a}_1^\mathsf{T} & — \\ — & \boldsymbol{a}_2^\mathsf{T} & — \\ & \vdots & \\ — & \boldsymbol{a}_m^\mathsf{T} & — \end{bmatrix} \begin{bmatrix} | & | & & | \\ \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_p \\ | & | & & | \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1^\mathsf{T}\boldsymbol{b}_1 & \boldsymbol{a}_1^\mathsf{T}\boldsymbol{b}_2 & \cdots & \boldsymbol{a}_1^\mathsf{T}\boldsymbol{b}_p \\ \boldsymbol{a}_2^\mathsf{T}\boldsymbol{b}_1 & \boldsymbol{a}_2^\mathsf{T}\boldsymbol{b}_2 & \cdots & \boldsymbol{a}_2^\mathsf{T}\boldsymbol{b}_p \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a}_m^\mathsf{T}\boldsymbol{b}_1 & \boldsymbol{a}_m^\mathsf{T}\boldsymbol{b}_2 & \cdots & \boldsymbol{a}_m^\mathsf{T}\boldsymbol{b}_p \end{bmatrix}$$

In terms of outer product:

$$\mathbf{C} = \mathbf{AB} = \begin{bmatrix} | & | & & | \\ \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} — & \boldsymbol{b}_1^\mathsf{T} & — \\ — & \boldsymbol{b}_2^\mathsf{T} & — \\ & \vdots & \\ — & \boldsymbol{b}_n^\mathsf{T} & — \end{bmatrix} = \sum_{i=1}^{n} \boldsymbol{a}_i \boldsymbol{b}_i^\mathsf{T}$$

Matrix-Matrix Product

Properties of matrix multiplication.

Matrix multiplication is

- Associative: $(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$
- Distributive: $\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{AB} + \mathbf{AC}$

BUT NOT

- Commutative: $\mathbf{AB} \neq \mathbf{BA}$

Inverse of a Matrix

The inverse of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is denoted $\mathbf{A}^{-1}$, and is the unique matrix such that

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} = \mathbf{A}\mathbf{A}^{-1}$$

Note that $\mathbf{A}^{-1}$ exists if and only if $\det(\mathbf{A}) \neq 0$. If $\det(\mathbf{A}) = 0$, it is called a **singular matrix**. The following are properties of the inverse; all assume that $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ are non-singular:

$$(\mathbf{A}^{-1})^{-1} = \mathbf{A}$$
$$(\mathbf{A}\mathbf{B})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$$
$$(\mathbf{A}^{-1})^{\mathsf{T}} = (\mathbf{A}^{\mathsf{T}})^{-1} \triangleq \mathbf{A}^{-T}$$

Eigenvalue decomposition (EVD)

Review of some standard material on the eigenvalue decomposition or EVD of square (real-valued) matrices.

Given a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$,we say that $\lambda \in \mathbb{R}$ is an eigenvalue of $\mathbf{A}$ and $\boldsymbol{u} \in \mathbb{R}^n$ is the corresponding eigenvector if

$$\mathbf{A}\boldsymbol{u} = \lambda \boldsymbol{u}, \quad \boldsymbol{u} \neq 0$$

The **characteristic equation** is given by

$$\det(\lambda \mathbf{I} - \mathbf{A}) = 0$$

Eigenvalue decomposition (EVD)

Diagonalization We can write all the eigenvector equations simultaneously as

$$\mathbf{AU} = \mathbf{U\Lambda}$$

where the columns of $\mathbf{U} \in \mathbb{R}^{n \times n}$ are the eigenvectors of $\mathbf{A}$ and $\mathbf{\Lambda}$ is a diagonal matrix whose entries are the eigenvalues of $\mathbf{A}$, i.e.,

$$\mathbf{U} \in \mathbb{R}^{n \times n} = \begin{bmatrix} | & | & & | \\ \boldsymbol{u}_1 & \boldsymbol{u}_2 & \cdots & \boldsymbol{u}_n \\ | & | & & | \end{bmatrix}, \quad \mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$$

If the eigenvectors of $\mathbf{A}$ are linearly independent, then the matrix $\mathbf{U}$ will be invertible, so we can write

$$\mathbf{A} = \mathbf{U\Lambda U}^{-1}$$

A matrix that can be written in this form is called **diagonalizable**.

Singular value decomposition (SVD)

SVD generalizes EVD to rectangular matrices. Any (real) matrix $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ can be decomposed as

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathsf{T}} = \sigma_1 \begin{pmatrix} | \\ \boldsymbol{u}_1 \\ | \end{pmatrix} \begin{pmatrix} - & \boldsymbol{v}_1^{\mathsf{T}} & - \end{pmatrix} + \cdots + \sigma_r \begin{pmatrix} | \\ \boldsymbol{u}_r \\ | \end{pmatrix} \begin{pmatrix} - & \boldsymbol{v}_r^{\mathsf{T}} & - \end{pmatrix}$$

where

- $\boldsymbol{U} \in \mathbb{R}^{m \times m}$ is an orthonormal matrix ($\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{I}_m$)
- $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ is an orthonormal matrix ($\boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^\top \boldsymbol{I}_n$)
- $\boldsymbol{S} \in \mathbb{R}^{m \times n}$ contains $r = \min(m, n)$ singular values $\sigma_i \geq 0$ on the main diagonal, 0s filling the rest of the matrix

Singular value decomposition (SVD)

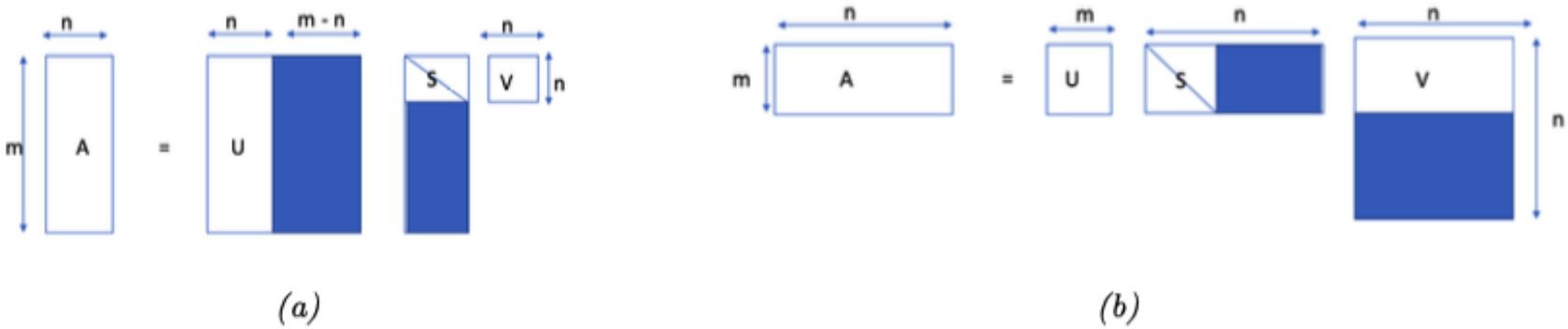


Figure 7.8: SVD decomposition of a matrix, $\mathbf{A} = \mathbf{USV}^{\mathsf{T}}$. The shaded parts of each matrix are not computed in the economy-sized version. (a) Tall skinny matrix. (b) Short wide matrix.

Gradients

Consider a function that maps a vector to a scalar, $f : \mathbb{R}^n \to \mathbb{R}$. The gradient of a function at a point $\boldsymbol{x}$ is the vector of its partial derivatives:

$$\boldsymbol{g} = \frac{\partial f}{\partial \boldsymbol{x}} = \nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

To emphasize the point at which the gradient is evaluated, we can write

$$\boldsymbol{g}(\boldsymbol{x}^*) \triangleq \left. \frac{\partial f}{\partial \boldsymbol{x}} \right|_{\boldsymbol{x}^*}$$

- Operator "∇" (*nabla*) maps a function f to another function $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^n$.
- Since $\boldsymbol{g}$ is a vector valued function, it is known as a **vector field**.

Directional Derivative

The directional derivative measures how much the function $f : \mathbb{R}^n \to \mathbb{R}$ changes along a direction $\boldsymbol{v}$ in space. It is defined as

$$D_{\boldsymbol{v}} f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) \cdot \boldsymbol{v}$$

"$\cdot$" denotes the dot product or the inner product.

Jacobian

Consider a function that maps a vector to another vector, $f : \mathbb{R}^n \to \mathbb{R}^m$. The Jacobian matrix of this function is a $m \times n$ matrix of partial derivatives:

$$\mathbf{J}_{\boldsymbol{f}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}^{\mathsf{T}}} \triangleq \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \nabla f_1(\boldsymbol{x})^{\mathsf{T}} \\ \vdots \\ \nabla f_m(\boldsymbol{x})^{\mathsf{T}} \end{pmatrix}$$

Note that *here* we lay out the results in the same orientation as the output of f; this is sometimes called **numerator layout** or the Jacobian formulation.

Hessian

Consider a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, the Hessian matrix is a symmetric $n \times n$ matrix of second partial derivatives:

$$\mathbf{H}_f = \frac{\partial^2 f}{\partial \boldsymbol{x}^2} = \nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ & \vdots & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

We see that the Hessian is the Jacobian of the gradient.

Functions that map scalars to scalars

Consider a differentiable function $f : \mathbb{R} \to \mathbb{R}$

$$\frac{d}{dx}\left[f(x) + g(x)\right] = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$$

$$\frac{d}{dx}\left[f(x)g(x)\right] = f(x)\frac{dg(x)}{dx} + g(x)\frac{df(x)}{dx}$$

$$\frac{d}{dx}f(u(x)) = \frac{du}{dx}\frac{df(u)}{du}$$

The last relationship here is known as the **chain rule of calculus**. This is important for backpropagation algorithm to train neural networks.

Product Rule for Vectors

$$\frac{\partial\ u(\boldsymbol{x})^\top v(\boldsymbol{x})}{\partial \boldsymbol{x}} = \left[\frac{\partial v(\boldsymbol{x})}{\partial \boldsymbol{x}}\right] u(\boldsymbol{x}) + \left[\frac{\partial u(\boldsymbol{x})}{\partial \boldsymbol{x}}\right] v(\boldsymbol{x})$$

Useful Relations

Consider a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$

$$\frac{\partial(\boldsymbol{a}^\mathsf{T}\boldsymbol{x})}{\partial \boldsymbol{x}} = \boldsymbol{a}$$

$$\frac{\partial(\boldsymbol{b}^\mathsf{T}\mathbf{A}\boldsymbol{x})}{\partial \boldsymbol{x}} = \mathbf{A}^\mathsf{T}\boldsymbol{b}$$

$$\frac{\partial(\boldsymbol{x}^\mathsf{T}\mathbf{A}\boldsymbol{x})}{\partial \boldsymbol{x}} = (\mathbf{A} + \mathbf{A}^\mathsf{T})\boldsymbol{x}$$

Useful Relations

Consider a differentiable function $f : \mathbb{R}^{m \times n} \to \mathbb{R}$

$$\frac{\partial f}{\partial \mathbf{X}} = \begin{pmatrix} \frac{\partial f}{\partial x_{11}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ & \vdots & \\ \frac{\partial f}{\partial x_{m1}} & \cdots & \frac{\partial f}{\partial x_{mn}} \end{pmatrix}$$

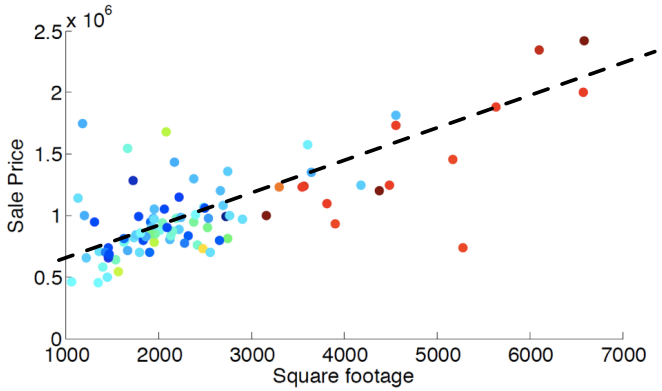
Outline

How do we fit a model to a given dataset?

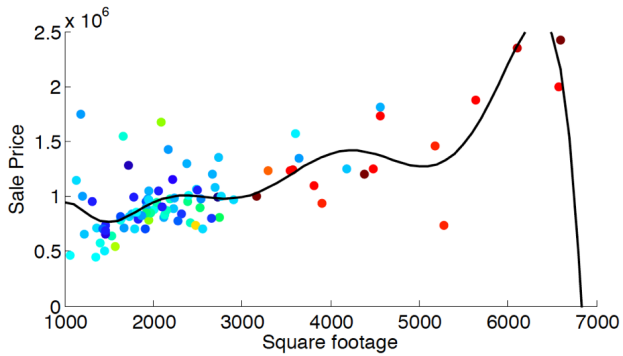
- The core problem in machine learning is parameter estimation.
- This requires solving an **optimization problem**, where we try to find the values for a set of variables $\boldsymbol{\theta} \in \Theta$, that **minimize** a scalar-valued **loss function or cost function** $\mathcal{L} : \boldsymbol{\theta} \to \mathbb{R}$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \, \mathcal{L}(\boldsymbol{\theta})$$

- The parameter space $\Theta \subset \mathbb{R}^D$, where D is the number of variables. Thus we are interested in **continuous optimization**.



Fitting a line (linear model) to Data.



Fitting a polynomial (degree > 1)

Optimization

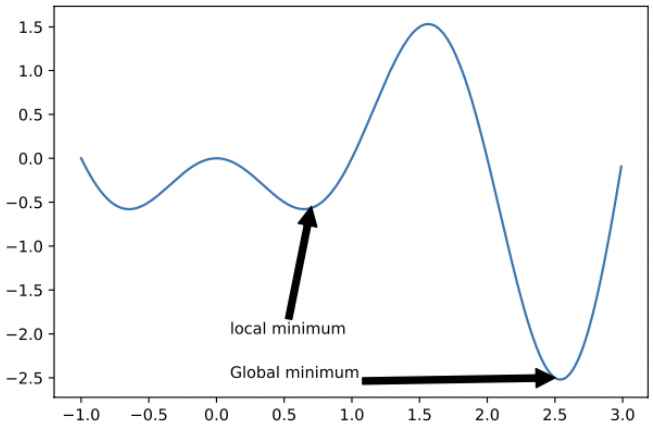
Definition: Optimization refers to choosing the best element from some set of available alternatives. A general form is as follows:

$$\begin{aligned} \text{minimize} \quad & \mathcal{L}(\boldsymbol{\theta}) \\ \text{subject to} \quad & f_i(\boldsymbol{\theta}) \leq 0, i = 1, \ldots, m \\ & h_i(\boldsymbol{\theta}) = 0, i = 1, \ldots, p. \end{aligned} \tag{1}$$

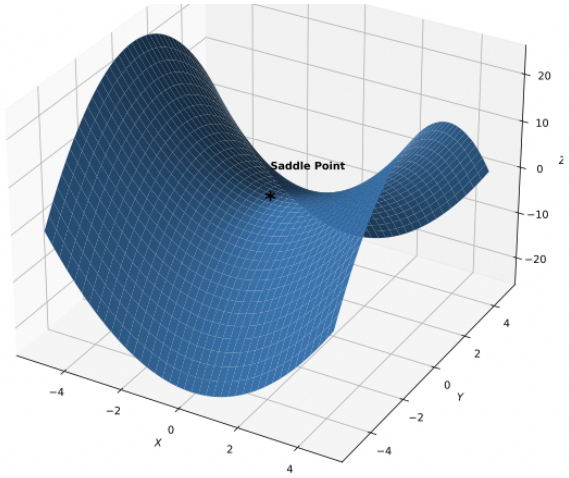
Difficulties:

1. Local or global optimum?
2. Difficulty to find a feasible point,
3. Stopping criteria,
4. Poor convergence rate,
5. Numerical issues

Local and Global Optimization



(a)



(b)

Figure 8.1: (a) Illustration of local and global minimum in 1d. Generated by extrema_fig_1d.ipynb. (b) Illustration of a saddle point in 2d. Generated by saddle.ipynb.

Local and Global Optimization

- A point that satisfies the equation below is called a global optimum. Finding such a point is called global optimization

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \, \mathcal{L}(\boldsymbol{\theta})$$

- This is in general intractable, and we just try to find a **local optimum**. For continuous problems, this is defined to be a point $\boldsymbol{\theta}^*$ which has lower (or equal) cost than "nearby" points.
- A local minimum could be surrounded by other local minima with the same objective value; this is known as a **flat local minimum**.
- A point is said to be a **strict local minimum** if its cost is strictly lower than those of neighboring points.

Local and Global Optimization

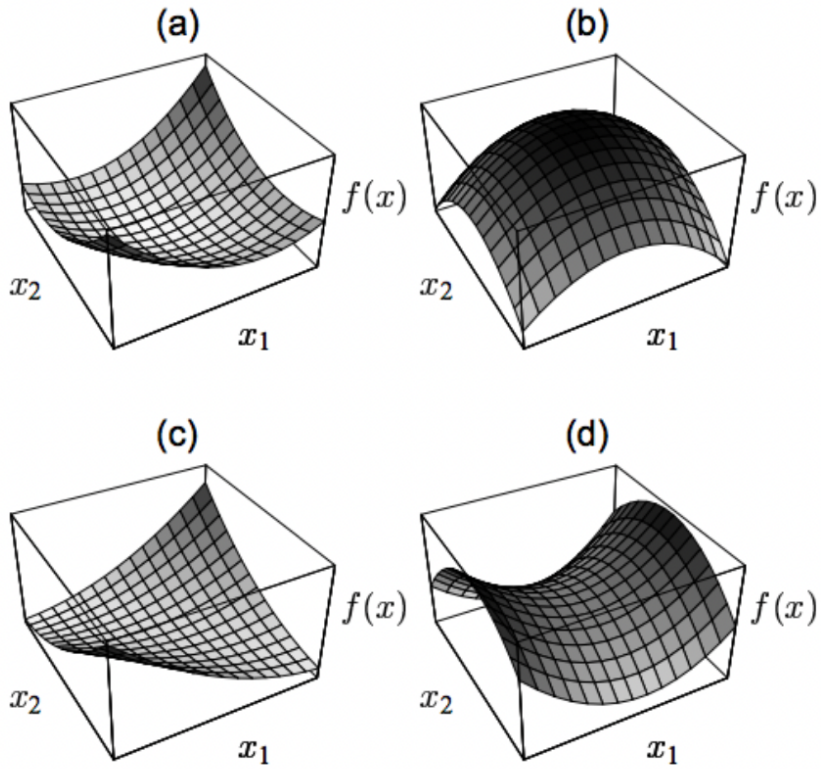


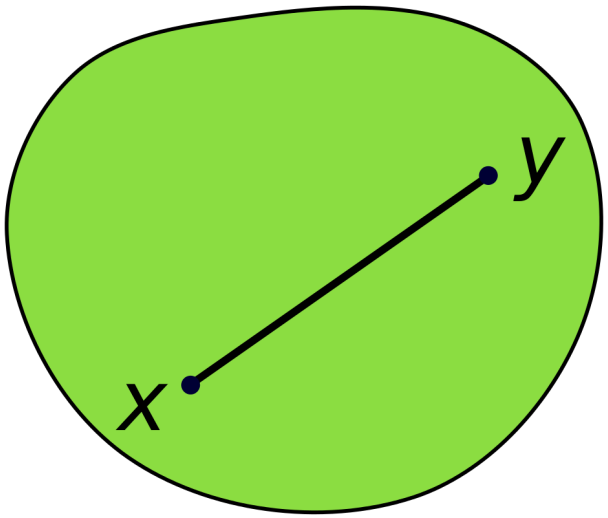
Figure 8.6: The quadratic form $f(\boldsymbol{x}) = \boldsymbol{x}^{\mathsf{T}}\mathbf{A}\boldsymbol{x}$ in 2d. (a) $\mathbf{A}$ is positive definite, so f is convex. (b) $\mathbf{A}$ is negative definite, so f is concave. (c) $\mathbf{A}$ is positive semidefinite but singular, so f is convex, but not strictly. Notice the valley of constant height in the middle. (d) $\mathbf{A}$ is indefinite, so f is neither convex nor concave. The stationary point in the middle of the surface is a saddle point. From Figure 5 of [She94].

Optimality Conditions for local vs global optima

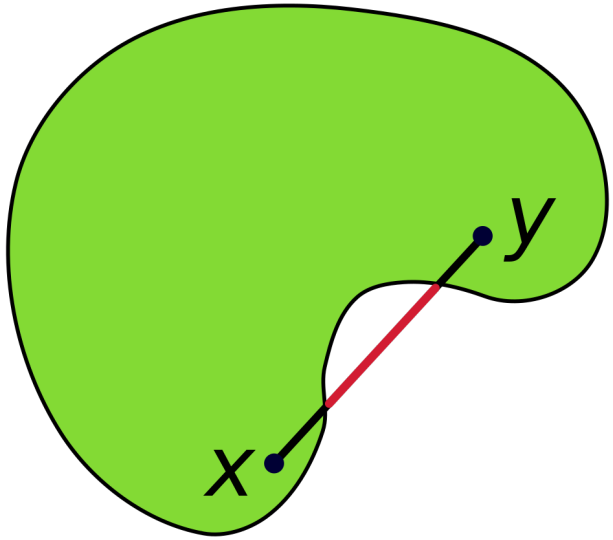
- For continuous, twice differentiable functions, we can precisely characterize the points which correspond to local minima.
- Let $\boldsymbol{g}(\boldsymbol{\theta}) = \nabla \mathcal{L}(\boldsymbol{\theta})$ be the gradient vector, and $\mathbf{H}(\boldsymbol{\theta}) = \nabla \mathcal{L}(\boldsymbol{\theta})$ be the Hessian matrix.
- Consider a point $\boldsymbol{\theta}^* \in \mathbb{R}^D$, and let $\boldsymbol{g}^* = \boldsymbol{g}(\boldsymbol{\theta})|_{\boldsymbol{\theta}^*}$ be the gradient at that point, and $\mathbf{H}^* = \mathbf{H}(\boldsymbol{\theta})|_{\boldsymbol{\theta}^*}$ be the corresponding Hessian.
- One can show that the following conditions characterize every local minimum:
  - **Necessary condition**: If $\boldsymbol{\theta}^*$ is a local minimum, then we must have $\boldsymbol{g}^* = 0$ (i.e., $\boldsymbol{\theta}^*$ must be a stationary point), and $\mathbf{H}^*$ must be positive semi-definite.
  - **Sufficient condition**: If $\boldsymbol{g}^* = 0$ and $\mathbf{H}^*$ is positive definite, then $\boldsymbol{\theta}^*$ is a local optimum.

Convex Sets and Functions

Convex Set A set S is convex iff for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$, $\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y} \in S$ for any $\lambda \in [0, 1]$,



Convex Polygon



Non-Convex Polygon

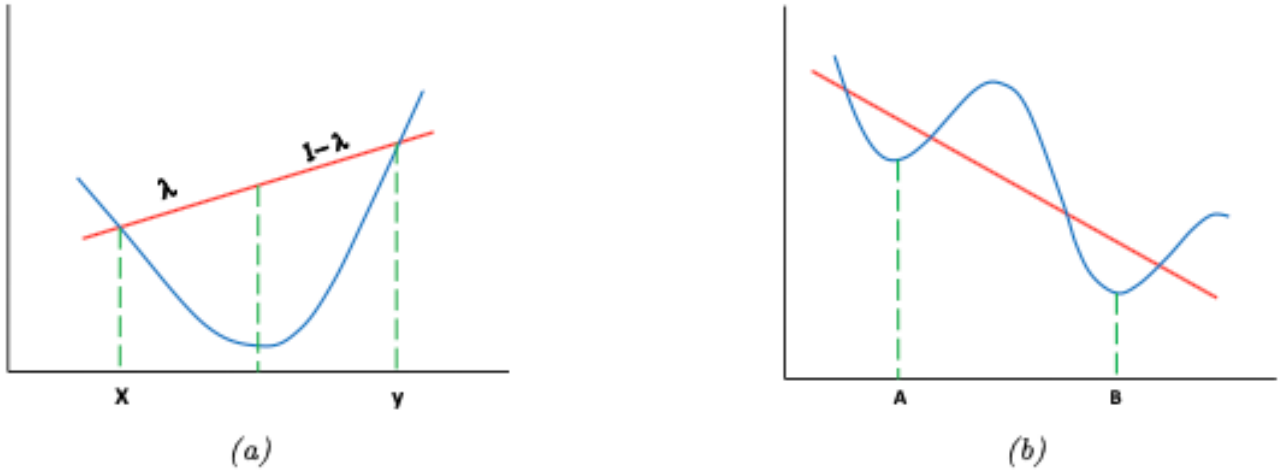
Convex Sets and Functions

Convex Functions: Function $f(\cdot)$ is a convex function, if for a convex set $\mathcal{S}$, any two points $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$ and any $\lambda \in [0, 1]$ satisfy

$$f(\lambda \boldsymbol{x} + (1 - \lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1 - \lambda) f(\boldsymbol{y}).$$

Concave Functions A function f is said to be *concave* if $-f$ is convex.

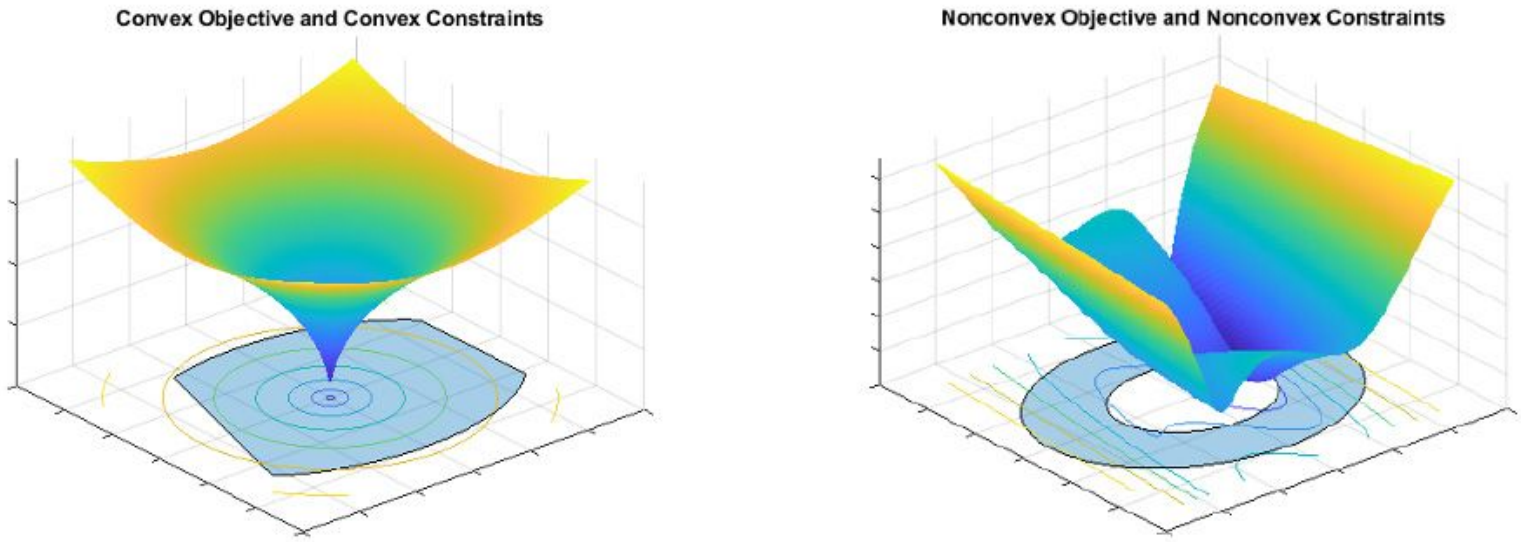
Non-convex Functions A function which is neither convex nor concave.



(a) (b)

*Figure 8.5: (a) Illustration of a convex function. We see that the chord joining $(x, f(x))$ to $(y, f(y))$ lies above the function. (b) A function that is neither convex nor concave. **A** is a local minimum, **B** is a global minimum.*

Convex Sets and Functions



Convex and Non-convex Objectives (image source)

Convex Optimization

Convex Optimization is minimization (maximization) of a convex (concave) function over a convex set, e.g., Linear Programming (LP), Quadratic Programming (QP), and Semi-Definite Programming (SDP).

A number of popular convex optimization algorithms:

- Gradient descent
- Conjugate gradient
- Newton's method
- Quasi-Newton method
- Subgradient method

How do we know if a function f is convex?

Theorem 8.1.1. *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable over its domain. Then f is convex iff $\mathbf{H} = \nabla^2 f(\boldsymbol{x})$ is positive semi definite (Section 7.1.5.3) for all $\boldsymbol{x} \in \mathrm{dom}(f)$. Furthermore, f is strictly convex if $\mathbf{H}$ is positive definite.*

For example, consider the quadratic form

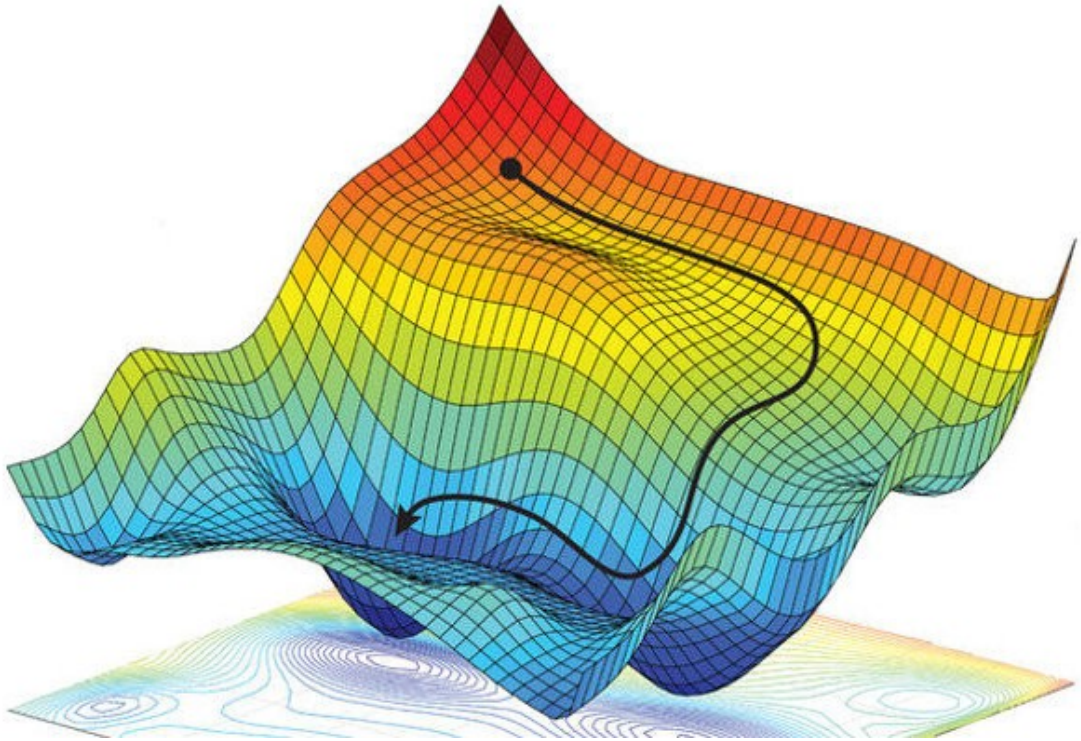
$$f(\boldsymbol{x}) = \boldsymbol{x}^\top \mathbf{A} \boldsymbol{x}$$

This is

- **convex** if $\mathbf{A}$ is *positive semi definite*, and
- **strictly convex** if $\mathbf{A}$ is *positive definite*
- It is neither convex nor concave if $\mathbf{A}$ has eigenvalues of mixed sign.

Deep learning involves non-convex optimization

BUT it is extremely hard to get any type of guarantees for non-convex optimizations. Therefore, convex optimization is still very popular and practical in a wide range of domains!



A non-convex function (image source)

Gradient and Stochastic Gradient Descent

We describe two simple yet extremely popular methods

- **Gradient Descent (GD)**: simple and fundamental
- **Stochastic Gradient Descent (SGD)**: faster, effective for large-scale problems

Gradient is sometimes referred to as *first-order* information of a function. Therefore, these methods are called *first-order methods*.

Gradient Descent (GD)

Goal: minimize a function $F(\boldsymbol{w})$. For instance, the loss function with the choice of $\boldsymbol{w}$

Algorithm: keep moving in the *negative gradient direction*

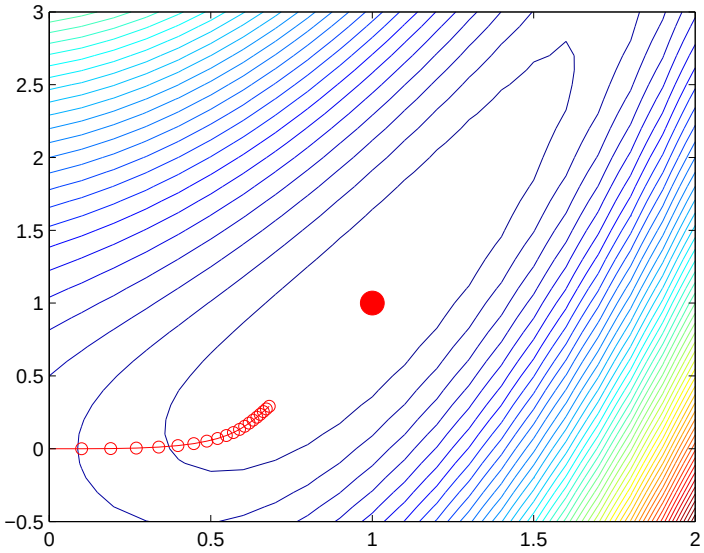
Start from some $\boldsymbol{w}^{(0)}$. For $t = 0, 1, 2, \ldots$

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \nabla F(\boldsymbol{w}^{(t)})$$

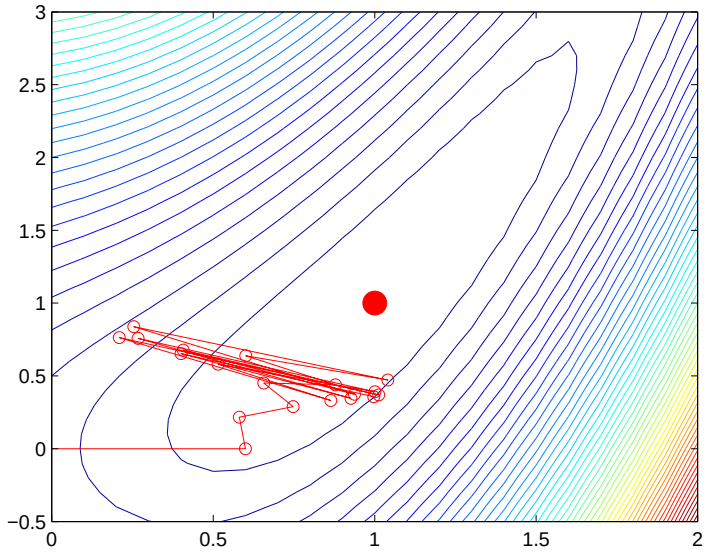
where $\eta > 0$ is called step size or learning rate

- in theory η should be set in terms of some parameters of F
- in practice we just try several small values

What does GD look like?



reasonable η decreases function value



but large η is unstable

Slow: each update makes one pass of the entire training set!

Stochastic Gradient Descent (SGD)

GD: keep moving in the negative gradient direction

SGD: keep moving in some *noisy* negative gradient direction

$$\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta \tilde{\nabla} F(\boldsymbol{w}^{(t)})$$

where $\tilde{\nabla} F(\boldsymbol{w}^{(t)})$ is a random variable (called **stochastic gradient**) s.t.

$$\mathbb{E}\left[\tilde{\nabla} F(\boldsymbol{w}^{(t)})\right] = \nabla F(\boldsymbol{w}^{(t)}) \qquad \text{(unbiasedness)}$$

The gradient calculation uses only a sample, is therefore "noisy", variants use a mini-batch of samples.

Key point: it could be *much faster to obtain a stochastic gradient*!

Convergence Guarantees

Many for both GD and SGD on convex objectives.

They tell you at most how many iterations you need to achieve

$$F(\boldsymbol{w}^{(t)}) - F(\boldsymbol{w}^*) \leq \epsilon$$

Even for *non-convex objectives* such as those encountered in deep learning, many recent works show effectiveness of GD/SGD.

Convergence also depends on the Condition Number κ

Condition number for a matrix $\mathbf{A}$ is defined as

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

Condition number of the Hessian is a great indicator of how the landscape of the function (to be optimized) looks like. The closer to 1 the more *well conditioned* the matrix.

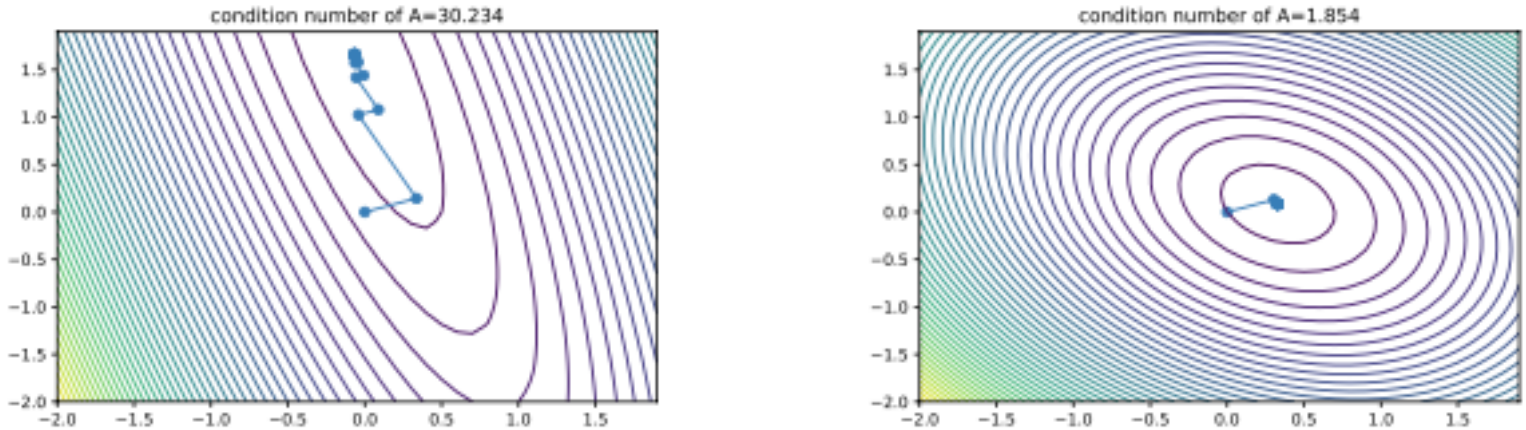


Figure 8.12: Illustration of the effect of condition number κ on the convergence speed of steepest descent with exact line searches. (a) Large κ. (b) Small κ. Generated by lineSearchConditionNum.ipynb.

Reading

[PiML 1]

- 7.1.2.1, 7.1.2.1, 7.1.2.4, 7.1.3, 7.1.4.1, 7.1.4.2, 7.1.4.3, 7.1.5.1, 7.1.5.3, 7.1.5.4
- 7.2.1-3
- 7.3.1
- 7.4.1-2
- 7.5.1
- 7.8.2, 7.8.3, 7.8.5 (intro), 7.8.6, 7.8.7
- 8.1 (introduction), 8.1.1 (introduction), 8.1.1.1, 8.1.2 (material from slides only), 8.1.3 (8.1.3.1-8.1.3.3)

For the subsections, results covered in the slides are important, the rest will be covered on-need basis.

References I