EE2211 Introduction to Machine Learning

Lecture 4

Semester 1
2025/2026

more linear algebra

Helen Juan Zhou

helen.zhou@nus.edu.sg

Department of Electrical and Computer Engineering
National University of Singapore

Acknowledgement:
EE2211 development team
(Kar-Ann Toh, Xinchao Wang, Thomas Yeo, Helen Zhou, Chen Khong, Vincent Tan, Robby Tan and Haizhou Li)

Course Contents

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Probability, Statistics, and Matrix
- Fundamental Machine Learning Algorithms I (Helen)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Helen)
  - Over-fitting, bias/variance trade-off
  - Optimization, Gradient descent
  - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
  - Performance Issues
  - K-means Clustering
  - Neural Networks

Systems of Linear Equations

Module II Contents

- Operations on Vectors and Matrices
- Systems of Linear Equations
- Set and Functions
- Derivative and Gradient
- Least Squares, Linear Regression
- Linear Regression with Multiple Outputs
- Linear Regression for Classification
- Ridge Regression
- Polynomial Regression

Recap on Notations, Vectors, Matrices

Scalar	Numerical value	15, -3.5
Variable	Take scalar values	*x* or *a*
Vector	An ordered list of scalar values	$\mathbf{x}$ or $\mathbf{a}$
	Attributes of a vector	$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$
Matrix	A rectangular array of numbers arranged in rows and columns	$\mathbf{X} = \begin{bmatrix} 2 & 4 \\ 21 & -6 \end{bmatrix}$

Capital Sigma $\sum_{i=1}^{m} x_i = x_1 + x_2 + \ldots + x_{m-1} + x_m$

Capital Pi $\prod_{i=1}^{m} x_i = x_1 \cdot x_2 \cdot \ldots \cdot x_{m-1} \cdot x_m$

Operations on Vectors and Matrices

Operations on Vectors: summation and subtraction

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \end{bmatrix}$$

$$\mathbf{x} - \mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \end{bmatrix}$$

Operations on Vectors and Matrices

Operations on Vectors: scalar

$$a\,\mathbf{x} = a\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} ax_1 \\ ax_2 \end{bmatrix}$$

$$\frac{1}{a}\mathbf{x} = \frac{1}{a}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{a}x_1 \\ \frac{1}{a}x_2 \end{bmatrix}$$

Operations on Vectors and Matrices

Matrix or Vector Transpose:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{x}^T = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} \\ x_{2,1} & x_{2,2} & x_{2,3} \\ x_{3,1} & x_{3,2} & x_{3,3} \end{bmatrix}, \quad \mathbf{X}^T = \begin{bmatrix} x_{1,1} & x_{2,1} & x_{3,1} \\ x_{1,2} & x_{2,2} & x_{3,2} \\ x_{1,3} & x_{2,3} & x_{3,3} \end{bmatrix}$$

Python demo 1

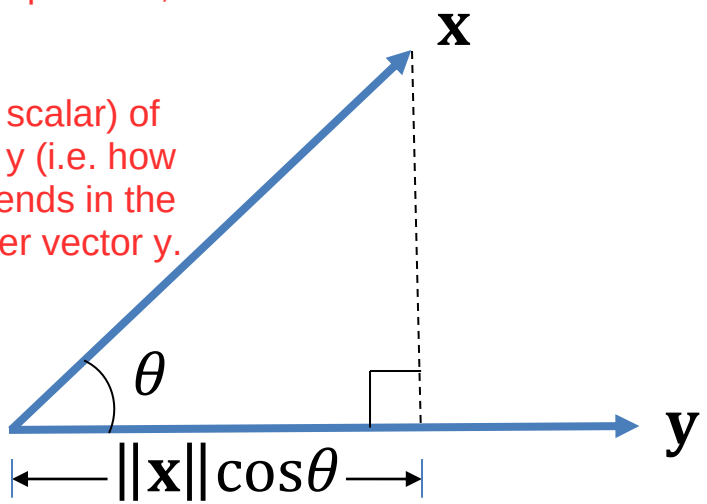
Operations on Vectors and Matrices

Dot Product or Inner Product of Vectors:

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^T \mathbf{y}$$
$$= [x_1 \quad x_2] \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$
$$= x_1\, y_1 + x_2 y_2$$

Resolve x into components, one in direction of y.

length (hence scalar) of projection of x on y (i.e. how much vector xextends in the direction of another vector y.



Geometric definition:

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\|\, \|\mathbf{y}\| \cos\theta$$

where θ is the angle between $\mathbf{x}$ and $\mathbf{y}$,
and $\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}}$ is the Euclidean length of vector $\mathbf{x}$

E. g. $\mathbf{a} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ➔ $\mathbf{a} \cdot \mathbf{c}$ = 2*1 + 3 *0 = 2

Operations on Vectors and Matrices

Matrix-Vector Product

row times column

gives a column vector

- if W * x is of (m x n) * (n x1)
- result is a (m x 1) column vector

$$\mathbf{Wx} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 + w_{1,3}x_3 \\ w_{2,1}x_1 + w_{2,2}x_2 + w_{2,3}x_3 \end{bmatrix}$$

Operations on Vectors and Matrices

Vector-Matrix Product

$$\mathbf{x}^T\mathbf{W} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \end{bmatrix}$$

$$= \begin{bmatrix} (x_1 w_{1,1} + x_2 w_{2,1}) & (x_1 w_{1,2} + x_2 w_{2,2}) & (x_1 w_{1,3} + x_2 w_{2,3}) \end{bmatrix}$$

- if x^TW is of (1 x m) * (m x n)
- result is a (1 x n) row vector

gives a row vector

Operations on Vectors and Matrices

Matrix-Matrix Product

still row times column

$$\mathbf{XW} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,d} \end{bmatrix} \begin{bmatrix} w_{1,1} & \cdots & w_{1,h} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,h} \end{bmatrix}$$

$$= \begin{bmatrix} (x_{1,1}w_{1,1} + \cdots + x_{1,d}w_{d,1}) & \cdots & (x_{1,1}w_{1,h} + \cdots + x_{1,d}w_{d,h}) \\ \vdots & \ddots & \vdots \\ (x_{m,1}w_{1,1} + \cdots + x_{m,d}w_{d,1}) & \cdots & (x_{m,1}w_{1,h} + \cdots + x_{m,d}w_{d,h}) \end{bmatrix}$$

$$= \begin{bmatrix} \sum_{i=1}^{d} x_{1,i}w_{i,1} & \cdots & \sum_{i=1}^{d} x_{1,i}w_{i,h} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{d} x_{m,i}w_{i,1} & \cdots & \sum_{i=1}^{d} x_{m,i}w_{i,h} \end{bmatrix}$$

- product will be outer rainbow (row 1 x col 2)
- can multiply only if inner rainbow col 1 == row 2

If **X** is *m* x *d* and **W** is *d* x *h, th*en the outcome is a *m* x *h* matrix

Operations on Vectors and Matrices

Matrix inverse

Definition:

A *d-by-d* square matrix **A** is **invertible** (also **nonsingular**)

if there exists a *d-by-d* square matrix **B** such that

$\mathbf{AB} = \mathbf{BA} = \mathbf{I}$ (identity matrix)

refer to equivalent statements

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$ *d-by-d* dimension

Ref: https://en.wikipedia.org/wiki/Invertible_matrix

Operations on Vectors and Matrices

Matrix inverse computation

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \operatorname{adj}(\mathbf{A})$$

- $\det(\mathbf{A})$ is the determinant of $\mathbf{A}$
- $\operatorname{adj}(\mathbf{A})$ is the adjugate or adjoint of $\mathbf{A}$

Determinant computation

transpose of cofactor matrix of A = `C'`

Example: 2x2 matrix

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\det(\mathbf{A}) = |\mathbf{A}| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Ref: https://en.wikipedia.org/wiki/Invertible_matrix

Operations on Vectors and Matrices

- $\mathrm{adj}(\mathbf{A})$ is the **adjugate or adjoint of** $\mathbf{A}$
- $\mathrm{adj}(\mathbf{A})$ is the transpose of the **cofactor matrix** $\mathbf{C}$ of $\mathbf{A}$ → $\mathrm{adj}(\mathbf{A}) = \mathbf{C}^{\mathrm{T}}$
- **Minor** of an element in a matrix $\mathbf{A}$ is defined as the determinant obtained by deleting the row and column in which that element lies

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\text{Minor of } a_{12} \text{ is } M_{12} = \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}$$

- The (i, j) entry of the **cofactor matrix** $\mathbf{C}$ is the minor of (i, j) element times a sign factor

$$\text{Cofactor } C_{ij} = (-1)^{i+j} M_{ij}$$

- The **determinant** of $\mathbf{A}$ can also be defined by minors as

$$\det(\mathbf{A}) = \sum_{j=1}^{k} = a_{ij} C_{ij} = (-1)^{i+j} a_{ij} M_{ij}$$

- just use matlab `cofactor()`
- `C = cofactor(A) returns the matrix of cofactors of A.`
- `If A is invertible, then inv(A) = C' / det(A).`
- `C = cofactor(A, i, j) returns the cofactor of row i, column j of A.`
- the T shape thing

Ref: https://en.wikipedia.org/wiki/Invertible_matrix

Operations on Vectors and Matrices

Minor of a_{12} is $M_{12} = \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}$ $\quad \text{adj}(\mathbf{A}) = \mathbf{C}^{\mathrm{T}}$

Cofactor $C_{ij} = (-1)^{i+j} M_{ij}$ $\quad \det(\mathbf{A}) = \sum_{j=1}^{k} (-1)^{i+j} a_{ij} M_{ij}$

- E.g. $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$$\mathbf{C} = \begin{bmatrix} d & -c \\ -b & a \end{bmatrix}$$

- $\text{adj}(\mathbf{A}) = \mathbf{C}^T = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$ $\quad \det(\mathbf{A}) = |\mathbf{A}| = ad - bc$

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})} \text{adj}(\mathbf{A}) = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Ref: https://en.wikipedia.org/wiki/Invertible_matrix

Operations on Vectors and Matrices

Determinant computation $\det(\mathbf{A}) = \sum_{j=1}^{k} (-1)^{i+j} a_{ij} M_{ij}$

Example: 3x3 matrix, use the first row (i = 1)

$$|A| = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} \square & \square & \square \\ \square & e & f \\ \square & h & i \end{vmatrix} - b \begin{vmatrix} \square & \square & \square \\ d & \square & f \\ g & \square & i \end{vmatrix} + c \begin{vmatrix} \square & \square & \square \\ d & e & \square \\ g & h & \square \end{vmatrix}$$

$$= a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$= a(ei - fh) - b(di - fg) + c(dh - eg)$$

Python demo 2

Ref: https://en.wikipedia.org/wiki/Determinant

Operations on Vectors and Matrices

Consider a 3×3 matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

The minor of $a_{11} = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}$

Its cofactor matrix is

$$\mathbf{C} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix}.$$

Ref: https://en.wikipedia.org/wiki/Determinant

Operations on Vectors and Matrices

Consider a 3×3 matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

The minor of $a_{12} = \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}$

Its cofactor matrix is

$$\mathbf{C} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix}.$$

Ref: https://en.wikipedia.org/wiki/Determinant

Operations on Vectors and Matrices

Consider a 3×3 matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

The minor of $a_{13} = \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$

Its cofactor matrix is

$$\mathbf{C} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix}.$$

Ref: https://en.wikipedia.org/wiki/Determinant

Operations on Vectors and Matrices

Consider a 3×3 matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

The minor of $a_{21} = \begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix}$

Its cofactor matrix is

$$\mathbf{C} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix}.$$

Ref: https://en.wikipedia.org/wiki/Determinant

Operations on Vectors and Matrices

Consider a 3×3 matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}.$$

The minor of $a_{22} = \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix}$

Its cofactor matrix is

$$\mathbf{C} = \begin{pmatrix} +\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ +\begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & +\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{pmatrix}.$$

Ref: https://en.wikipedia.org/wiki/Determinant

Operations on Vectors and Matrices

Example

Find the cofactor matrix of $\mathbf{A}$ given that $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 1 & 0 & 6 \end{bmatrix}$.

Solution:

$$a_{11} \Rightarrow \begin{vmatrix} 4 & 5 \\ 0 & 6 \end{vmatrix} = 24, \quad a_{12} \Rightarrow -\begin{vmatrix} 0 & 5 \\ 1 & 6 \end{vmatrix} = 5, \quad a_{13} \Rightarrow \begin{vmatrix} 0 & 4 \\ 1 & 0 \end{vmatrix} = -4,$$

$$a_{21} \Rightarrow -\begin{vmatrix} 2 & 3 \\ 0 & 6 \end{vmatrix} = -12, \quad a_{22} \Rightarrow \begin{vmatrix} 1 & 3 \\ 1 & 6 \end{vmatrix} = 3, \quad a_{23} \Rightarrow -\begin{vmatrix} 1 & 2 \\ 1 & 0 \end{vmatrix} = 2,$$

$$a_{31} \Rightarrow \begin{vmatrix} 2 & 3 \\ 4 & 5 \end{vmatrix} = -2, \quad a_{32} \Rightarrow -\begin{vmatrix} 1 & 3 \\ 0 & 5 \end{vmatrix} = -5, \quad a_{33} \Rightarrow \begin{vmatrix} 1 & 2 \\ 0 & 4 \end{vmatrix} = 4,$$

The cofactor matrix C is thus $\begin{bmatrix} 24 & 5 & -4 \\ -12 & 3 & 2 \\ -2 & -5 & 4 \end{bmatrix}$.

Ref: https://www.mathwords.com/c/cofactor_matrix.htm

Systems of Linear Equations

Module II Contents

- Operations on Vectors and Matrices
- **Systems of Linear Equations**
- Set and Functions
- Derivative and Gradient
- Least Squares, Linear Regression
- Linear Regression with Multiple Outputs
- Linear Regression for Classification
- Ridge Regression
- Polynomial Regression

Systems of Linear Equations

- Consider a system of m linear equations with d variables or unknowns $w_1, \ldots, w_d$:

$$
\begin{aligned}
x_{1,1}w_1 + x_{1,2}w_2 + \cdots + x_{1,d}\, w_d &= y_1 \\
x_{2,1}w_1 + x_{2,2}w_2 + \cdots + x_{2,d}\, w_d &= y_2 \\
&\vdots \\
x_{m,1}w_1 + x_{m,2}w_2 + \cdots + x_{m,d}\, w_d &= y_m .
\end{aligned}
$$

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "**Introduction to Applied Linear Algebra**", Cambridge University Press, 2018 (Chp8.3)

Systems of Linear Equations

These equations can be written compactly in matrix-vector notation:

$$\mathbf{X}\mathbf{w} = \mathbf{y}$$

w^T X = y^T== X^T w = y
- so eq and variables in X are now swapped
where col rep eq and row rep unknowns

Where

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}.$$

col is number of features/variables
row is number of equations

y is the output () - this is the labels of output by model

Note:

- The data matrix $\mathbf{X} \in \mathcal{R}^{m \times d}$ and the target vector $\mathbf{y} \in \mathcal{R}^{m}$ are given
- The unknown vector of parameters $\mathbf{w} \in \mathcal{R}^{d}$ is to be learnt

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "**Introduction to Applied Linear Algebra**", Cambridge University Press, 2018 (Chp8.3)

Systems of Linear Equations

A set of linear equations can have no solution, one solution, or multiple solutions:

$$\mathbf{X}\mathbf{w} = \mathbf{y}$$

we are always trying to solve for w because we want to find the coefficients of the system of equations such that all the variable values in X get give y using the weights w

Where

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}.$$

$\mathbf{X}$ is Square	Even-determined	m = d	Equal number of equations and unknowns
$\mathbf{X}$ is Tall	Over-determined	m > d	More number of equations than unknowns
$\mathbf{X}$ is Wide	Under-determined	m < d	Fewer number of equations than unknowns

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "**Introduction to Applied Linear Algebra**", (Chp8.3 & 11) & [Ref 5] Tan's notes, (Chp 4)

Systems of Linear Equations

$$\mathbf{X}\mathbf{w} = \mathbf{y}, \quad \mathbf{X} \in \mathcal{R}^{m \times d}, \mathbf{w} \in \mathcal{R}^{d \times 1}, \mathbf{y} \in \mathcal{R}^{m \times 1}$$

1. Square or even-determined system: $m = d$

- Equal number of equations and unknowns, i.e., $\mathbf{X} \in \mathcal{R}^{d \times d}$
- **One unique solution** if $\mathbf{X}$ is invertible or all rows/columns of $\mathbf{X}$ are linearly independent for w
- If all rows or columns of $\mathbf{X}$ are linearly independent, then $\mathbf{X}$ is invertible.

Solution:

If $\mathbf{X}$ is invertible (or $\mathbf{X}^{-1}\mathbf{X} = \mathbf{I}$), then pre-multiply both sides by $\mathbf{X}^{-1}$

$$\mathbf{X}^{-1}\mathbf{X}\,\mathbf{w} = \mathbf{X}^{-1}\mathbf{y}$$
$$\Rightarrow \quad \hat{\mathbf{w}} = \mathbf{X}^{-1}\mathbf{y}$$

(Note: we use a *hat* on top of $\mathbf{w}$ to indicate that it is a specific point in the space of $\mathbf{w}$)

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "**Introduction to Applied Linear Algebra**", Cambridge University Press, 2018 (Chp11)

Systems of Linear Equations

Example 1

$$w_1 + w_2 = 4 \quad (1)$$
$$w_1 - 2w_2 = 1 \quad (2)$$

Two unknowns
Two equations

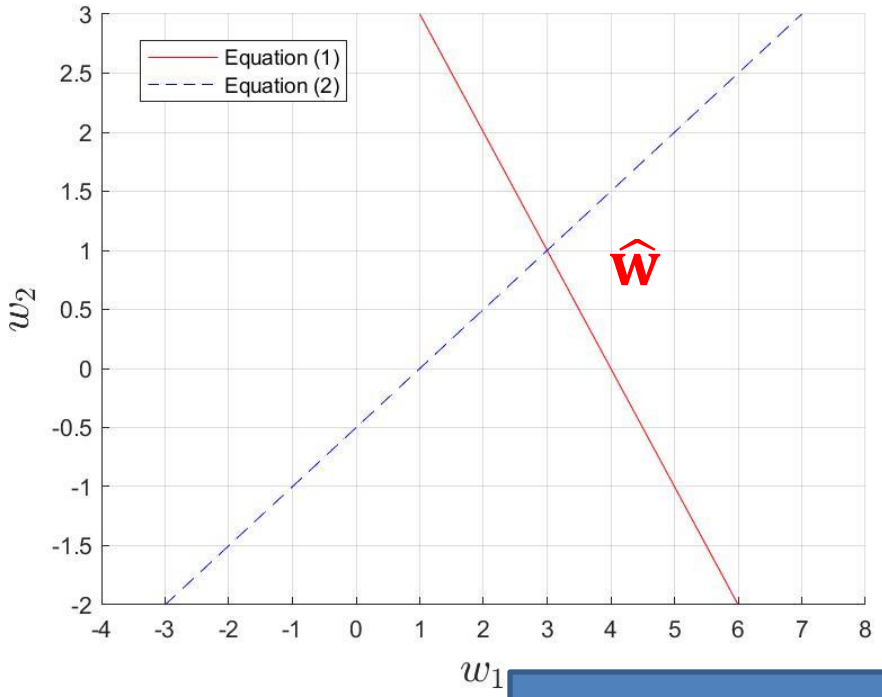
$$\mathbf{X}\quad\mathbf{w}\quad\mathbf{y}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

$$\hat{\mathbf{w}} = \mathbf{X}^{-1}\mathbf{y}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & -2 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

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



Python demo 3

$$\mathbf{A}^{-1} = \frac{1}{\det(\mathbf{A})}\mathrm{adj}(\mathbf{A}) \qquad \mathrm{adj}(\mathbf{A}) = \mathbf{C}^T = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \qquad \det(\mathbf{A}) = ad - bc$$

Systems of Linear Equations

$$\mathbf{X}\mathbf{w} = \mathbf{y}, \quad \mathbf{X} \in \mathcal{R}^{m \times d}, \mathbf{w} \in \mathcal{R}^{d \times 1}, \mathbf{y} \in \mathcal{R}^{m \times 1}$$

2. Over-determined system: $m > d$

- More equations than unknowns
- $\mathbf{X}$ is non-square (tall) and hence not invertible
- **Has no exact solution in general** *
- An **approximated solution** is available using the left inverse

this is least squares approximation

If the **left-inverse** of $\mathbf{X}$ exists such that $\mathbf{X}^{\dagger}\mathbf{X} = \mathbf{I}$, then pre-multiply both sides by $\mathbf{X}^{\dagger}$ results in

$$\mathbf{X}^{\dagger}\mathbf{X}\,\mathbf{w} = \mathbf{X}^{\dagger}\mathbf{y}$$
$$\Rightarrow \hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y}$$

substitute

Definition:

A matrix **B** that satisfies $\boldsymbol{B}_{d\,x\,m}\boldsymbol{A}_{m\,x\,d} = \mathbf{I}$ is called a **left-inverse** of $\mathbf{A}$.

The **left-inverse** of $\mathbf{X}$: $\mathbf{X}^{\dagger} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ given $\mathbf{X}^T\mathbf{X}$ is invertible.

Note: * exception: when rank($\mathbf{X}$) = rank([$\mathbf{X}$,$\mathbf{y}$]), there is a solution.

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "**Introduction to Applied Linear Algebra**", Cambridge University Press, 2018 (Chp11.1-11.2, 11.5)

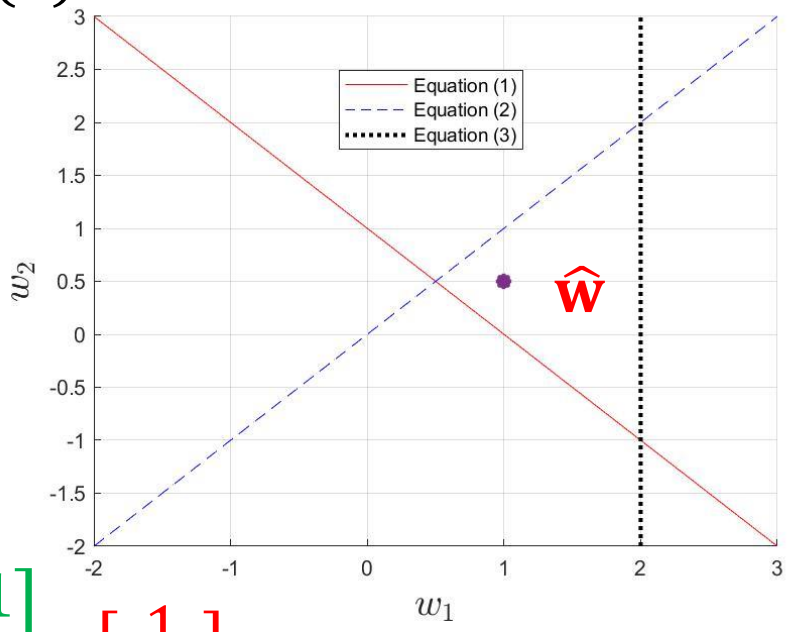
Systems of Linear Equations

Example 2

$$w_1 + w_2 = 1 \quad (1)$$
$$w_1 - w_2 = 0 \quad (2)$$
$$w_1 = 2 \quad (3)$$

Two unknowns
Three equations

$$\underset{\mathbf{X}}{\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix}} \underset{\mathbf{w}}{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}} = \underset{\mathbf{y}}{\begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}}$$



No exact solution

Approximated solution

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$= \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$

$\mathbf{X}^T\mathbf{X}$ is invertible

Python demo 4

Systems of Linear Equations

$\mathbf{X}\mathbf{w} = \mathbf{y}, \quad \mathbf{X} \in \mathcal{R}^{m \times d}, \mathbf{w} \in \mathcal{R}^{d \times 1}, \mathbf{y} \in \mathcal{R}^{m \times 1}$

3. Under-determined system: $m < d$

- More unknowns than equations
- Infinite number of solutions in general *

If the **right-inverse** of $\mathbf{X}$ exists such that $\mathbf{X}\mathbf{X}^{\dagger} = \mathbf{I}$, then the d-vector $\mathbf{w} = \mathbf{X}^{\dagger}\mathbf{y}$ (one of the infinite cases) satisfies the equation $\mathbf{X}\mathbf{w} = \mathbf{y}$, i.e.,

$$\mathbf{X}\mathbf{w} = \mathbf{y} \quad \Rightarrow \quad \mathbf{X}\mathbf{X}^{\dagger}\mathbf{y} = \mathbf{y}$$
$$\Rightarrow \quad \mathbf{I}\mathbf{y} = \mathbf{y}$$

Definition:

A matrix **B** that satisfies $\mathbf{A}_{m\,x\,d}\mathbf{B}_{d\,x\,m} = \mathbf{I}$ is called a **right-inverse** of $\mathbf{A}$.

The **right-inverse** of $\mathbf{X}$: $\mathbf{X}^{\dagger} = \mathbf{X}^{T}(\mathbf{X}\mathbf{X}^{T})^{-1}$ given $\mathbf{X}\mathbf{X}^{T}$ is invertible.

If $\mathbf{X}$ is right−invertible, we can find a unique constrained solution.

Note: * exception: no solution if the system is inconsistent rank($\mathbf{X}$) < rank([$\mathbf{X}$,$\mathbf{y}$]

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "**Introduction to Applied Linear Algebra**", Cambridge University Press, 2018 (Chp11.1-11.2, 11.5)

Systems of Linear Equations

3. Under-determined system: $m < d$

Derivation:

$\mathbf{X}\mathbf{w} = \mathbf{y}, \quad \mathbf{X} \in \mathcal{R}^{m \times d}, \mathbf{w} \in \mathcal{R}^{d \times 1}, \mathbf{y} \in \mathcal{R}^{m \times 1}$

A unique solution is yet possible by constraining the search using $\mathbf{w} = \mathbf{X}^T \mathbf{a}$

If $\mathbf{X}\mathbf{X}^T$ is invertible, let $\mathbf{w} = \mathbf{X}^T \mathbf{a}$, then

$$\mathbf{X}\mathbf{X}^T \mathbf{a} = \mathbf{y}$$

$$\Rightarrow \quad \hat{\mathbf{a}} = (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$$

$$\Rightarrow \hat{\mathbf{w}} = \mathbf{X}^T \hat{\mathbf{a}} = \mathbf{X}^T (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$$

$\mathbf{X}^\dagger$

right-inverse

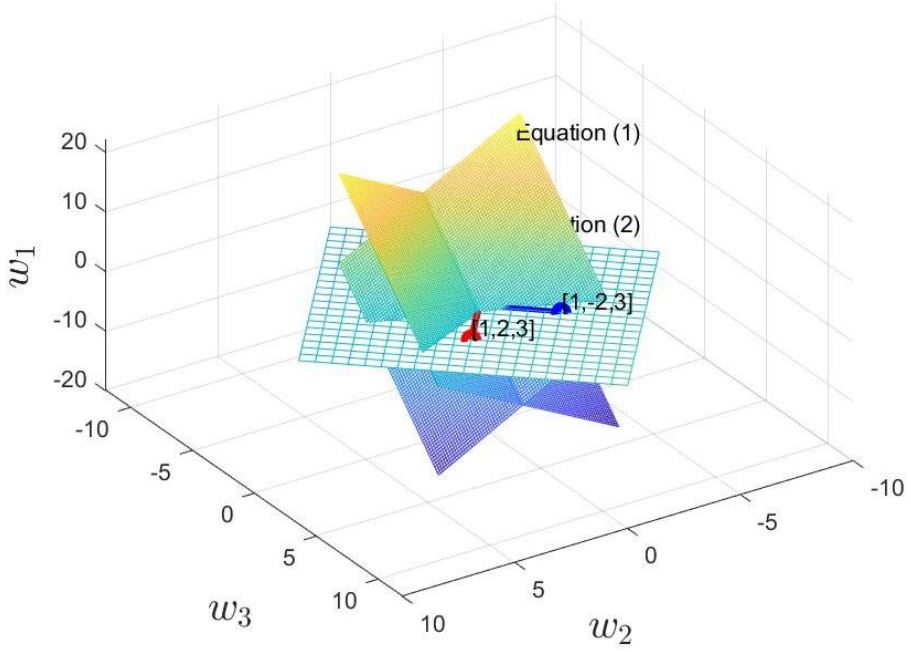
Systems of Linear Equations

Example 3

$$w_1 + 2w_2 + 3w_3 = 2 \quad (1)$$
$$w_1 - 2w_2 + 3w_3 = 1 \quad (2)$$

Three unknowns
Two equations

$$\underset{\mathbf{X}}{\begin{bmatrix} 1 & 2 & 3 \\ 1 & -2 & 3 \end{bmatrix}} \underset{\mathbf{w}}{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}} = \underset{\mathbf{y}}{\begin{bmatrix} 2 \\ 1 \end{bmatrix}}$$



Infinitely many solutions along the intersection line

Here $\mathbf{X}\mathbf{X}^T$ is invertible

$$\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$$

$$= \begin{bmatrix} 1 & 1 \\ 2 & -2 \\ 3 & 3 \end{bmatrix} \begin{bmatrix} 14 & 6 \\ 6 & 14 \end{bmatrix}^{-1} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.25 \\ 0.45 \end{bmatrix}$$

Constrained solution

Systems of Linear Equations

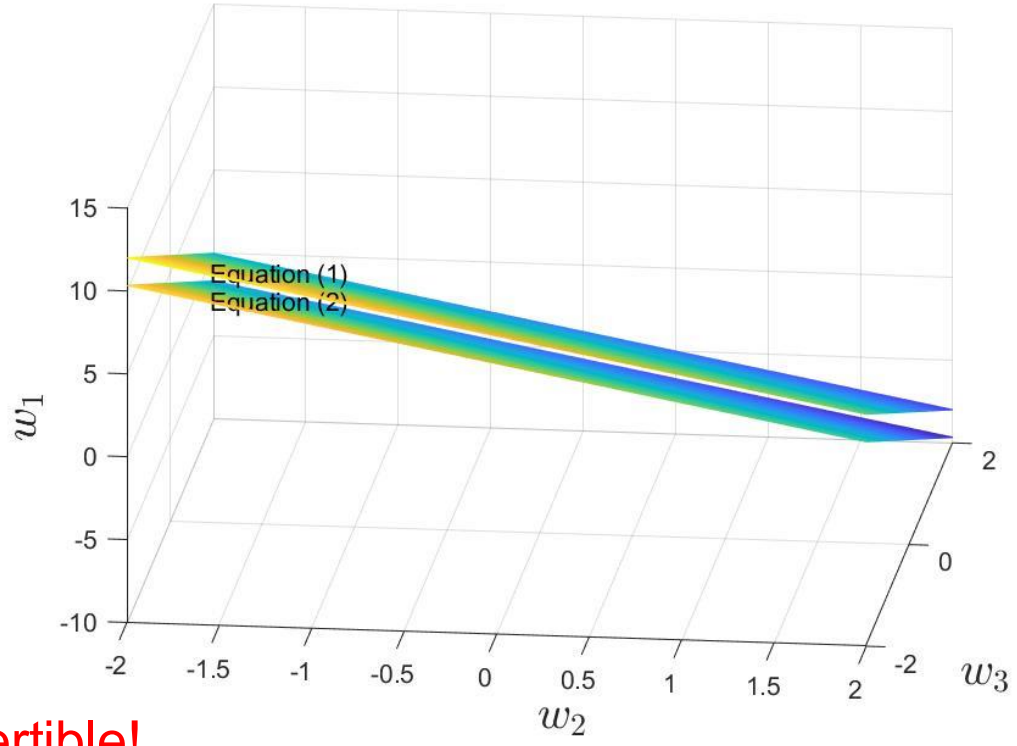
Example 4

$$w_1 + 2w_2 + 3w_3 = 2 \quad (1)$$
$$3w_1 + 6w_2 + 9w_3 = 1 \quad (2)$$

Three unknowns
Two equations

$$\mathbf{X} \qquad \mathbf{w} \qquad \mathbf{y}$$

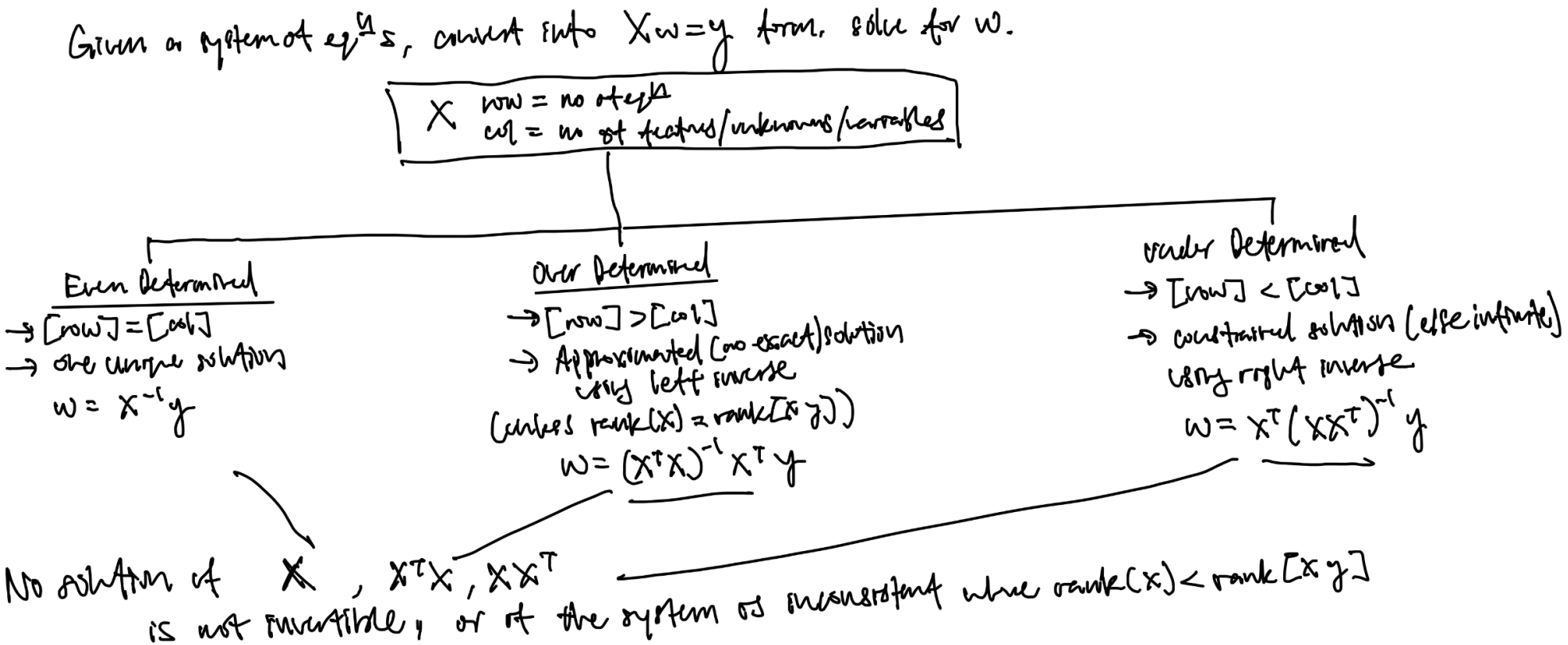
$$\begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 9 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$



Both $\mathbf{X}\mathbf{X}^T$ and $\mathbf{X}^T\mathbf{X}$ are not invertible!

There is no solution for the system

Summary


Given a system of eqns, convert into Xw=y form, solve for w.
X row = no of eqns
col = no of features/unknowns/variables
Even Determined
→ [row] = [col]
→ one unique solution
w = X⁻¹y
Over Determined
→ [row] > [col]
→ Approximated (no exact) solution
using left inverse
(unless rank(X) = rank[X y])
w = (XᵀX)⁻¹Xᵀy
Under Determined
→ [row] < [col]
→ constrained solution (else infinite)
using right inverse
w = Xᵀ(XXᵀ)⁻¹y
No solution of X, XᵀX, XXᵀ
is not invertible, or if the system of inconsistent where rank(X) < rank[X y]

Systems of Linear Equations

Module II Contents

- Operations on Vectors and Matrices
- Systems of Linear Equations
- **Set and Functions**
- Derivative and Gradient
- Least Squares, Linear Regression
- Linear Regression with Multiple Outputs
- Linear Regression for Classification
- Ridge Regression
- Polynomial Regression

Notations: Set

- A **set** is an unordered collection of unique elements
  - Denoted as a calligraphic capital character e.g., $\mathcal{S}, \mathcal{R}, \mathcal{N}$ etc
  - When an element x belongs to a set $\mathcal{S}$, we write $x \in \mathcal{S}$
- A set of numbers can be finite - include a fixed amount of values
  - Denoted using accolades, e.g. {1, 3, 18, 23, 235} or $\{x_1, x_2, x_3, x_4, \ldots, x_d\}$
- A set can be infinite and include all values in some interval
  - If a set of real numbers includes all values between *a* and *b*, including *a* and *b*, it is denoted using square brackets as [*a, b*]
  - If the set does not include the values *a* and *b*, it is denoted using parentheses as (*a, b*)
- Examples:
  - The special set denoted by $\mathcal{R}$ includes all real numbers from minus infinity to plus infinity
  - The set [0, 1] includes values like 0, 0.0001, 0.25, 0.9995, and 1.0

Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (p4 of chp2).

Notations: Set operations

- **Intersection** of two sets:

$$\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cap \mathcal{S}_2$$

Example: {1,3,5,8} ∩ {1,8,4} = {1,8}

- **Union** of two sets:

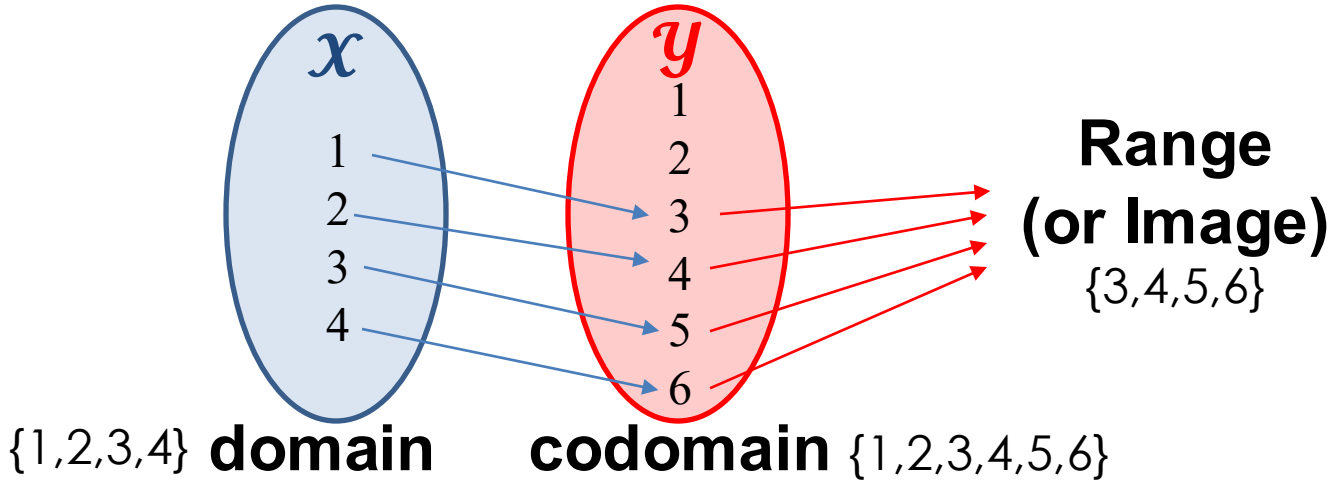
$$\mathcal{S}_3 \leftarrow \mathcal{S}_1 \cup \mathcal{S}_2$$

Example: {1,3,5,8} ∪ {1,8,4} = {1,3,4,5,8}

Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (p4 of chp2).

Functions

- A **function** is a relation that associates each element x of a set $\mathcal{X}$, the **domain** of the function, to a single element y of another set $\mathcal{Y}$, the **codomain** of the function
- If the function is called f, this relation is denoted $y = f(x)$
  - The element x is the argument or input of the function
  - y is the value of the function or the output
- The symbol used for representing the input is the variable of the function
  - $f(x)$ f is a function of the variable x; $f(x, w)$ f is a function of the variable x and w



Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (p6 of chp2).

Functions

- A scalar function can have vector argument
  - E.g. $y = f(\mathbf{x}) = x_1 + x_2 + 2x_3$
- A vector function, denoted as $\mathbf{y} = \mathbf{f}(\mathbf{x})$ is a function that returns a vector $\mathbf{y}$
  - Input argument can be a **vector** $\mathbf{y} = \mathbf{f}(\mathbf{x})$ or a **scalar** $\mathbf{y} = \mathbf{f}(x)$
  - E.g. $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -x_1 \\ x_2 \end{bmatrix}$
  - E.g. $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -2x_1 \\ 3x_1 \end{bmatrix}$

Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (p7 of chp2).

Functions

- The notation $f: \mathcal{R}^d \rightarrow \mathcal{R}$ means that f is a function that maps real d-vectors to real numbers
  - i.e., f is a scalar-valued function of d-vectors
- If $\mathbf{x}$ is a d-vector argument, then $f(\mathbf{x})$ denotes the value of the function f at $\mathbf{x}$
  - i.e., $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_d)$, $\mathbf{x} \in \mathcal{R}^d$, $f(\mathbf{x}) \in \mathcal{R}$

- Example: we can define a function $f: \mathcal{R}^4 \rightarrow \mathcal{R}$ by

$$f(\mathbf{x}) = x_1 + x_2 - x_4^2$$

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", 2018 (Ch 2, p29)

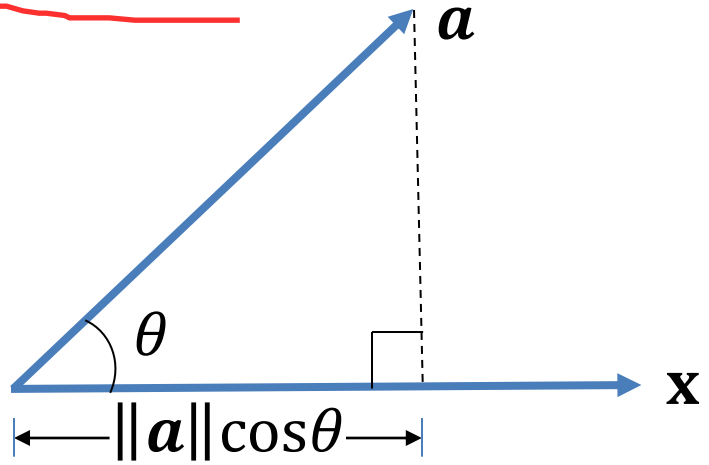
Functions

The inner product function

- Suppose $\boldsymbol{a}$ is a *d*-vector. We can define a scalar valued function f of *d*-vectors, given by

$$f(\mathbf{x}) = \boldsymbol{a}^T\mathbf{x} = a_1x_1 + a_2x_2 + \cdots a_dx_d \qquad (1)$$

  for any *d*-vector $\mathbf{x}$
- The inner product of its *d*-vector argument $\mathbf{x}$ with some (fixed) *d*-vector $\boldsymbol{a}$
- We can also think of f as forming a **weighted sum** of the elements of $\mathbf{x}$; the elements of $\boldsymbol{a}$ give the weights



Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (p30)

Functions

Linear Functions

just mean it can be expressed as a linear combination of vectors

A function $f: \mathcal{R}^d \rightarrow \mathcal{R}$ is **linear** if it satisfies the following two properties:

- **Homogeneity**
  - For any *d*-vector $\mathbf{x}$ and any scalar α, $f(\alpha\mathbf{x}) = \alpha f(\mathbf{x})$
  - Scaling the (vector) argument is the same as scaling the function value

- **Additivity**
  - For any *d*-vectors $\mathbf{x}$ and $\mathbf{y}$, $f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$
  - Adding (vector) arguments is the same as adding the function values

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (p31)

Functions

Linear Functions

Superposition and linearity

- The inner product function $f(\mathbf{x}) = \boldsymbol{a}^T\mathbf{x}$ defined in equation (1) (slide 9) satisfies the property

$$\begin{aligned} f(\alpha\mathbf{x} + \beta\mathbf{y}) &= \boldsymbol{a}^T(\alpha\mathbf{x} + \beta\mathbf{y}) \\ &= \boldsymbol{a}^T(\alpha\mathbf{x}) + \boldsymbol{a}^T(\beta\mathbf{y}) \\ &= \alpha(\boldsymbol{a}^T\mathbf{x}) + \beta(\boldsymbol{a}^T\mathbf{y}) \\ &= \alpha f(\mathbf{x}) + \beta f(\mathbf{y}) \end{aligned}$$

  for all d-vectors $\mathbf{x}$, $\mathbf{y}$, and all scalars α, β.

- This property is called superposition, which consists of homogeneity and additivity
- A function that satisfies the superposition property is called linear

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (p30)

Functions

Linear Functions

- If a function f is linear, superposition extends to linear combinations of any number of vectors:

$$f(\alpha_1 \mathbf{x}_1 + \cdots + \alpha_k \mathbf{x}_k) = \alpha_1 f(\mathbf{x}_1) + \cdots + \alpha_k f(\mathbf{x}_k)$$

for any d vectors $\mathbf{x}_1 + \cdots + \mathbf{x}_k$, and any scalars $\alpha_1 + \cdots + \alpha_k$.

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (p30)

Functions

Linear and Affine Functions

linear combination + a constant (of so called offset/bias)

A linear function plus a constant is called an affine function

A linear function $f: \mathcal{R}^d \rightarrow \mathcal{R}$ is **affine** if and only if it can be expressed as $f(\mathbf{x}) = \boldsymbol{a}^T\mathbf{x} + b$ for some d-vector $\boldsymbol{a}$ and scalar b, which is called the offset (or bias)

$Ax+b$

Example:

$$f(\mathbf{x}) = 2.3 - 2x_1 + 1.3x_2 - x_3$$

This function is affine, with $b = 2.3$, $\boldsymbol{a}^T = [-2,\ 1.3,\ -1]$.

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (p32)

Functions

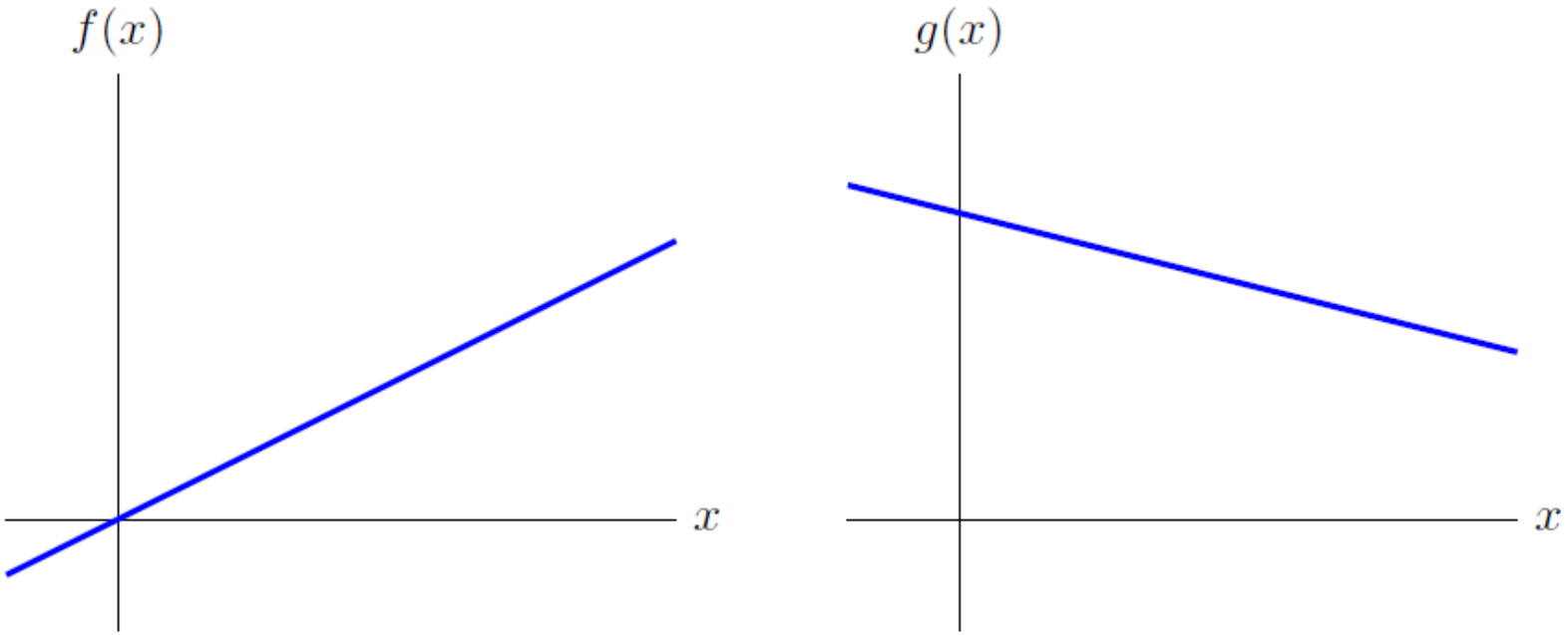


Figure 2.1 *Left.* The function f is linear. *Right.* The function g is affine, but not linear.

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (p33)

Summary

- Operations on Vectors and Matrices
  - Dot-product, matrix inverse
- Systems of Linear Equations $\mathbf{X}\mathbf{w} = \mathbf{y}$
  - Matrix-vector notation, linear dependency, invertible
  - Even-, over-, under-determined linear systems
- Set and Functions

Assignment 1 (week 6)
Tutorial 4

$\mathbf{X}$ is Square	Even-determined	m = d	One unique solution in general	$\hat{\mathbf{w}} = \mathbf{X}^{-1}\mathbf{y}$
$\mathbf{X}$ is Tall	Over-determined	m > d	No exact solution in general; An approximated solution	$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ Left-inverse
$\mathbf{X}$ is Wide	Under-determined	m < d	Infinite number of solutions in general; Unique constrained solution	$\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$ Right-inverse

- Scalar and vector functions
- Inner product function
- Linear and affine functions

python package *numpy*
Inverse: *numpy.linalg.inv(X)*
Transpose: *X.T*

EE2211 Introduction to Machine Learning

Lecture 5

Semester 1
2025/2026

Helen Juan Zhou
helen.zhou@nus.edu.sg

Department of Electrical and Computer Engineering
National University of Singapore

Acknowledgement:
EE2211 development team
(Xinchao, Helen, Thomas, Kar-Ann, Vincent, Chen Khong, Robby, and Haizhou)

Course Contents

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Probability and Statistics
- Fundamental Machine Learning Algorithms I (Helen)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Helen)
  - Over-fitting, bias/variance trade-off
  - Optimization, Gradient descent
  - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
  - Performance Issues
  - K-means Clustering
  - Neural Networks

Midterm L1 to L6
Trial Quiz

Least Squares and Linear Regression

Module II Contents

- Notations, Vectors, Matrices (introduced in L3)
- Operations on Vectors and Matrices
- Systems of Linear Equations
- Set and Functions
- Derivative and Gradient
- Least Squares, Linear Regression
- Linear Regression with Multiple Outputs
- Linear Regression for Classification
- Ridge Regression
- Polynomial Regression

Recap: Linear and Affine Functions

Linear Functions

A function $f: \mathcal{R}^d \rightarrow \mathcal{R}$ is **linear** if it satisfies the following two properties:

- **Homogeneity** $f(\alpha \mathbf{x}) = \alpha f(\mathbf{x})$ Scaling
- **Additivity** $f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$ Adding

Ax

can be expressed as a linear combination

Inner product function

$$f(\mathbf{x}) = \boldsymbol{a}^T \mathbf{x} = a_1 x_1 + a_2 x_2 + \cdots a_d x_d$$

only in this form

linear function vs inner product:inner product returns a scalar from dot product of two vectors.
Linear function is take in vector and can express as Ax, and as a linear combination, not about the output

Affine function

$f(\mathbf{x}) = \boldsymbol{a}^T \mathbf{x} + b$ scalar b is called the offset (or bias)

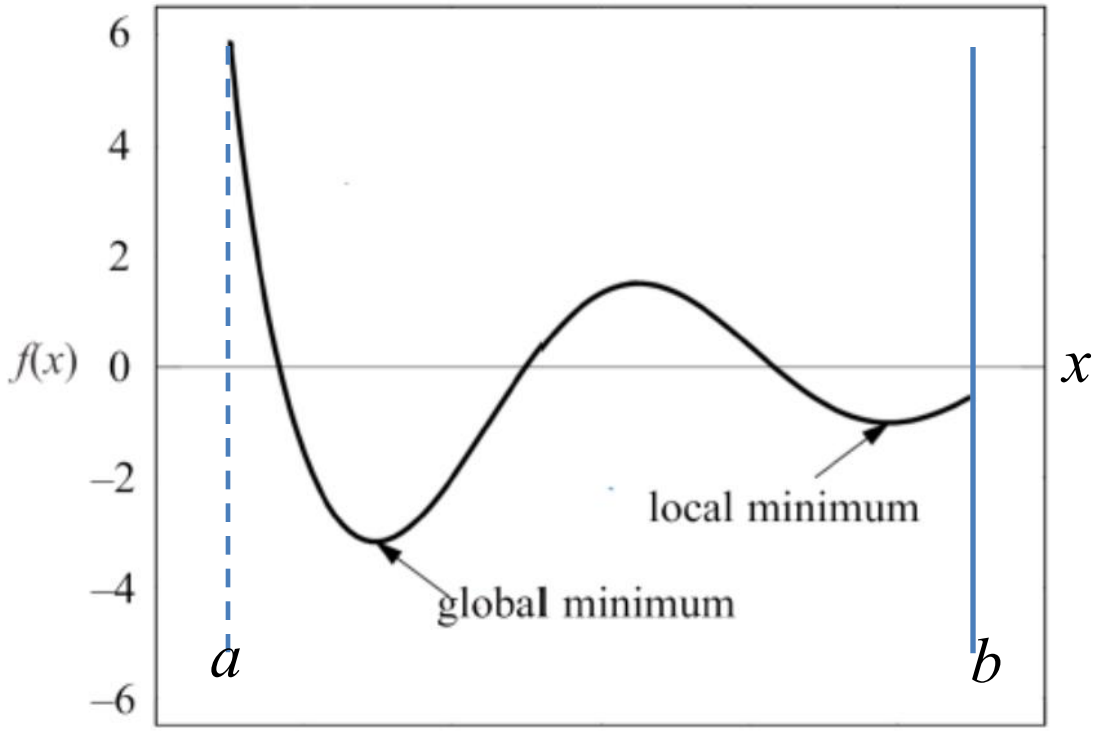
or Ax + b

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (p31)

Functions: Maximum and Minimum

- $f(x)$ has a **local minimum** at $x = c$ if $f(x) \geq f(c)$ for every x in some open interval around $x = c$
- $f(x)$ has a **global minimum** at $x = c$ if $f(x) \geq f(c)$ for all x in the domain of f

A local and a global minima of a function



$$a < x \leq b$$

Note: An **interval** is a set of real numbers with the property that any number that lies between two numbers in the set is also included in the set.
An **open interval** does not include its endpoints and is denoted using parentheses. E.g. (0, 1) means "all numbers greater than 0 and less than 1".

Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (p6-7 of chp2).

Functions: Maximum and Minimum

Max and Arg Max

- Given a set of values $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$,
- The operator $\max_{a \in \mathcal{A}} f(a)$ returns the highest value $f(a)$ for all elements in the set $\mathcal{A}$
- The operator $\arg\max_{a \in \mathcal{A}} f(a)$ returns the element of the set $\mathcal{A}$ that maximizes $f(a)$
- When the set is **implicit** or **infinite**, we can write

$$\max_a f(a) \quad \text{or} \quad \arg\max_a f(a)$$

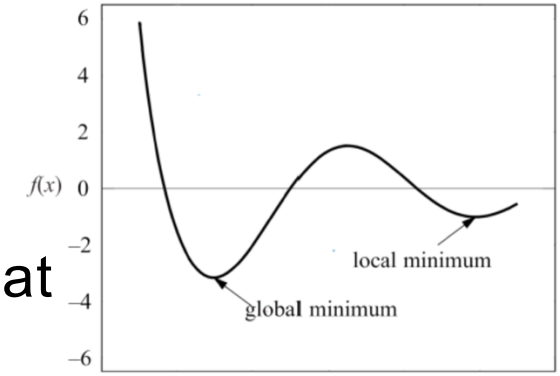
E.g. $f(a) = 3a$, $a \in [0,1]$ → $\max_a f(a) = 3$ and $\arg\max_a f(a) = 1$

Min and **Arg Min** operate in a similar manner

Note: **arg max** returns a value from the **domain** of the function and **max** returns from the **range (codomain)** of the function.

Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (p6-7 of chp2).

Derivative and Gradient



- The **derivative f'** of a function f is a function that describes how fast f grows (or decreases)
  - If the derivative is a constant value, e.g. 5 or −3
    - The function f grows (or decreases) constantly at any point x of its domain
  - When the derivative f' is a function
    - If f' is positive at some x, then the function f grows at this point
    - If f' is negative at some x, then the function f decreases at this point
    - The derivative of zero at x means that the function's slope at x is horizontal (e.g. maximum or minimum points)

- The process of finding a derivative is called **differentiation.**

- **Gradient** is the generalization of derivative for functions that take several inputs (or one input in the form of a vector or some other complex structure).

Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (p8 of chp2).

Derivative and Gradient

The gradient of a function is a vector of **partial derivatives**

Differentiation of a scalar function w.r.t. a vector

If $f(\mathbf{x})$ is a scalar function of d variables, $\mathbf{x}$ is a d x1 vector. Then differentiation of $f(\mathbf{x})$ w.r.t. $\mathbf{x}$ results in a d x1 vector

$$\frac{df(\mathbf{x})}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix}$$

This is referred to as the **gradient** of $f(\mathbf{x})$ and often written as $\nabla_{\mathbf{x}} f$.

E.g. $f(\mathbf{x}) = ax_1 + bx_2 \quad \nabla_{\mathbf{x}} f = \begin{bmatrix} a \\ b \end{bmatrix}$

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Appendix)

Derivative and Gradient

Partial Derivatives

Differentiation of a vector function w.r.t. a vector

If $\mathbf{f}(\mathbf{x})$ is a vector function of *size h* x1 and $\mathbf{x}$ is a *d* x1 vector. Then differentiation of $\mathbf{f}(\mathbf{x})$ results in a *h* x *d* matrix

$$\frac{d\mathbf{f}(\mathbf{x})}{d\mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_h}{\partial x_1} & \cdots & \frac{\partial f_h}{\partial x_d} \end{bmatrix}$$

The matrix is referred to as the **Jacobian** of $\mathbf{f}(\mathbf{x})$

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Appendix)

Derivative and Gradient

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

Some Vector-Matrix Differentiation Formulae

$$\frac{d\mathbf{Ax}}{d\mathbf{x}} = \mathbf{A}$$

$$\frac{d(\boldsymbol{b}^T\mathbf{x})}{d\mathbf{x}} = \boldsymbol{b} \qquad \frac{d(\mathbf{y}^T\mathbf{Ax})}{d\mathbf{x}} = \mathbf{A}^T\mathbf{y}$$

$$\frac{d(\mathbf{x}^T\mathbf{Ax})}{d\mathbf{x}} = (\mathbf{A} + \mathbf{A}^T)\mathbf{x}$$

$$f(\mathbf{x}) = \boldsymbol{a}^T\mathbf{x} = a_1x_1 + a_2x_2 + \cdots a_dx_d$$

Derivations: https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Appendix)

Linear Regression

- **Linear regression** is a popular regression learning algorithm that learns a model which is a linear combination of features of the input example.

$$\mathbf{X}\mathbf{w} = \mathbf{y}, \quad \mathbf{X} \in \mathcal{R}^{m \times d}, \mathbf{w} \in \mathcal{R}^{d \times 1}, \mathbf{y} \in \mathcal{R}^{m \times 1}$$

$$\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,d} \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (p3 of chp3).

Linear Regression

x as an instance is a column vector, number of rows represent the number of features. multiple x will be stacked by transposing them, hence features are now the columns

Problem Statement: To predict the unknown y for a given $\mathbf{x}$ **(testing)**

- We have a collection of labeled examples (**training**) $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$
  - m is the size of the collection
  - $\mathbf{x}_i$ is the d-dimensional feature vector of example $i = 1, \ldots, m$ (input)
  - y_i is a real-valued target (1-D)
  - Note:

linear regression can be used for both problems

    - when y_i is **continuous** valued, it is a **regression problem**
    - when y_i is **discrete** valued, it is a **classification problem**

- We want to build a model $f_{\mathbf{w},b}(\mathbf{x})$ as a linear combination of features of example $\mathbf{x}$: $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b$

  where $\mathbf{w}$ is a d-dimensional vector of parameters and b is a real number.
- The notation $f_{\mathbf{w},b}$ means that the model f is parametrized by two values: $\mathbf{w}$ and b

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (chp.14)

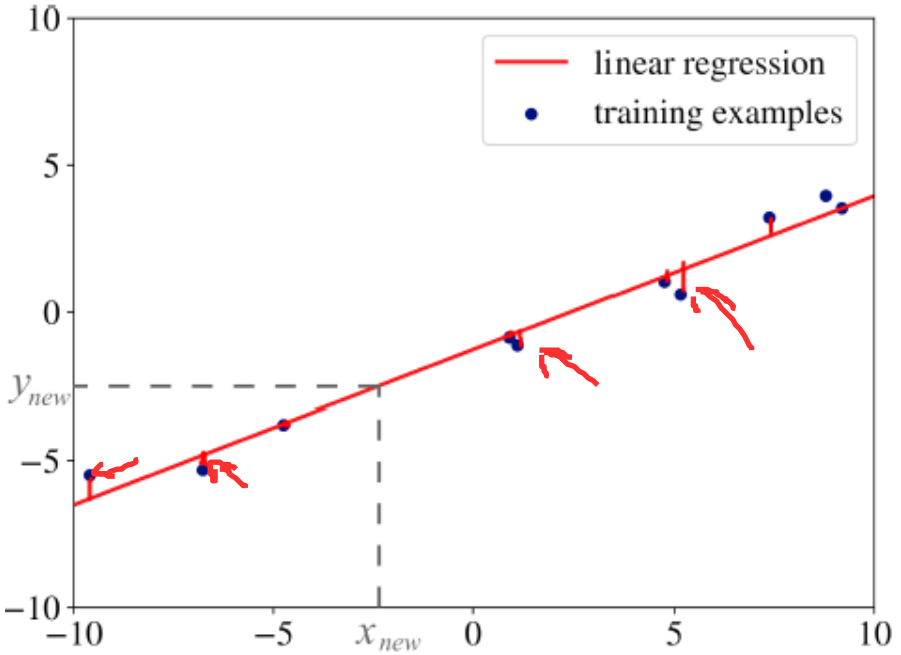
Linear Regression

Learning objective function

- To find the optimal values for $\mathbf{w}^*$ and b^* which **minimizes** the following expression:

$$\frac{1}{m}\sum_{i=1}^{m}(f_{\mathbf{w},b}(\mathbf{x}_i) - \mathrm{y}_i)^2$$

- In mathematics, the expression we minimize or maximize is called an **objective function**, or, simply, an **objective**



This is a minimisation problem, hence cost function is the objective function here, which is the sum of the loss function. Loss function is the error output by model compared to actual y. And we want this to be minimum for best model

$(f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2$ is called the **loss function**: a measure of the difference between $f_{\mathbf{w}}(\mathbf{x}_i)$ and y_i or a penalty for misclassification of example *i*.

Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (chp3.1.2)

Linear Regression

Learning objective function (using simplified notation hereon)

- To find the optimal values for $\mathbf{w}^*$ which **minimizes** the following expression:

this is the cost function

$$\sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2$$

we are minimising ALL (sum) the errors between the output of predicted model and the actual training data output values

with $f_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{x}^T\mathbf{w}$;

where we define $\mathbf{w} = [b, w_1, \ldots w_d]^T = [w_0, w_1, \ldots w_d]^T$,

and $\mathbf{x}_i = [1, x_{i,1}, \ldots x_{i,d}]^T = [x_{i,0}, x_{i,1}, \ldots x_{i,d}]^T$, $i = 1, \ldots, m$

- This particular choice of the loss function is called **squared error loss**

Note: The normalization factor $\frac{1}{m}$ can be omitted as it does not affect the optimization.

Linear Regression

$$\sum_{i=1}^{m} (f_{\mathbf{w},b}(\mathbf{x}_i) - \mathrm{y}_i)^2$$

- All model-based learning algorithms have a **loss function**
- What we do to find the best model is **to minimize the objective** known as the **cost function**
- **Cost function** is a sum of **loss functions** over training set plus possibly some model complexity penalty (regularization)

- In linear regression, the cost function is given by the *average loss*, also called the **empirical risk** because we do not have all the data (e.g. testing data)
  - The average of all penalties is obtained by applying the model to the training data

Ref: [Book1] Andriy Burkov, "The Hundred-Page Machine Learning Book", 2019 (chp3.1.2)

Linear Regression

Learning (Training)

- Consider the set of feature vector $\mathbf{x}_i$ and target output y_i indexed by $i = 1, \ldots, m$, a linear model $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ can be stacked as

$$f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w} \quad \Longleftrightarrow \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Learning Model

Learning target vector

$$= \begin{bmatrix} \mathbf{x}_1^T\mathbf{w} \\ \vdots \\ \mathbf{x}_m^T\mathbf{w} \end{bmatrix}$$

where $\mathbf{x}_i^T\mathbf{w} = [1, x_{i,1}, \ldots, x_{i,d}] \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$

Note: The **bias/offset term** is responsible for **translating** the line/plane/hyperplane away from the origin.

Linear Regression

Least Squares Regression

In vector-matrix notation, the minimization of the objective function can be written compactly using $\mathbf{e} = \mathbf{X}\mathbf{w} - \mathbf{y}$:

$$\begin{aligned} \mathrm{J}(\mathbf{w}) &= \mathbf{e}^T\mathbf{e} \\ &= (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= (\mathbf{w}^T\mathbf{X}^T - \mathbf{y}^T)(\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{w}^T\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{y}^T\mathbf{y} \\ &= \mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{y}^T\mathbf{X}\mathbf{w} + \mathbf{y}^T\mathbf{y}. \end{aligned}$$

Note: when $\boldsymbol{f}_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w}$, then

$$\sum_{i=1}^{m}(f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y}).$$

a new and compact way to write the cost function

Linear Regression

Differentiating $\mathrm{J}(\mathbf{w})$ with respect to $\mathbf{w}$ and setting the result to $\mathbf{0}$:

$$\frac{\partial}{\partial \mathbf{w}} \mathrm{J}(\mathbf{w}) = \mathbf{0}$$

$$\frac{\partial}{\partial \mathbf{w}} (\mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{y}^T \mathbf{X}\mathbf{w} + \mathbf{y}^T \mathbf{y}) = \mathbf{0}$$

$$\Rightarrow 2\mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{X}^T \mathbf{y} = \mathbf{0}$$

$$\Rightarrow \mathbf{X}^T \mathbf{X}\mathbf{w} = \mathbf{X}^T \mathbf{y}$$

$\Rightarrow$ Any minimizer $\hat{\mathbf{w}}$ of $\mathrm{J}(\mathbf{w})$ must satisfy $\mathbf{X}^T(\mathbf{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}$.

If $\mathbf{X}^T \mathbf{X}$ is invertible, then

Learning/training: $\hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$

Prediction/testing: $\hat{f}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new} \hat{\mathbf{w}}$

Linear Regression

Example 1 **Training set** $\{(x_i, y_i)\}_{i=1}^{m}$

$$\{x = -9\} \rightarrow \{y = -6\}$$
$$\{x = -7\} \rightarrow \{y = -6\}$$
$$\{x = -5\} \rightarrow \{y = -4\}$$
$$\{x = 1\} \rightarrow \{y = -1\}$$
$$\{x = 5\} \rightarrow \{y = 1\}$$
$$\{x = 9\} \rightarrow \{y = 4\}$$

$$\mathbf{X} \quad \mathbf{w} \quad \mathbf{y}$$

$$\begin{bmatrix} 1 & -9 \\ 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \\ 1 & 9 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -6 \\ -6 \\ -4 \\ -1 \\ 1 \\ 4 \end{bmatrix}$$

read the output W:
- first row is always for the bias term
- if multiple column, they correspond to the multiple column output respectively

This set of linear equations has no exact solution

However, $\mathbf{X}^T\mathbf{X}$ is invertible

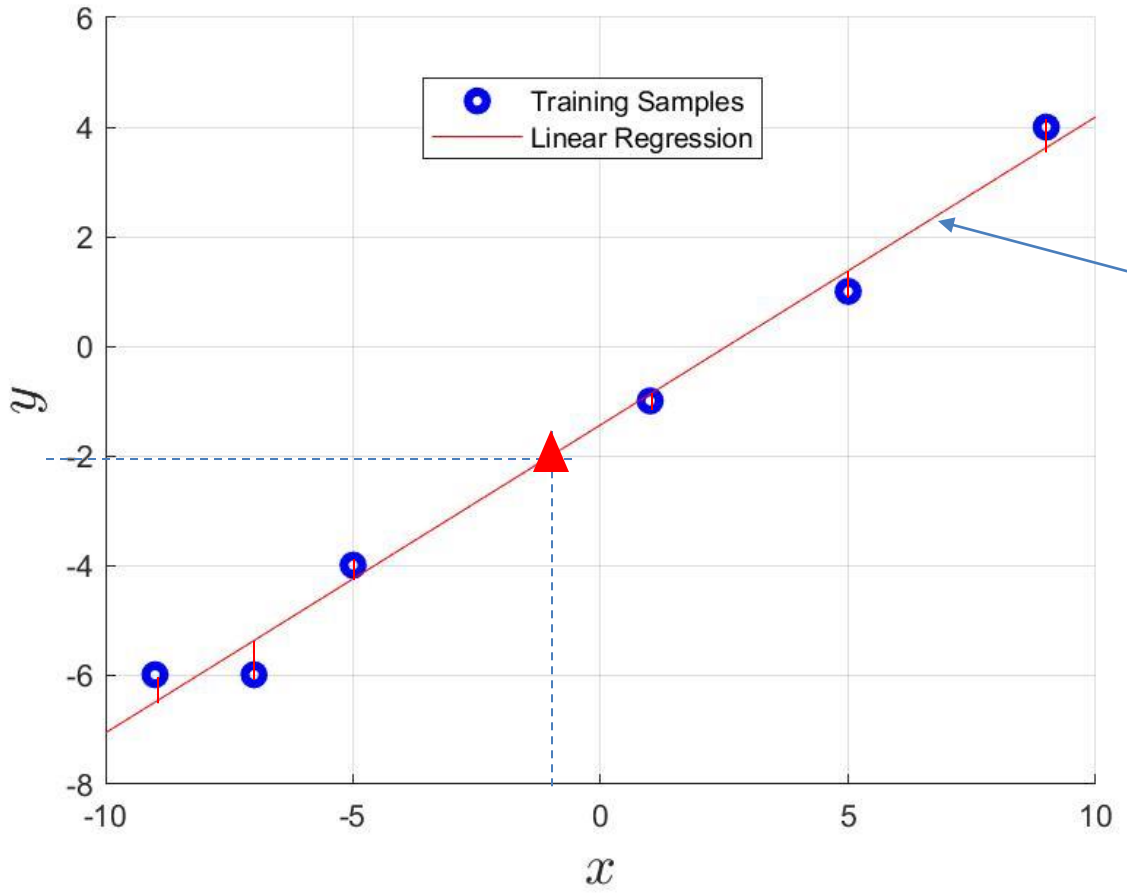
Least square approximation

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$= \begin{bmatrix} 6 & -6 \\ -6 & 262 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -9 & -7 & -5 & 1 & 5 & 9 \end{bmatrix} \begin{bmatrix} -6 \\ -6 \\ -4 \\ -1 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} -1.4375 \\ 0.5625 \end{bmatrix}$$

bias

Linear Regression



Linear Regression on one-dimensional samples

$$\hat{y} = \mathbf{X}\hat{\mathbf{w}}$$
$$= \mathbf{X}\begin{bmatrix} -1.4375 \\ 0.5625 \end{bmatrix}$$
$$y = -1.4375 + 0.5625x$$

Prediction:
Test set

$$\{x = -1\} \rightarrow \{y = ?\}$$

$$\hat{y} = [1 \quad -1]\begin{bmatrix} -1.4375 \\ 0.5625 \end{bmatrix}$$
$$= -2$$

Python demo 1

Linear Regression

bias, but my code dont need

Example 2

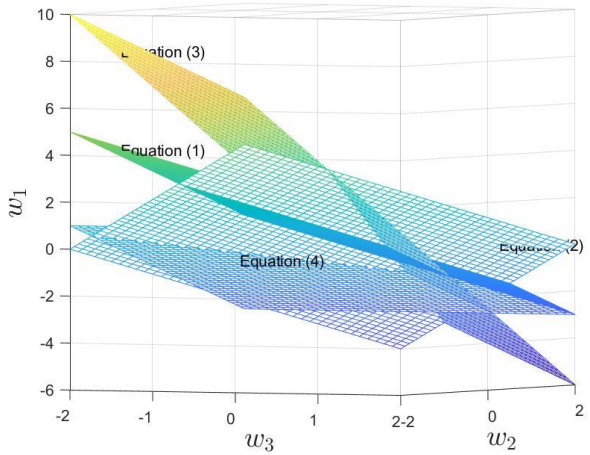
$\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$

$\{x_1 = 1, x_2 = 1, \quad x_3 = 1\} \rightarrow \{y = 1\}$
$\{x_1 = 1, x_2 = -1, x_3 = 1\} \rightarrow \{y = 0\}$
$\{x_1 = 1, x_2 = 1, \quad x_3 = 3\} \rightarrow \{y = 2\}$
$\{x_1 = 1, x_2 = 1, \quad x_3 = 0\} \rightarrow \{y = -1\}$

Training set

$$\begin{matrix} \mathbf{X} & & & \mathbf{w} & \mathbf{y} \end{matrix}$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix}$$



This set of linear equations has no exact solution

However, $\mathbf{X}^T\mathbf{X}$ is invertible

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

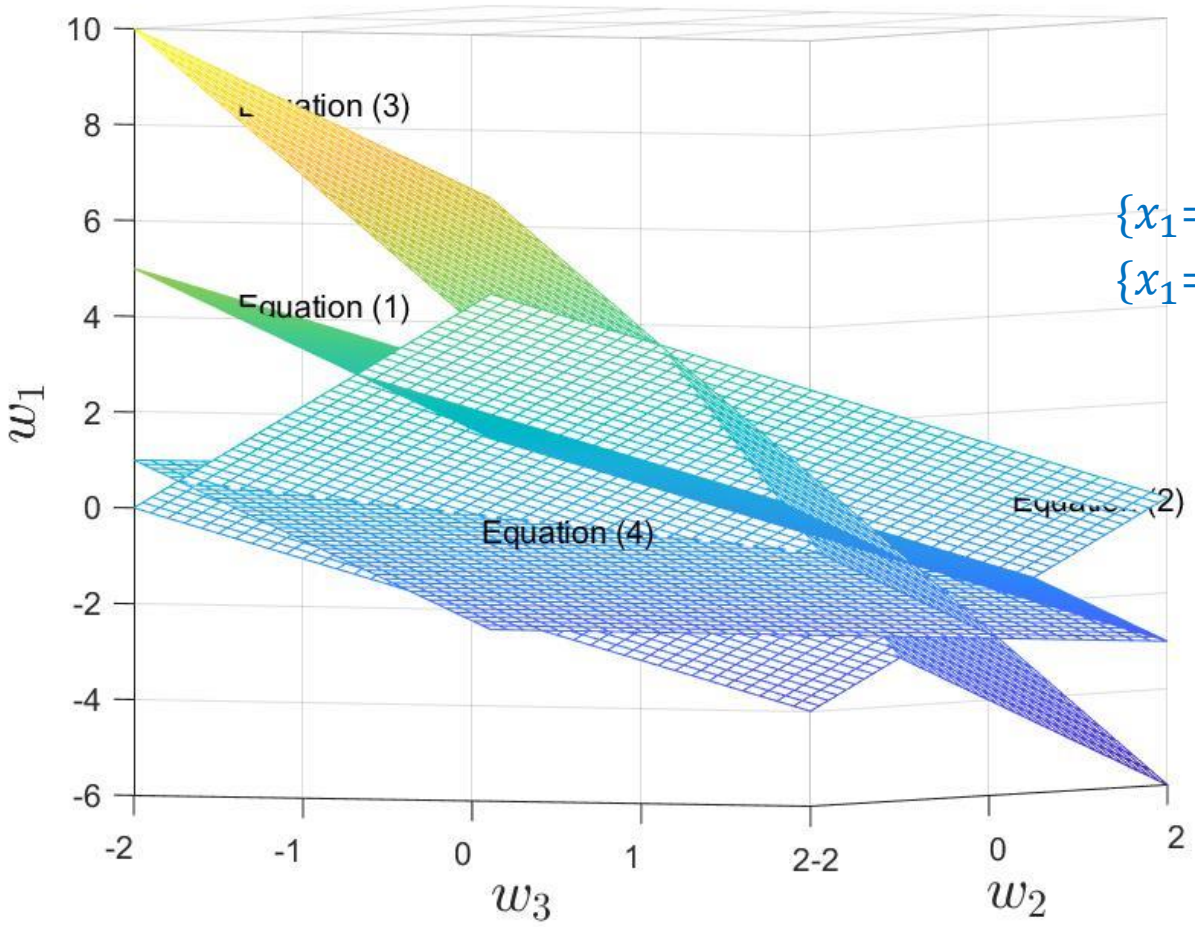
Least square approximation

$$= \begin{bmatrix} 4 & 2 & 5 \\ 2 & 4 & 3 \\ 5 & 3 & 11 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix} = \begin{bmatrix} -0.7500 \\ 0.1786 \\ 0.9286 \end{bmatrix}$$

bias

Linear Regression

The four linear equations



Prediction:

Test set

$\{x_1 = 1,\ x_2 = 6,\quad x_3 = 8\} \rightarrow \{y = ?\}$

$\{x_1 = 1,\ x_2 = 0,\ x_3 = -1\} \rightarrow \{y = ?\}$

$$\hat{\boldsymbol{y}} = \hat{\boldsymbol{f}}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\,\hat{\mathbf{w}}$$

$$\hat{\boldsymbol{y}} = \begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -0.7500 \\ 0.1786 \\ 0.9286 \end{bmatrix}$$

$$= \begin{bmatrix} 7.7500 \\ -1.6786 \end{bmatrix}$$

Linear Regression

Learning of Vectored Function (Multiple Outputs)

For one sample: a linear model $\mathbf{f_w}(\mathbf{x}) = \mathbf{x}^T\mathbf{W}$ Vector function

For m samples: $\mathbf{F_w}(\mathbf{X}) = \mathbf{XW} = \mathbf{Y}$

Sample 1 ⋮ Sample m

$$\begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix} \mathbf{W} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \cdots & x_{m,d} \end{bmatrix} \begin{bmatrix} w_{0,1} & \cdots & w_{0,h} \\ w_{1,1} & \cdots & w_{1,h} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,h} \end{bmatrix}$$

h

Sample 1's output ⋮ Sample m's output

$$\begin{bmatrix} y_{1,1} & \cdots & y_{1,h} \\ & \vdots & \\ y_{m,1} & \cdots & y_{m,h} \end{bmatrix}$$

m, h

$$\mathbf{X} \in \mathcal{R}^{m\times(d+1)},\ \mathbf{W} \in \mathcal{R}^{(d+1)\times h},\ \mathbf{Y} \in \mathcal{R}^{m\times h}$$

Linear Regression

Objective: $\sum_{i=1}^{m}(\mathbf{f_w}(\mathbf{x}_i) - \mathbf{y}_i)^2 = \boldsymbol{E}^T\boldsymbol{E}$

Least Squares Regression of Multiple Outputs

In matrix notation, the sum of squared errors cost function can be written compactly using $\mathbf{E} = \mathbf{XW} - \mathbf{Y}$:

$$\mathrm{J}(\mathbf{W}) = \mathrm{trace}(\mathbf{E}^T\mathbf{E})$$
$$= \mathrm{trace}[(\mathbf{XW} - \mathbf{Y})^T(\mathbf{XW} - \mathbf{Y})]$$

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning/training: $\widehat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$

Prediction/testing: $\hat{\mathbf{F}}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\widehat{\mathbf{W}}$

Ref: Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning", (2nd ed., 12th printing) 2017 (chp.3.2.4)

Linear Regression

Least Squares Regression of Multiple Outputs

$$J(\mathbf{W}) = \text{trace}(\mathbf{E}^T\mathbf{E})$$

$$= \text{trace}\left(\begin{bmatrix}\mathbf{e}_1^T \\ \vdots \\ \mathbf{e}_h^T\end{bmatrix}[\mathbf{e}_1 \quad \mathbf{e}_2 \quad \dots \quad \mathbf{e}_h]\right)$$

$$= \text{trace}\left(\begin{bmatrix}\mathbf{e}_1^T\mathbf{e}_1 & \mathbf{e}_1^T\mathbf{e}_2 & \dots & \mathbf{e}_1^T\mathbf{e}_h \\ \mathbf{e}_2^T\mathbf{e}_1 & \mathbf{e}_2^T\mathbf{e}_2 & \dots & \mathbf{e}_2^T\mathbf{e}_h \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{e}_h^T\mathbf{e}_1 & \mathbf{e}_h^T\mathbf{e}_2 & \dots & \mathbf{e}_h^T\mathbf{e}_h\end{bmatrix}\right) = \sum_{k=1}^{h}\mathbf{e}_k^T\mathbf{e}_k$$

Linear Regression of multiple outputs

Example 3

Training set $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$

$$\{x_1=1, x_2=1, \quad x_3=1\} \rightarrow \{y_1=1, \quad y_2=0\}$$
$$\{x_1=1, x_2=-1, x_3=1\} \rightarrow \{y_1=0, \quad y_2=1\}$$
$$\{x_1=1, x_2=1, \quad x_3=3\} \rightarrow \{y_1=2, y_2=-1\}$$
$$\{x_1=1, x_2=1, \quad x_3=0\} \rightarrow \{y_1=-1, y_2=3\}$$

$\mathbf{X}$ $\mathbf{W}$ $\mathbf{Y}$

Bias →
$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & -1 \\ -1 & 3 \end{bmatrix}$$

read the output W:
- first row is always for the bias term
- if multiple column, they correspond to the multiple column output respectively

This set of linear equations has NO exact solution

Least square approximation

$$\hat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

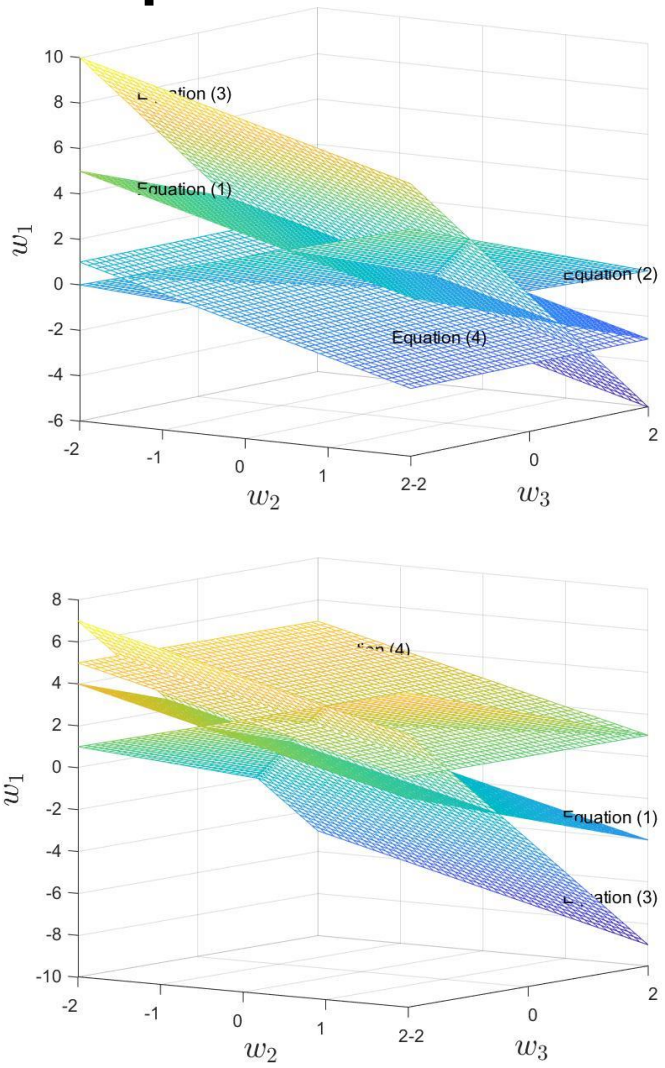
$\mathbf{X}^T\mathbf{X}$ is invertible

$$= \begin{bmatrix} 4 & 2 & 5 \\ 2 & 4 & 3 \\ 5 & 3 & 11 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & -1 \\ -1 & 3 \end{bmatrix} = \begin{bmatrix} -0.75 & 2.25 \\ 0.1786 & 0.0357 \\ 0.9286 & -1.2143 \end{bmatrix}$$

bias

Linear Regression of multiple outputs

Example 3



Prediction:

Test set: two new samples

$\{x_1 = 1,\ x_2 = 6,\quad x_3 = 8\} \rightarrow \{y_1 = ?,\ y_2 = ?\}$

$\{x_1 = 1,\ x_2 = 0,\ x_3 = -1\} \rightarrow \{y_1 = ?,\ y_2 = ?\}$

$$\hat{\mathbf{Y}} = \mathbf{X}_{new}\,\hat{\mathbf{W}}$$

Bias →

$$= \begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -0.75 & 2.25 \\ 0.1786 & 0.0357 \\ 0.9286 & -1.2143 \end{bmatrix}$$

$$= \begin{bmatrix} 7.75 & -7.25 \\ -1.6786 & 3.4643 \end{bmatrix}$$

Python demo 2

Linear Regression of multiple outputs

Example 4

The values of feature x and their corresponding values of multiple outputs target **y** are shown in the table below.

Based on the least square regression, what are the values of **w**?
Based on the current mapping, when x = 2, what is the value of **y**?

x	[3]	[4]	[10]	[6]	[7]
y	[0, 5]	[1.5, 4]	[-3, 8]	[-4, 10]	[1, 6]

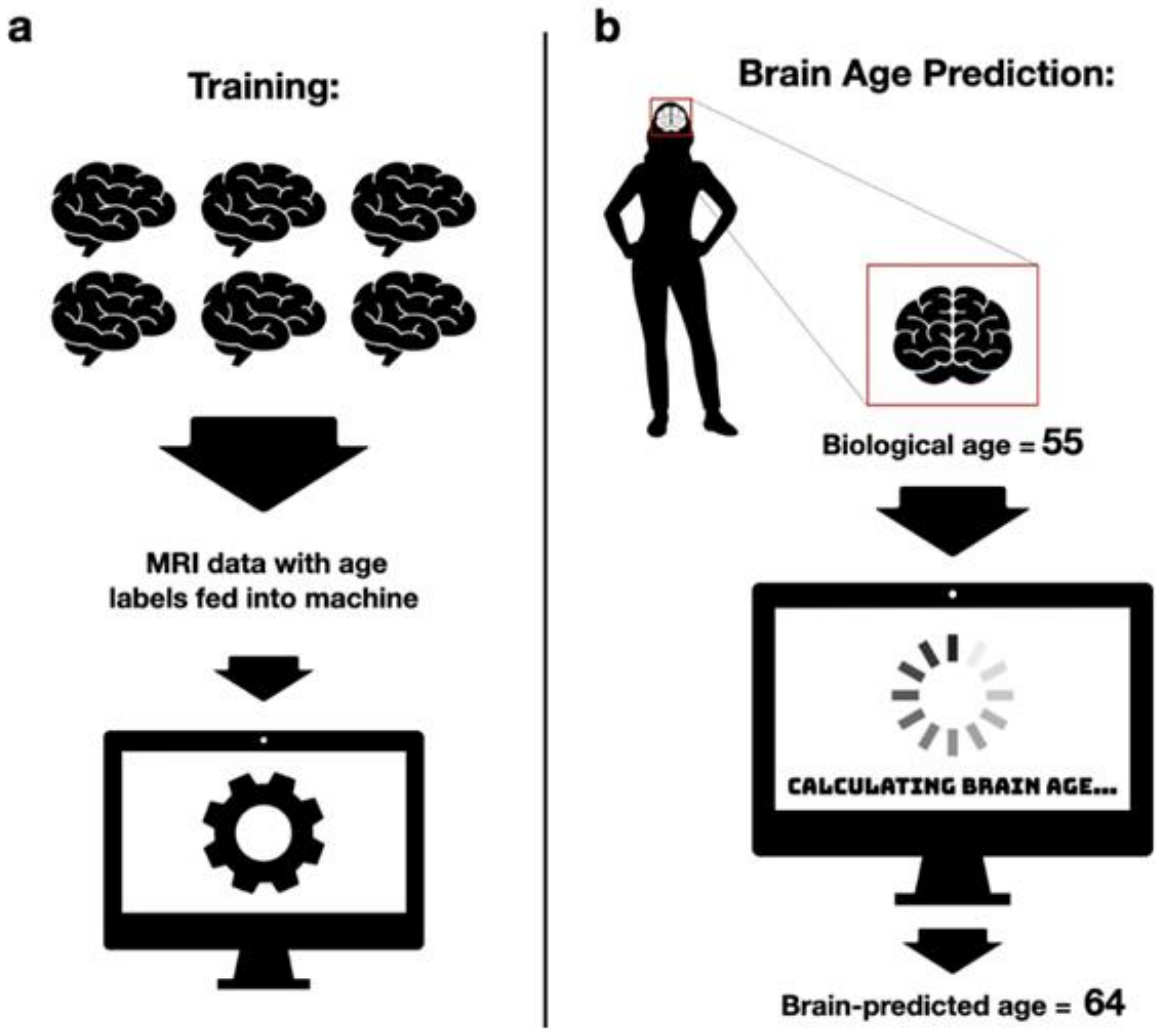
$$\widehat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \begin{bmatrix} 1.9 & 3.6 \\ -0.4667 & 0.5 \end{bmatrix}$$

Python demo 3

$$\widehat{\mathbf{Y}_{new}} = \mathbf{X}_{new}\,\widehat{\mathbf{W}} = [1 \quad 2]\,\widehat{\mathbf{W}} = [\,0.9667 \quad 4.6]$$

Prediction

Example: Brain Age Prediction



- Input:
  - Brain measures
- Output:
  - Age

https://www.med.uio.no/norment/english/research/this-months-researcher/dani-beck-english.html

Summary

- Notations, Vectors, Matrices
- Operations on Vectors and Matrices
  - Dot-product, matrix inverse
- Systems of Linear Equations $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w} = \mathbf{y}$
  - Matrix-vector notation, linear dependency, invertible
  - Even-, over-, under-determined linear systems
- Functions, Derivative and Gradient
  - Inner product, linear/affine functions
  - Maximum and minimum, partial derivatives, gradient
- Least Squares, Linear Regression
  - Objective function, loss function
  - Least square solution, training/learning and testing/prediction
  - Linear regression with multiple outputs

Midterm (L1 to L6)

Learning/training $\hat{\mathbf{w}} = (\mathbf{X}_{train}^T \mathbf{X}_{train})^{-1} \mathbf{X}_{train}^T \mathbf{y}_{train}$

Prediction/testing $\mathbf{y}_{test} = \mathbf{X}_{test}\, \hat{\mathbf{w}}$

- Classification
- Ridge Regression
- Polynomial Regression

Python packages: numpy, pandas, matplotlib.pyplot, numpy.linalg, and sklearn.metrics (for mean_squared_error), numpy.linalg.pinv

EE2211 Introduction to Machine Learning

Lecture 6

Juan Helen Zhou
helen.zhou@nus.edu.sg

Electrical and Computer Engineering Department
National University of Singapore

Acknowledgement:
EE2211 development team
(Xinchao, Helen, Thomas, Kar-Ann, Chen Khong, Robby and Haizhou

Ridge Regression & Polynomial Regression

Midterm – Oct 11th (L1 to L6, 1-2pm)
Assignment 1 & 2

Lecture 7 rescheduled to Oct. 1st (Wed), 2-4pm

Module II Contents

- Notations, Vectors, Matrices
- Operations on Vectors and Matrices
- Systems of Linear Equations
- Functions, Derivative and Gradient
- Least Squares, Linear Regression
- Linear Regression with Multiple Outputs
- Linear Regression for Classification
- Ridge Regression
- Polynomial Regression

Review: Linear Regression

Learning of Scalar Function (Single Output)

For one sample: a linear model $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ scalar function

For *m* samples: $\boldsymbol{f}_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w} = \mathbf{y}$

$$\mathbf{y} = \begin{bmatrix} \mathbf{x}_1^T\mathbf{w} \\ \vdots \\ \mathbf{x}_m^T\mathbf{w} \end{bmatrix} \quad \text{where} \quad \mathbf{x}_i^T = [1, x_{i,1}, \ldots, x_{i,d}]$$

$$\mathbf{X} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \cdots & x_{m,d} \end{bmatrix} \qquad \mathbf{w} = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Objective: $\sum_{i=1}^{m}(f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2 = \mathbf{e}^T\mathbf{e} = (\mathbf{X}\mathbf{w} - \mathbf{y})^T(\mathbf{X}\mathbf{w} - \mathbf{y})$

Learning/training when $\mathbf{X}^T\mathbf{X}$ is invertible

Least square solution: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

Prediction/testing: $\boldsymbol{y}_{new} = \hat{\boldsymbol{f}}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$

Review: Linear Regression

Learning of Vectored Function (Multiple Outputs)

$$\mathbf{F_w}(\mathbf{X}) = \mathbf{XW} = \mathbf{Y}$$

Sample 1 ------->
$$\begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix} \mathbf{W} = \begin{bmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \cdots & x_{m,d} \end{bmatrix} \begin{bmatrix} w_{0,1} & \cdots & w_{0,h} \\ w_{1,1} & \cdots & w_{1,h} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \cdots & w_{d,h} \end{bmatrix}$$
Sample m ------->

Sample 1's output ->
$$\begin{bmatrix} y_{1,1} & \cdots & y_{1,h} \\ \vdots & & \\ y_{m,1} & \cdots & y_{m,h} \end{bmatrix}$$
Sample m's output ->

Least Squares Regression

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning/training: $\hat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$

Prediction/testing: $\hat{\mathbf{F}}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{W}}$

$\mathbf{X} \in \mathcal{R}^{m\times(d+1)}$, $\mathbf{W} \in \mathcal{R}^{(d+1)\times h}$, $\mathbf{Y} \in \mathcal{R}^{m\times h}$

Linear Regression (for classification)

Linear Methods for Classification

assign one class to positive 1 and the other to negative 1

- We have a collection of labeled examples
  - m is the size of the collection
  - $\mathbf{x}_i$ is the d-dimensional feature vector of example $i = 1, \ldots, m$
  - y_i is discrete target label (e.g., $y_i \in \{-1, +1\}$ or $\{0, 1\}$ for binary classification problems)
  - Note:
    - when y_i is **continuous** valued → a **regression problem**
    - when y_i is **discrete** valued →a **classification problem**
- Linear model: $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b$ or in compact form $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ (having the offset term absorbed into the inner product)

Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (chp.14)

Linear Regression (for classification)

Linear Methods for Classification

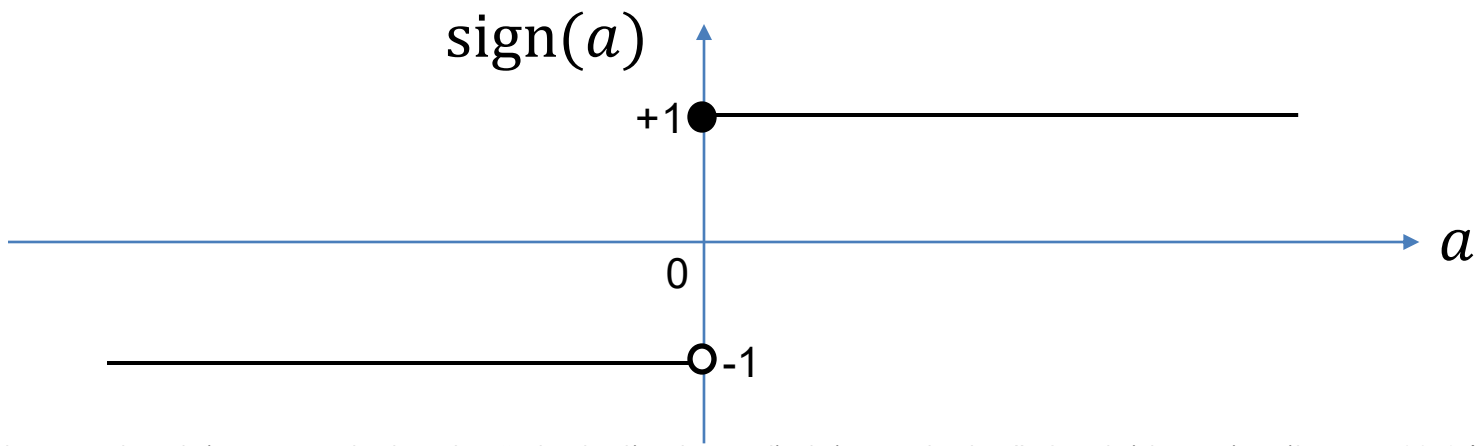
Binary Classification:

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$, $y_i \in \{-1, +1\}, i = 1, \ldots, m$

Prediction: $\hat{f}^c_{\mathbf{w}}(\mathbf{x}_{new}) = \text{sign}(\mathbf{x}^T_{new}\hat{\mathbf{w}})$ for each row $\mathbf{x}^T_{new}$ of $\mathbf{X}_{new}$

$\mathbf{sign}(a) = +1$ for $a \geq 0$ and -1 for $a < 0$



Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (chp.14)

Linear Regression (for classification)

Example 1 **Training set** $\{x_i, y_i\}_{i=1}^{m}$

$$\{x = -9\} \rightarrow \{y = -1\}$$
$$\{x = -7\} \rightarrow \{y = -1\}$$
$$\{x = -5\} \rightarrow \{y = -1\}$$
$$\{x = 1\} \rightarrow \{y = +1\}$$
$$\{x = 5\} \rightarrow \{y = +1\}$$
$$\{x = 9\} \rightarrow \{y = +1\}$$

$$\mathbf{X} \quad \mathbf{w} \quad \mathbf{y}$$

Bias $\rightarrow$
$$\begin{bmatrix} 1 & -9 \\ 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \\ 1 & 9 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

This set of linear equations has NO exact solution

$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

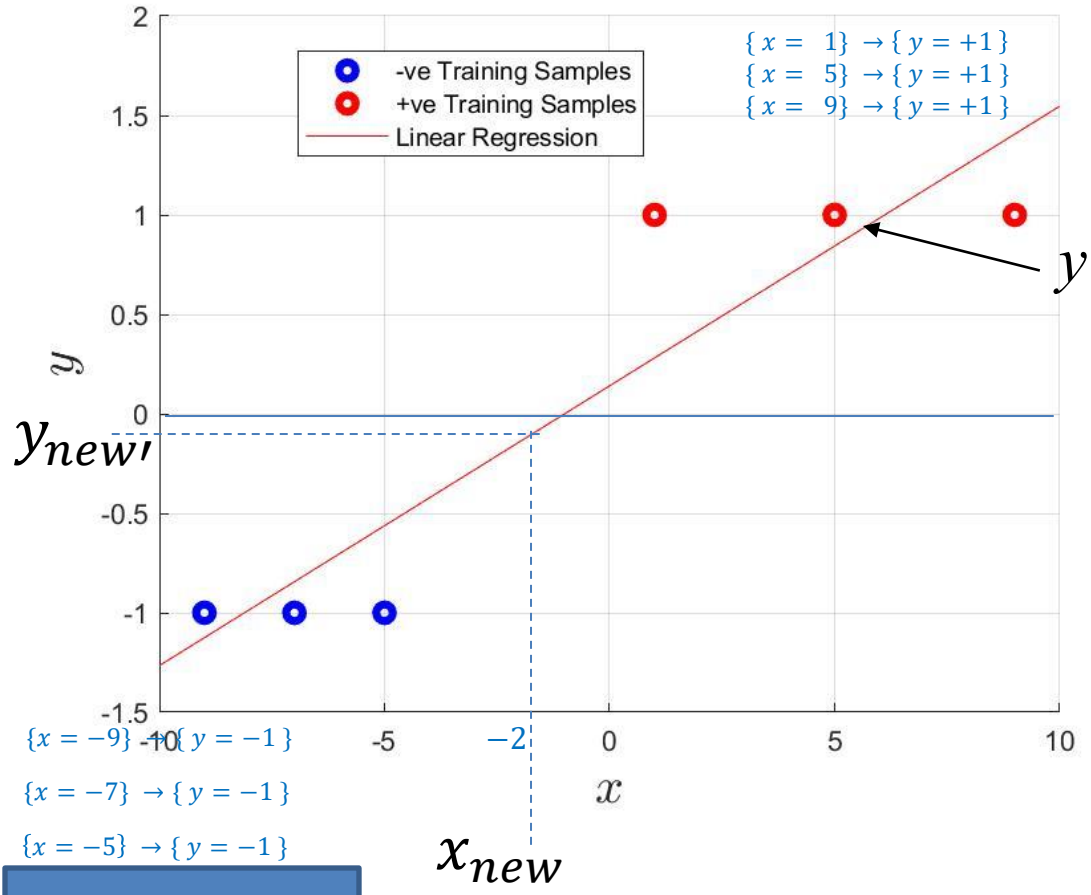
$\mathbf{X}^T\mathbf{X}$ is invertible

$$= \begin{bmatrix} 6 & -6 \\ -6 & 262 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -9 & -7 & -5 & 1 & 5 & 9 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1406 \\ 0.1406 \end{bmatrix}$$

Least square approximation

Linear Regression (for classification)

Example 1



$$\hat{y} = \text{sign}(\mathbf{X}\hat{\mathbf{w}}) = \text{sign}(\mathbf{X}\begin{bmatrix}0.1406\\0.1406\end{bmatrix})$$

$$y' = \mathbf{X}\hat{\mathbf{w}} = 0.1406 + 0.1406x$$

Prediction:

Test set $\{x = -2\} \rightarrow \{y = ?\}$

$$y_{new} = \hat{f}^{c}_{\mathbf{w}}(\mathbf{x}_{new}) = \text{sign}(\mathbf{x}_{new}\hat{\mathbf{w}})$$

$$= \text{sign}(\begin{bmatrix}\overset{\text{Bias}}{\downarrow}\\1 & -2\end{bmatrix}\begin{bmatrix}0.1406\\0.1406\end{bmatrix})$$

$$= \text{sign}(-0.1406) = -1$$

Python demo 1

Linear Regression for one-dimensional classification

Linear Regression (for classification)

Linear Methods for Classification

Multi-Category Classification:

If $\mathbf{X}^T\mathbf{X}$ is invertible, then

Learning: $\widehat{\mathbf{W}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}, \quad \mathbf{Y} \in \mathbf{R}^{m\times C}$

Prediction: $\hat{f}^C_{\mathbf{W}}(\mathbf{x}_{new}) = \arg\max_{k=1,\dots,C}\left(\mathbf{x}^T_{new}\widehat{\mathbf{W}}(:,k)\right)$ for each $\mathbf{x}^T_{new}$ of $\mathbf{X}_{new}$

$\hat{y}_{new}$

Each row (of $i = 1, \dots, m$) in $\mathbf{Y}$ has an **one-hot** encoding/assignment:

e.g., target for class-1 is labelled as $\mathbf{y}_i^T = [1, 0, 0, \dots, 0]$ for the ith sample,
target for class-2 is labelled as $\mathbf{y}_j^T = [0, 1, 0, \dots, 0]$ for the jth sample,
target for class-C is labelled as $\mathbf{y}_m^T = \underbrace{[0, 0, \dots, 0, 1]}_{C}$ for the mth sample.

Ref: Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning", (2nd ed., 12th printing) 2017 (chp.4)

Linear Regression (for classification)

Example 2 Three class classification

Training set

$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$

$\{x_1 = 1, \quad x_2 = 1\} \rightarrow \{y_1 = 1, y_2 = 0, y_3 = 0\}$ Class 1

$\{x_1 = -1, \quad x_2 = 1\} \rightarrow \{y_1 = 0, y_2 = 1, y_3 = 0\}$ Class 2

$\{x_1 = 1, \quad x_2 = 3\} \rightarrow \{y_1 = 1, y_2 = 0, y_3 = 0\}$ Class 1

$\{x_1 = 1, \quad x_2 = 0\} \rightarrow \{y_1 = 0, y_2 = 0, y_3 = 1\}$ Class 3

$$\underset{\text{Bias}\rightarrow}{\mathbf{X}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix} \mathbf{W} \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix} = \mathbf{Y} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This set of linear equations has NO exact solution.

Least square approximation

$$\hat{\mathbf{W}} = \mathbf{X}^{\dagger}\mathbf{Y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$\mathbf{X}^T\mathbf{X}$ is invertible

$$= \begin{bmatrix} 4 & 2 & 5 \\ 2 & 4 & 3 \\ 5 & 3 & 11 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 3 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.2857 & -0.5 & 0.2143 \\ 0.2857 & 0 & -0.2857 \end{bmatrix}$$

Linear Regression (for classification)

Example 2 Prediction

Test set $\mathbf{X}_{new}$ $\{x_1 = 6,\ x_2 = 8\} \rightarrow \{class\ 1, 2, or\ 3?\}$

$\{x_1 = 0,\ x_2 = -1\} \rightarrow \{class\ 1, 2, or\ 3?\}$

$$\hat{\mathbf{Y}} = \mathbf{X}_{new}\,\widehat{\mathbf{W}} = \begin{bmatrix} 1 & 6 & 8 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 & 0.5 & 0.5 \\ 0.2857 & -0.5 & 0.2143 \\ 0.2857 & 0 & -0.2857 \end{bmatrix}$$

Category prediction:

$$\hat{f}^{c}_{\mathbf{w}}(\mathbf{X}_{new}) = \arg\max_{k=1,\ldots,c}(\hat{\mathbf{Y}}(:,k))$$

$$= \arg\max_{k=1,\ldots,c}\left(\begin{bmatrix} 4 & -2.50 & -0.50 \\ -0.2587 & 0.50 & 0.7857 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{array}{l} \rightarrow \text{Class 1} \\ \rightarrow \text{Class 3} \end{array}$$

For each row of **Y**, the **column position** of the **largest** number (across all columns for that row) determines the **class label**.

E.g. in the first row, the maximum number is 4 which is in column 1. Therefore, the resulting predicted class is 1.

Python demo 2

Ridge Regression

w (weights) tells how much each feature influences the prediction, without penalty like in least squares, the goal is to only pick weights that minimise the squared error on training data, but if there are many features or features are highly correlated, the model will pick very large values of w to fit the training data perfectly.

Recall Linear regression

Objective: $\widehat{\mathbf{w}} = \text{argmin} \sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y})$

The learning computation: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

We cannot guarantee that the matrix $\mathbf{X}^T\mathbf{X}$ is invertible

Ridge regression: shrinks the regression coefficients *w* by imposing a penalty on their size

to prevent overfitting and handle issues like multicollinearity or non-invertible $X^T X$

Objective: $\widehat{\mathbf{w}} = \text{argmin} \sum_{i=1}^{m} (f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2 + \lambda \sum_{j=1}^{d} w_j^2$

$= \text{argmin} (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$

Here $\lambda \geq 0$ is a complexity parameter that controls the amount of shrinkage: the larger the value of λ, the greater the amount of shrinkage.

Note: *m* samples & *d* parameters

Ridge Regressio

- large weights mean the model is very sensitive: small changes in the feature cause big jumps in predictions. This is an undesirable overfitting where the model fits noise and performs poorly on new data
- in ridge regression, by adding the last term, we pay a cost for large coefficients. To keep the cost low, the model is "encouraged" to keep the weights small = smoother and more stable models that are less sensitive to noise

Using a linear model:

$$\min_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w}$$

Solution:

$$\frac{\partial}{\partial \mathbf{w}} \left((\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w} \right) = \mathbf{0}$$

$$\Rightarrow 2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y} + 2\lambda\mathbf{w} = \mathbf{0}$$

$$\Rightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$$\Rightarrow (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

where **I** is the *dxd* identity matrix

Here on, we shall focus on single column of output $\mathbf{y}$ in derivations in the sequel

Learning: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1} \mathbf{X}^T\mathbf{y}$

Ref: Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning", (2nd ed., 12th printing) 2017 (chp.3)

Ridge Regression

Ridge Regression in Primal Form (when m > d)

$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})$ is invertible for $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$

Ref: Hastie, Tibshirani, Friedman, "The Elements of Statistical Learning", (2nd ed., 12th printing) 2017 (chp.3)

Ridge Regression

Ridge Regression in Dual Form (when m < d)

$(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})$ is invertible for $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\,\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{X}_{new}) = \mathbf{X}_{new}\hat{\mathbf{w}}$

Derivation as homework (see tutorial 6).

Hint: start off with $(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\mathbf{w} = \mathbf{X}^T\mathbf{y}$ and make use of $\mathbf{w} = \mathbf{X}^T\mathbf{a}$ and $\boldsymbol{a} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w}), \lambda > 0$

Polynomial Regression

Motivation: nonlinear decision surface

- Based on the sum of products of the variables
- E.g. when the input dimension is $d=2$,

a polynomial function of degree = 2 is:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_{11} x_1^2 + w_{22} x_2^2 .$$

No. parameters for n-th order polynomial model with d input variables?

$$\binom{n+d}{d} = \frac{(n+d)!}{n!\,d!}$$

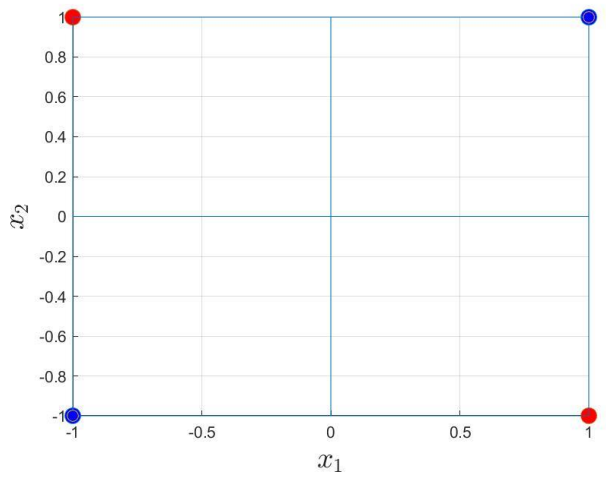
No. parameters for *degree n terms* with d input variables?

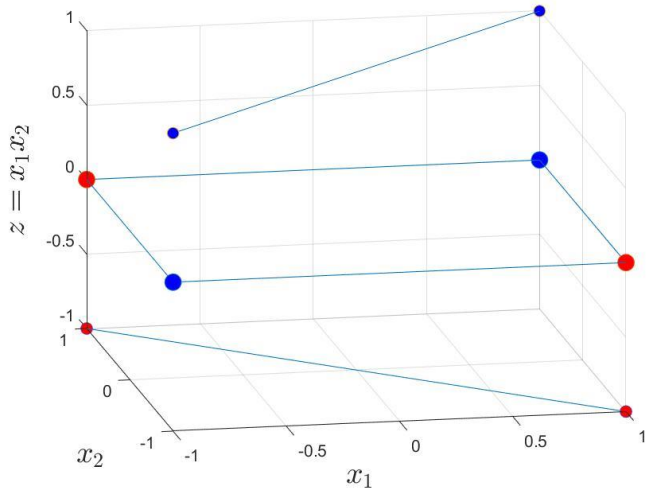
$$\binom{n+d-1}{d-1} = \frac{(n+d-1)!}{n!\,(d-1)!}$$

XOR problem

$$\mathbf{x}_1 = [+1 \quad +1]^\top \qquad y_1 = +1$$
$$\mathbf{x}_2 = [-1 \quad +1]^\top \qquad y_2 = -1$$
$$\mathbf{x}_3 = [+1 \quad -1]^\top \qquad y_3 = -1$$
$$\mathbf{x}_4 = [-1 \quad -1]^\top \qquad y_4 = +1$$

$$f_{\mathbf{w}}(\mathbf{x}) = x_1 x_2$$





Polynomial Regression

Polynomial Expansion

- The linear model $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ can be written as

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$$
$$= \sum_{i=0}^{d} x_i w_i\,, \qquad x_0 = 1$$
$$= w_0 + \sum_{i=1}^{d} x_i w_i\,.$$

equivalent to just polynomial transformation + linear regression

- By including additional terms involving the products of pairs of components of $\mathbf{x}$, we obtain a quadratic model:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i\, x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}\, x_i x_j.$$

2nd order: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_{12}\,x_1x_2 + w_{11}\,x_1^2 + w_{22}\,x_2^2$

3rd order: $f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1x_1 + w_2x_2 + w_{12}\,x_1x_2 + w_{11}\,x_1^2 + w_{22}\,x_2^2 + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk}\,x_ix_jx_k,\ d = 2$

$$C(n,r) = \frac{(n+r-1)!}{r!\,(n-1)!}$$

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Chp.5)

Polynomial Regression

Generalized Linear Discriminant Function

- In general:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i\, x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}\, x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk}\, x_i x_j x_k + \cdots$$

Weierstrass Approximation Theorem: Every continuous function defined on a closed interval [*a*, *b*] can be uniformly approximated as closely as desired by a polynomial function.

- Suppose f is a continuous real-valued function defined on the real interval $[a, b]$.
- For every $\varepsilon > 0$, there exists a polynomial p such that for all x in $[a, b]$, we have $|f(x) - p(x)| < \varepsilon$.

(Ref: https://en.wikipedia.org/wiki/Stone%E2%80%93Weierstrass_theorem)

Notes:

- For high dimensional input features (large *d value*) and high *polynomial order*, the number of polynomial terms becomes explosive! (i.e., grows exponentially)
- For high dimensional problems, polynomials of order larger than 3 is seldom used.

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Chp.5) online

Polynomial Regression

Generalized Linear Discriminant Function

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij} x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} w_{ijk} x_i x_j x_k + \cdots$$

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{P}\mathbf{w} \qquad \text{(Note: } \mathbf{P} \triangleq \mathbf{P}(\mathbf{X}) \text{ for symbol simplicity)}$$

$$= \begin{bmatrix} \boldsymbol{p}_1^T \mathbf{w} \\ \vdots \\ \boldsymbol{p}_m^T \mathbf{w} \end{bmatrix}$$

where $\boldsymbol{p}_l^T \mathbf{w} = [1, x_{l,1}, \ldots, x_{l,d}, \ldots, x_{l,i}x_{l,j}, \ldots, x_{l,i}x_{l,j}x_{l,k}, \ldots] \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \\ \vdots \\ w_{ij} \\ \vdots \\ w_{ijk} \\ \vdots \end{bmatrix}$

$l = 1, \ldots, m$; *d* denotes the dimension of input features; *m* denotes the number of samples

Ref: Duda, Hart, and Stork, "Pattern Classification", 2001 (Chp.5)

Example 3

$\{x_1 = 0, x_2 = 0\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 1\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 0\} \rightarrow \{y = +1\}$
$\{x_1 = 0, x_2 = 1\} \rightarrow \{y = +1\}$

Training set $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$

2nd order polynomial model

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_{12} x_1 x_2 + w_{11} x_1^2 + w_{22} x_2^2$$

$$= \begin{bmatrix} 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_{12} \\ w_{11} \\ w_{22} \end{bmatrix}$$

Python demo 3

Stack the 4 training samples as a matrix

$$\mathbf{P} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,1}x_{1,2} & x_{1,1}^2 & x_{1,2}^2 \\ 1 & x_{2,1} & x_{2,2} & x_{2,1}x_{2,2} & x_{2,1}^2 & x_{2,2}^2 \\ 1 & x_{3,1} & x_{3,2} & x_{3,1}x_{3,2} & x_{3,1}^2 & x_{3,2}^2 \\ 1 & x_{4,1} & x_{4,2} & x_{4,1}x_{4,2} & x_{4,1}^2 & x_{4,2}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Polynomial Regression

Summary

Ridge Regression in Primal Form (m > d)

For $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = (\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})^{-1}\,\mathbf{P}^T\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Ridge Regression in Dual Form (m < d)

For $\lambda > 0$,

Learning: $\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T + \lambda\mathbf{I})^{-1}\,\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Note: Change **X** to **P** with reference to slides 15/16; m & d refers to the size of **P** (not **X**)

Polynomial Regression

Summary

For Regression Applications

- Learn **continuous** valued y using either primal form or dual form
- Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

For Classification Applications

- Learn **discrete** valued y ($y \in \{-1, +1\}$) or $\mathbf{Y}$ (one-hot) using either primal form or dual form
- Binary Prediction: $\hat{f}^{C}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \text{sign}(\mathbf{P}_{new}\hat{\mathbf{w}})$
- Multi-Category Prediction: $\hat{f}^{C}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \arg\max_{k=1,\ldots,C}(\mathbf{P}_{new}\hat{\mathbf{W}}(:,k))$

Example 3 (cont'd)

$\{x_1 = 0, x_2 = 0\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 1\} \rightarrow \{y = -1\}$
$\{x_1 = 1, x_2 = 0\} \rightarrow \{y = +1\}$
$\{x_1 = 0, x_2 = 1\} \rightarrow \{y = +1\}$

Training set

2nd order polynomial model

$$\mathbf{P} = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & x_{1,1}x_{1,2} & x_{1,1}^2 & x_{1,2}^2 \\ 1 & x_{2,1} & x_{2,2} & x_{2,1}x_{2,2} & x_{2,1}^2 & x_{2,2}^2 \\ 1 & x_{3,1} & x_{3,2} & x_{3,1}x_{3,2} & x_{3,1}^2 & x_{3,2}^2 \\ 1 & x_{4,1} & x_{4,2} & x_{4,1}x_{4,2} & x_{4,1}^2 & x_{4,2}^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{y}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 6 & 3 & 3 \\ 1 & 3 & 3 & 1 \\ 1 & 3 & 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \\ -4 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} x_0 \\ \\ \\ \\ \\ x_{4,2} \end{matrix}$$

Python demo 3

Example 3 (cont'd)

Prediction

Test set

Test point 1: $\{x_1 = 0.1, x_2 = 0.1\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$

Test point 2: $\{x_1 = 0.9, x_2 = 0.9\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$

Test point 3: $\{x_1 = 0.1, x_2 = 0.9\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$

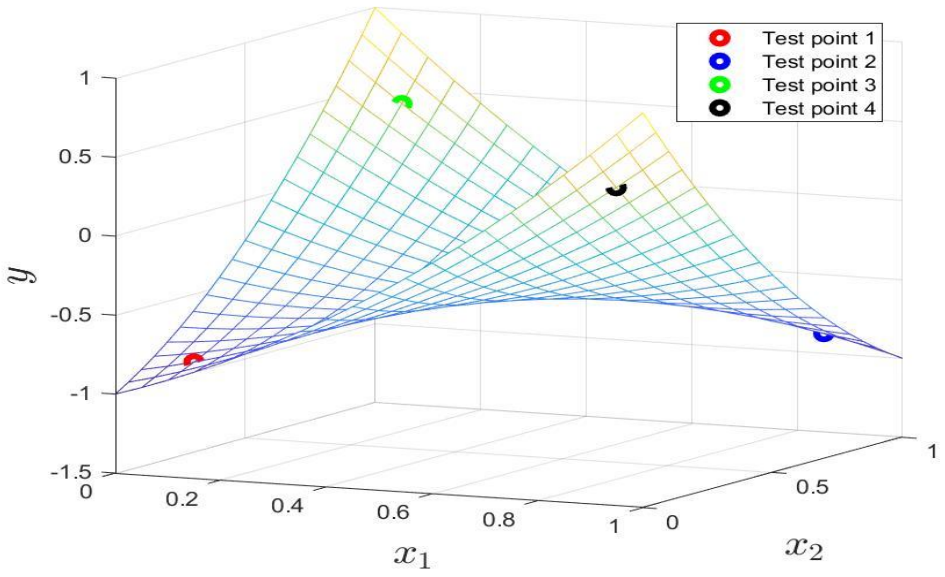
Test point 4: $\{x_1 = 0.9, x_2 = 0.1\} \rightarrow \{y = \text{class } -1 \text{ or } +1?\}$

$$\hat{\mathbf{y}} = \mathbf{P}_{new}\,\hat{\mathbf{w}} = \begin{bmatrix} 1 & 0.1 & 0.1 & 0.01 & 0.01 & 0.01 \\ 1 & 0.9 & 0.9 & 0.81 & 0.81 & 0.81 \\ 1 & 0.1 & 0.9 & 0.09 & 0.01 & 0.81 \\ 1 & 0.9 & 0.1 & 0.09 & 0.81 & 0.01 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ 1 \\ -4 \\ 1 \\ 1 \end{bmatrix}$$

$$[1 \quad x_1 \quad x_2 \quad x_1x_2 \quad x_1^2 \quad x_2^2]$$

$$\hat{f}^{c}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \text{sign}(\hat{\mathbf{y}}) = \text{sign}\left(\begin{bmatrix} -0.82 \\ -0.82 \\ 0.46 \\ 0.46 \end{bmatrix}\right)$$

$$= \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} \begin{array}{l} \dashrightarrow \text{Class } -1 \\ \dashrightarrow \text{Class } -1 \\ \dashrightarrow \text{Class } +1 \\ \dashrightarrow \text{Class } +1 \end{array}$$



Summary

- Notations, Vectors, Matrices
- Operations on Vectors and Matrices
- Systems of Linear Equations $f_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w} = \mathbf{y}$
- Functions, Derivative and Gradient
- Least Squares, Linear Regression with Single and Multiple Outputs
- Learning of vectored function, binary and multi-category classification
- Ridge Regression: penalty term, primal and dual forms
- Polynomial Regression: nonlinear decision boundary

Assignment 1 & 2

Primal form

Learning: $\hat{\mathbf{w}} = (\mathbf{P}^T\mathbf{P} + \lambda\mathbf{I})^{-1}\,\mathbf{P}^T\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Dual form

Learning: $\hat{\mathbf{w}} = \mathbf{P}^T(\mathbf{P}\mathbf{P}^T + \lambda\mathbf{I})^{-1}\,\mathbf{y}$

Prediction: $\hat{f}_{\mathbf{w}}(\mathbf{P}(\mathbf{X}_{new})) = \mathbf{P}_{new}\hat{\mathbf{w}}$

Hint: python packages: sklearn.preprocessing (PolynomialFeatures), np.sign, sklearn.model_selection (train_test_split), sklearn.preprocessing (OneHotEncoder)