EE2211 Introduction to Machine Learning

Lecture 5

Semester 2

2023/2024

Yueming Jin
ymjin@nus.edu.sg

Electrical and Computer Engineering Department
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 (Yueming)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Yueming)
  - 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

Least Squares and Linear Regression

<u>**Module II Contents**</u>

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

Inner product function

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

Affine function

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

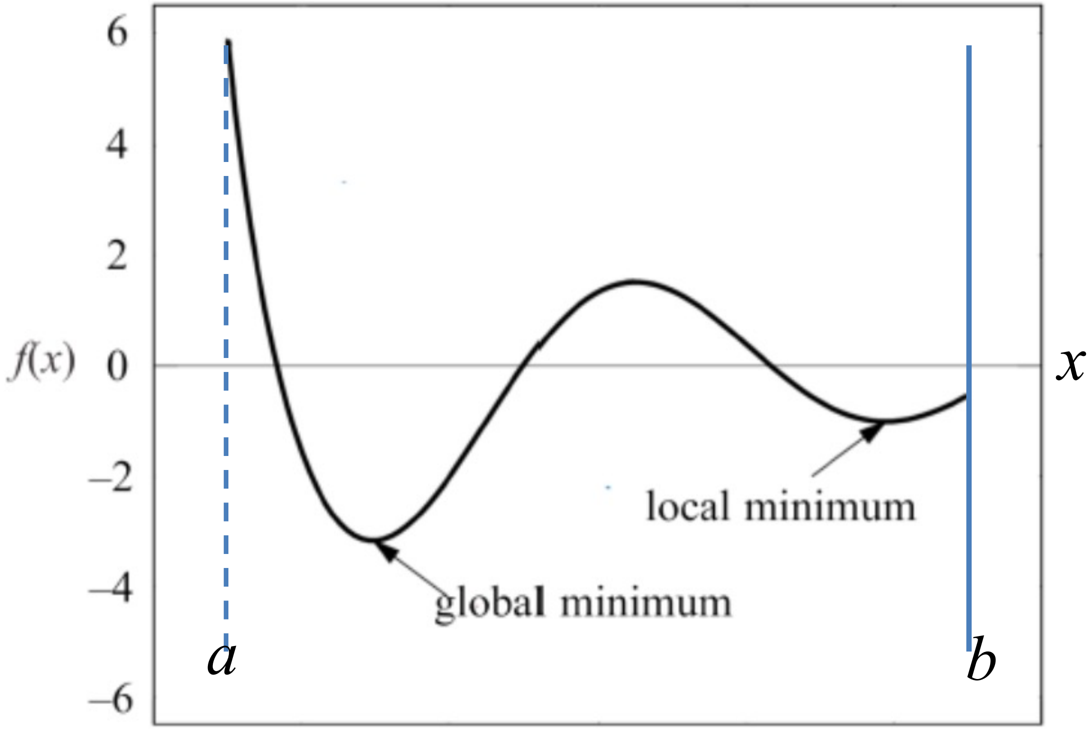
Ref: [Book4] Stephen Boyd and Lieven Vandenberghe, "Introduction to Applied Linear Algebra", Cambridge University Press, 2018 (p31)

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

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

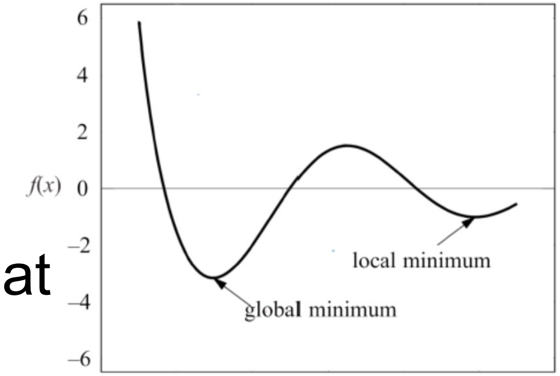
E.g. $f(a) = 3a,\ a \epsilon [0,1] \rightarrow \max_a f(a) = 3$ and $\arg\max_a f(a) = 1$

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

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

$I : R^3 \to R^1$

(x, y, z)

3×1

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

2×1

$\begin{bmatrix} a_{11} x_1 + a_{12} x_2 + a_{13} x_3 \\ a_{21} x_1 + a_{22} x_2 + a_{23} x_3 \end{bmatrix}$

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

3×1 $\quad 2 \times 3$

b^T

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

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

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

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.

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

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

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

Linear Regression

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

- We have a collection of labeled examples (**training**) $\{(\mathbf{x}_i, \mathrm{y}_i)\}_{i=1}^{m}$
  - 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:
    - 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)

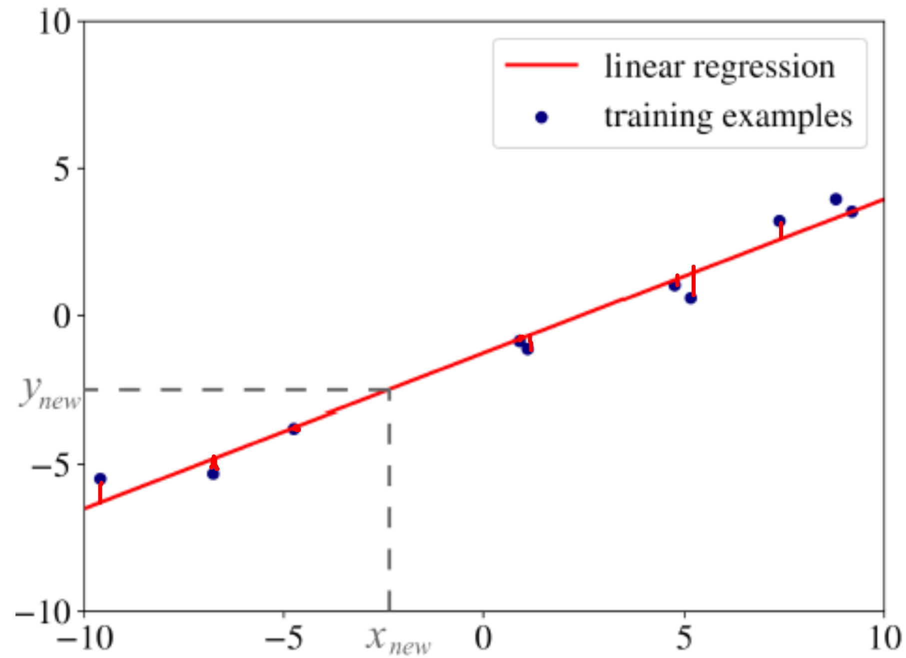
Linear Regression

Learning objective function

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

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

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



$(f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2$ 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:

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

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

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

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

- 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

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

- 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 $i = 1, \dots, m$, a linear model $f_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$ can be stacked as

$$\boldsymbol{f}_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{w} \Longleftrightarrow \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

Learning Model

Learning target vector

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

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

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

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

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

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

$$\underset{\mathbf{X}}{\begin{bmatrix} 1 & -9 \\ 1 & -7 \\ 1 & -5 \\ 1 & 1 \\ 1 & 5 \\ 1 & 9 \end{bmatrix}} \underset{\mathbf{w}}{\begin{bmatrix} w_0 \\ w_1 \end{bmatrix}} = \underset{\mathbf{y}}{\begin{bmatrix} -6 \\ -6 \\ -4 \\ -1 \\ 1 \\ 4 \end{bmatrix}}$$

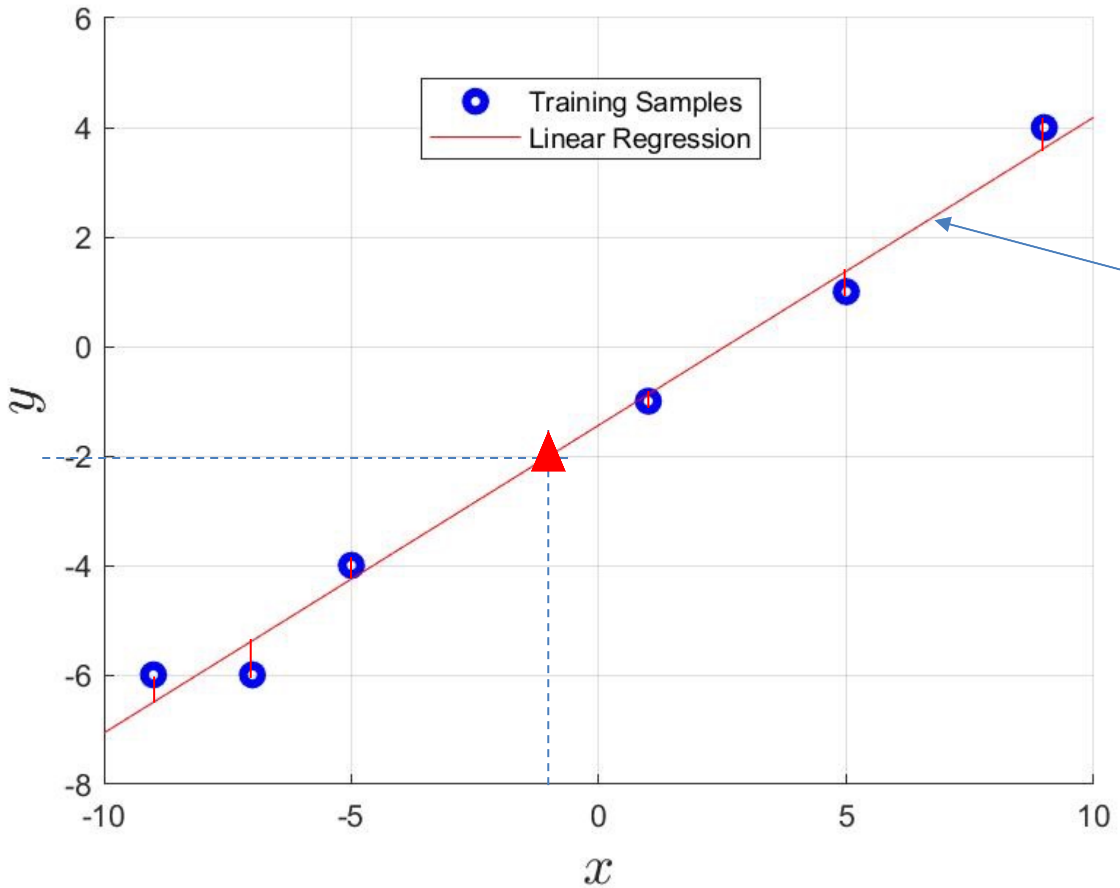
This set of linear equations has no exact solution

However, $\mathbf{X}^T\mathbf{X}$ is invertible **Least square approximation**

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

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

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

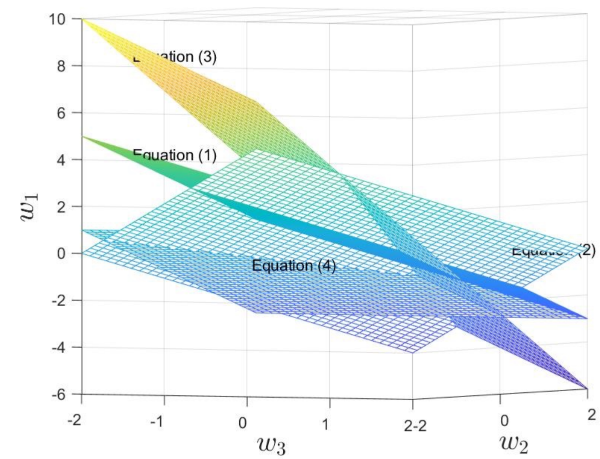
Linear Regression

Example 2 $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$

Training set

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

$$\underset{\mathbf{X}}{\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix}} \underset{\mathbf{w}}{\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}} = \underset{\mathbf{y}}{\begin{bmatrix} 1 \\ 0 \\ 2 \\ -1 \end{bmatrix}}$$



This set of linear equations has no exact solution

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

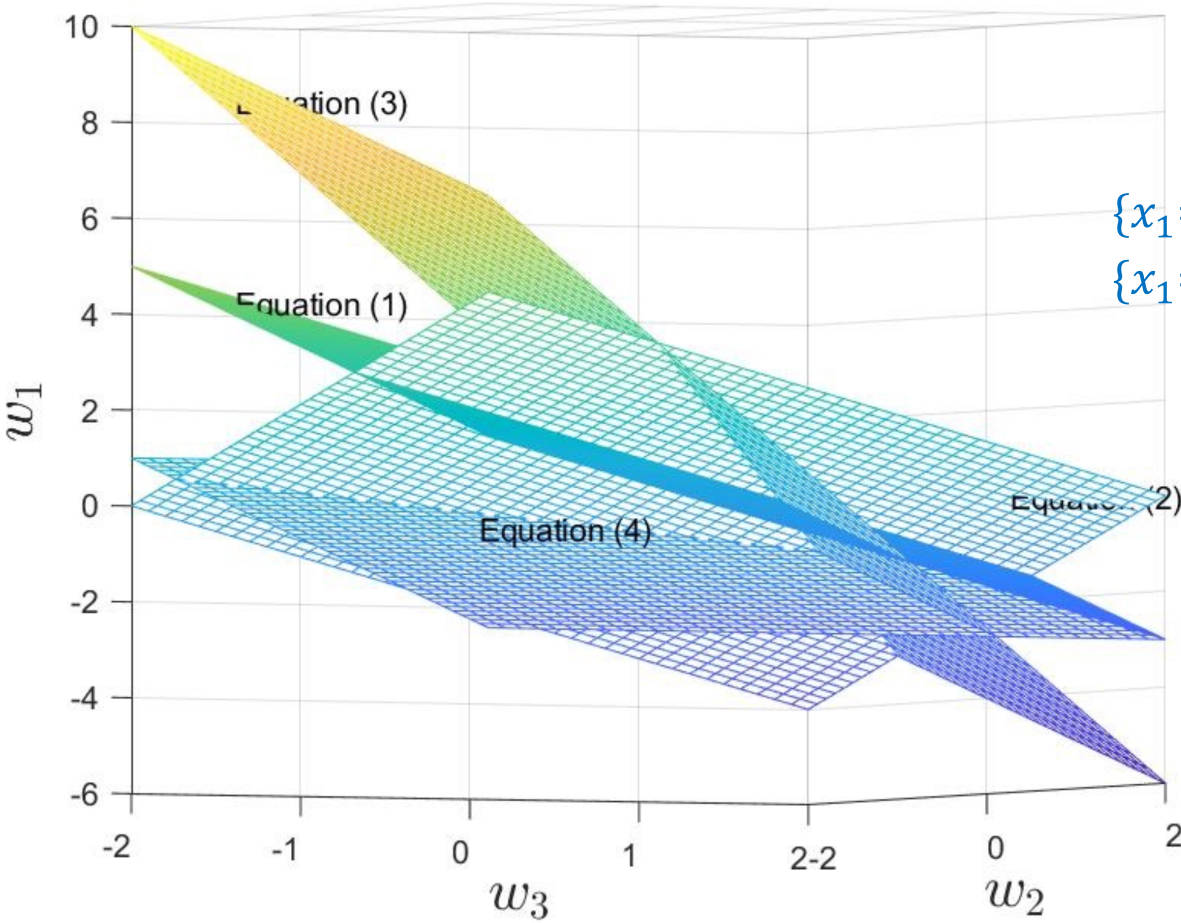
$$\hat{\mathbf{w}} = \mathbf{X}^{\dagger}\mathbf{y} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Least square approximation

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

Linear Regression

The four linear equations



Prediction:

Test set

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

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

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

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

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

Linear Regression

Learning of Vectored Function (Multiple Outputs)

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

For *m* samples: $\mathbf{F}_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{W} = \mathbf{Y}$

$$\begin{array}{l} \text{Sample 1} \dashrightarrow \\ \vdots \\ \text{Sample } m \dashrightarrow \end{array} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix} \mathbf{W} = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & \dots & x_{m,d} \end{bmatrix} \underbrace{\begin{bmatrix} w_{0,1} & \dots & w_{0,h} \\ w_{1,1} & \dots & w_{1,h} \\ \vdots & \ddots & \vdots \\ w_{d,1} & \dots & w_{d,h} \end{bmatrix}}_{h}$$

$$\begin{array}{l} \text{Sample 1's output} \dashrightarrow \\ \vdots \\ \text{Sample } m\text{'s output} \dashrightarrow \end{array} = \underbrace{\begin{bmatrix} y_{1,1} & \dots & y_{1,h} \\ & \vdots & \\ y_{m,1} & \dots & y_{m,h} \end{bmatrix}}_{h} \left.\vphantom{\begin{bmatrix} y \\ \vdots \\ y \end{bmatrix}}\right\} m$$

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

Linear Regression

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

Least Squares Regression of Multiple Outputs

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

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

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

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

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

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

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

Bias

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & 3 \\ 1 & 1 & 0 \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & -1 \\ -1 & 3 \end{bmatrix}}_{\mathbf{Y}}$$

This set of linear equations has NO exact solution

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

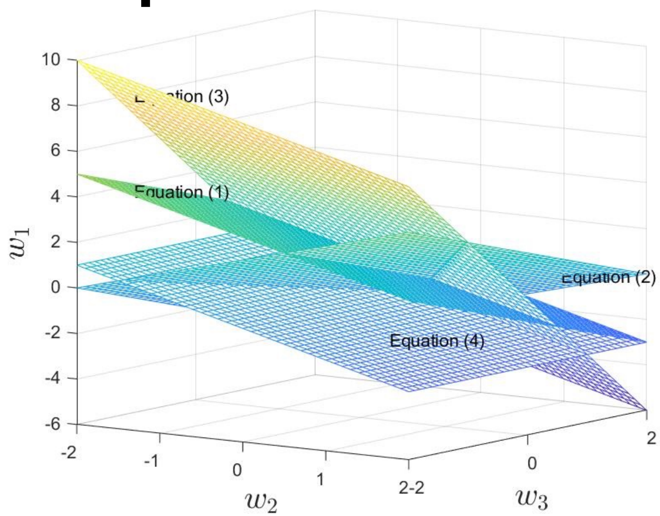
$\mathbf{X}^T\mathbf{X}$ is invertible

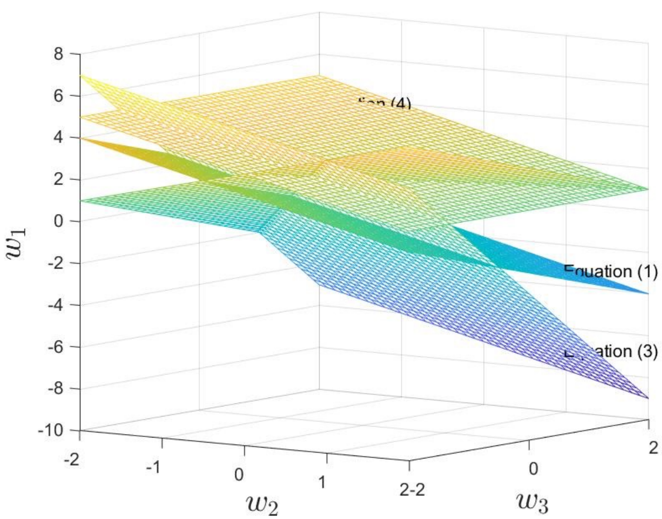
Least square approximation

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

Linear Regression of multiple outputs

Example 3





Prediction:

Test set: two new samples

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

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

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

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]

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

Python demo 3

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

Prediction

Summary

Midterm (L1 to L5)
Trial quiz

- 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

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