EE2211 Pre-Tutorial 5

Dr Feng LIN
feng_lin@nus.edu.sg

Agenda

- Recap
- Self-learning
- Tutorial 5

Today’s Attendance

Recap

- 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

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

Inner product function

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

Linear function is affine, but affine not necessarily to be linear function

Affine function

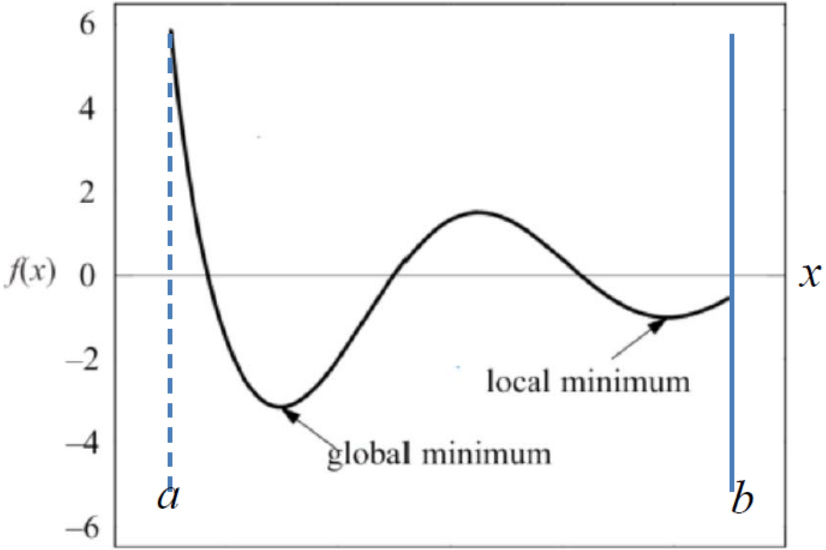
$f(\mathbf{x}) = \boldsymbol{a}^T \mathbf{x} + b$ scalar b is called the offset (or bias)

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 $\mathcal{A} = \{a_1, \ a_2, \dots, \ a_m\}$,
- 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

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

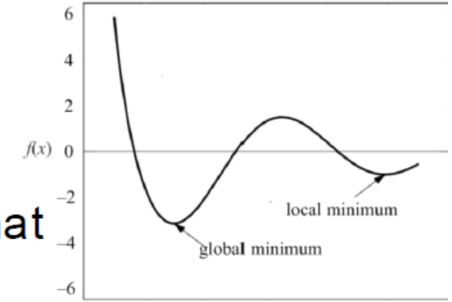
E.g. $f(a) = 3a$, $a \in [0,1]$ → $\max_{a} f(a) = 3$ and $\arg\max_{a} f(a) = 1$

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 $\nabla_{\mathbf{x}} f$.

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 $\mathbf{f}(\mathbf{x})$

Derivative and Gradient

Some Vector-Matrix Differentiation Formulae

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

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

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

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

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.

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

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

Linear Regression

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

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

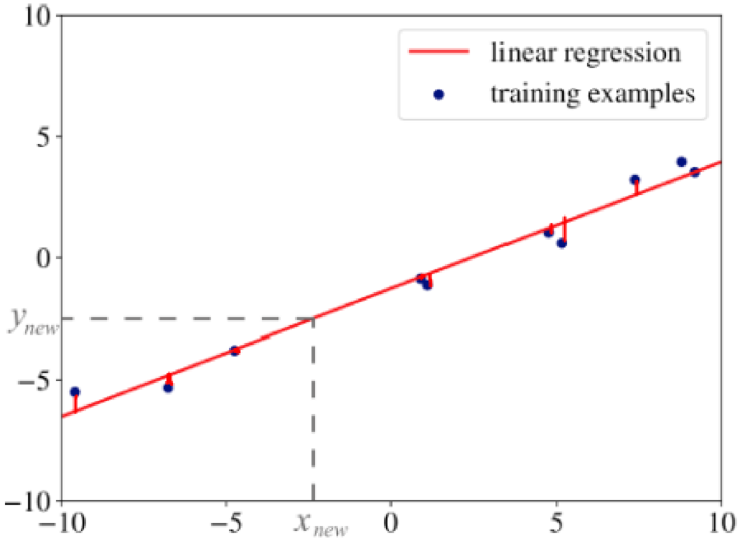
Linear Regression

Learning objective function

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

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

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



$(f_{\mathbf{w}}(\mathbf{x}_i) - \mathrm{y}_i)^2$ 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*.

Linear Regression

Learning objective function (using simplified notation hereon)

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

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

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

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

Learning Model

Learning target vector

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

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

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 $\mathbf{e} = \mathbf{X}\mathbf{w} - \mathbf{y}$:

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

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

Collect Data → Model and Loss Function → Learning/Training: compute w → Prediction /Testing: Compute y_{new}

$$Xw = y$$

- X: Samples
- y: Target values

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

- Linear or Affine function
- Squared error loss function

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

- Check the invertibility
- Least square approximation (left-inverse)

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

- Prediction for new inputs
- Testing: Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

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: $\mathbf{F}_{\mathbf{w}}(\mathbf{X}) = \mathbf{X}\mathbf{W} = \mathbf{Y}$

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

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

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

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 $\mathbf{E} = \mathbf{X}\mathbf{W} - \mathbf{Y}$:

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

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

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

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

THANK YOU