Lecture 15: Linear regression

Textbook: chapter 10

Dr. Huiping Cao

Regression problem

- One type of supervised learning.
- **Regression problem**: predict the value of one or more continuous target variables y given the value of a d-dimensional vector **x**.
  - Output is a real number, which is why the method is called regression
  - **x** are called explanatory variables.
  - y is called response/target variable.
- **Linear regression** is one of the most basic and important technique for predicting a value of an attribute (y).
  - It is used to fit values in a forecasting or predictive model.

Linear regression - example

- A collection of observations of the Old Faithful geyser in the USA Yellowstone National Park.

	eruptions	waiting
1	3.600	79
2	1.800	54
3	3.333	74
4	2.283	62
5	4.533	85
6	2.883	55

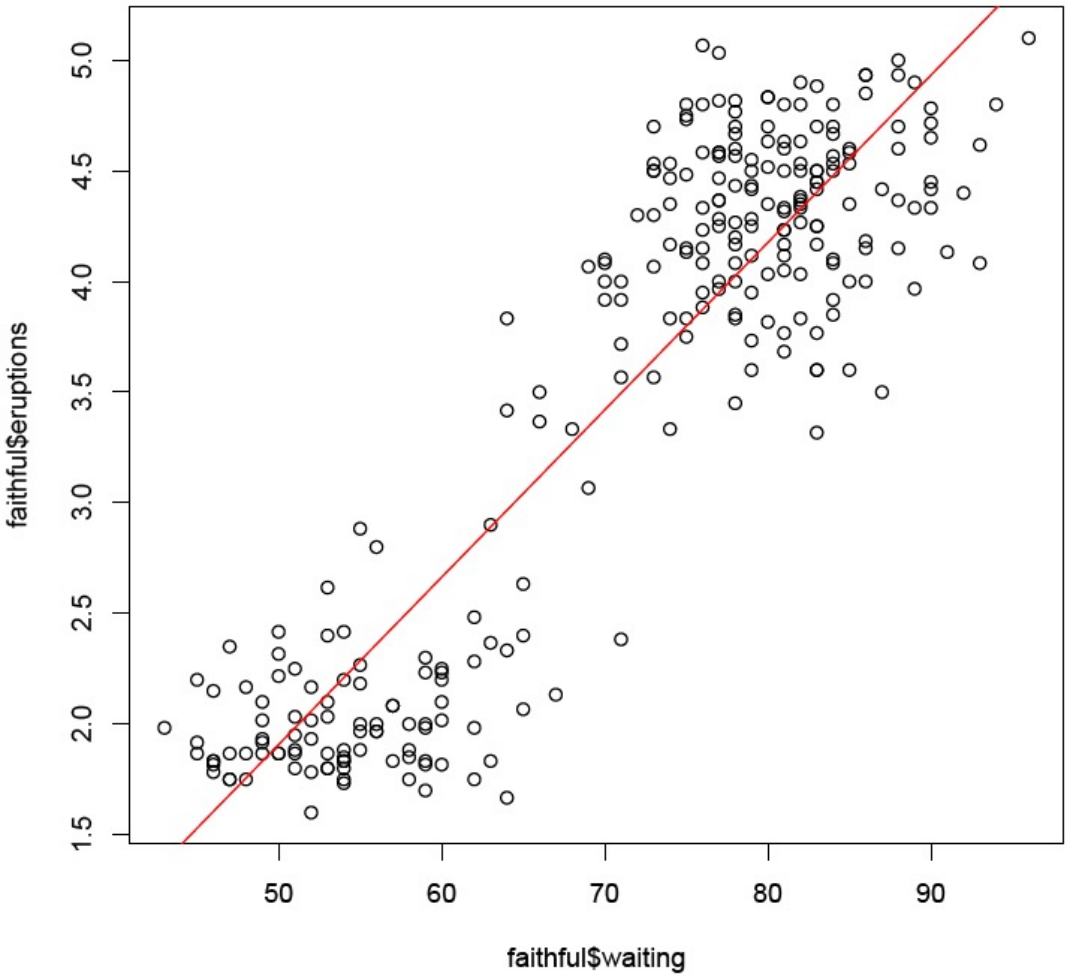
- There are two observation variables in the data set.
  - eruptions: the duration of the geyser eruptions.
  - waiting: the length of waiting period until the next eruption.
- Predict eruption time given waiting time.

Linear regression – representation

- Training set: $\{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$
  - $x^{(i)}$ is a d-dimensional data point (or vector)
  - n is the number of training data points.
  - *d is* number of features
  - $x_j^{(i)}$ is the value of the *j*-th feature of the *i*th instance.
- y = $(y^{(1)}, \ldots, y^{(n)})$ is output
- (x, y): one training example
- $(\mathbf{x}^{(\mathbf{i})}, y^{(i)})$: the i-th training example.
- Hypothesis h(·): The learning algorithm learns the hypothesis. Using h, we can predict y for any given **x**.

Linear regression – How do we represent h

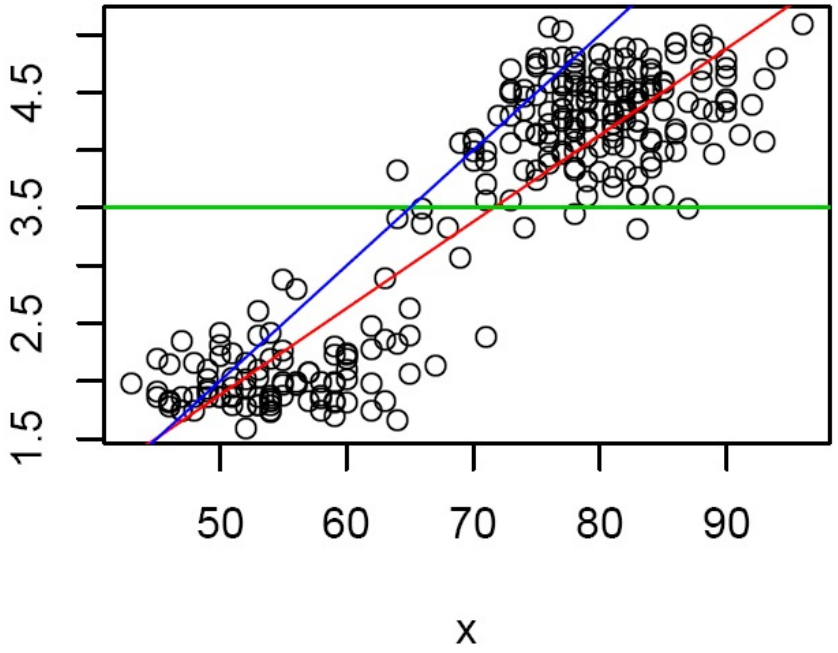
- Linear regression with one variable (univariate linear regression).
- $h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x$ Or,
- $h(\mathbf{x}) = w_0 + w_1 x$
- Linear regression can be understood as finding the best-fitting line through the sample points as shown in the following figure.



Linear regression – cost function

- Hypothesis $h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1 x$.
- How to choose $h_{\mathbf{w}}(\mathbf{x}) = w_0 + w_1$?
- Idea: Choose w_0 and w_1 such that $h_{\mathbf{w}}(\mathbf{x})$ is close to **y** for our training examples.
- Intuitively

$$minimize_{w_0,w_1} \sum_{i=1}^{n} \left(h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$$



Cost function

$$J(w_0, w_1) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$$

- Factor $\frac{1}{n}$ is to average the cost, and the factor $\frac{1}{2}$ is to make the analysis easier.
- J is called **squared error function**, or **cost function**, or squared error cost function.
- There are other cost functions, but squared error cost function is working well.
- Goal: $minimize_{w_0,w_1} J(w_0, w_1)$

Multivariate Linear Regression - motivation and notation

- Example
  - **y:** house prices
  - x_1: house square feet
  - x_2: number of bedrooms
  - x_3: age of home
    etc.

Multivariate Linear Regression - Hypothesis

- $h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) = w_0 + w_1 x_1^{(i)} + \ \ldots + w_d x_d^{(i)}$
  - E.g., $h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) = 80 + 0.1x_1^{(i)} + 0.01x_2^{(i)} + 3x_3^{(i)} - 2x_4^{(i)}$
- For convenience, denote $x_0^{(i)}$ = 1 to define a 0-th feature.
- We can define two vectors **x** and **w**

$$\mathbf{x}^{(i)} = \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \ldots \\ x_d^{(i)} \end{pmatrix} \in \mathbb{R}^{d+1}, \mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \ldots \\ w_d \end{pmatrix} \in \mathbb{R}^{d+1}$$

- $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$

Multivariate Linear Regression - cost function

$$J(w_0, w_1, \dots, w_n) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$$

or

$$J(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2$$

- Parameters: $w_0, w_1, w_2, \dots, w_d$ $\left(\text{or}, \mathbf{w} \in \mathbb{R}^{d+1}\right)$
- Linear regression models: linear functions of the **adjustable parameters**. The simplest form of linear regression models are also linear functions of the **input variables**.

Gradient descent algorithm for multivariate linear regression

- Cost function:

$$J(w) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_{\mathbf{w}}\left(\mathbf{x}_1^{(i)}\right) - y^{(i)}\right)^2$$

- Gradient descent
  - Repeat until convergence {

$$w_j = w_j - \alpha\frac{\partial}{\partial w_j}J(w) \text{ for } j = 0, \ldots, j = d$$

}

Batch gradient descent

Repeat until convergence {

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j} J(\mathbf{w}) = w_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right) \cdot x_j^{(i)}$$

}

$$w_0 = w_0 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right) \cdot x_0^{(i)}$$

$$w_1 = w_1 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right) \cdot x_1^{(i)}$$

$$w_2 = w_2 - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\mathbf{w}}\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right) \cdot x_2^{(i)}$$

Feature scaling

- **Example**: given $0 \leq x_1^{(i)} \leq 3$ and $-1 \leq x_2^{(i)} \leq 1.5$, then adding two features $-200 \leq x_3^{(i)} \leq 200$ and $-0.001 \leq x_4^{(i)} \leq 0.001$ makes the range very different.
- **z-score normalization**, replace Replace $x_j^{(i)}$ with $\frac{x_j^{(i)} - \mu_j}{\sigma_j}$ where σ_j is standard deviation
- **Mean normalization**: Replace $x_j^{(i)}$ with $\frac{x_j^{(i)} - \mu_j}{\max(x_j) - \min(x_j)}$
  - to make features have approximately zero mean (do not apply to $x_0^{(i)} = 1$)
  - E.g., $x_1^{(i)} = \frac{\text{size} - 1000}{2000}, x_2^{(i)} = \frac{\#\ of\ bedrooms - 2}{5}$

Solution – normal equation

- **Normal equation**: method to solve for **w** analytically
- Normal equation method has some advantages and disadvantages, which will be discussed later.
- General case: *n* examples $\left(\mathbf{x}^{(1)}, y^{(1)}\right), (\mathbf{x}^{(2)}, y^{(2)}),\ldots, (\mathbf{x}^{(n)}, y^{(n)})$ and each $\mathbf{x}^{(i)}$ has d features. I.e.,

$$\mathbf{x}^{(i)} = \begin{pmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \ldots \\ x_d^{(i)} \end{pmatrix} \in \mathbb{R}^{d+1}$$

Solution – normal equation

- Construct a $n \times (d + 1)$ matrix X by using the training examples and adding extra feature $x_0^{(i)}$ with values 1

$$X = \begin{pmatrix} --- \left(x^{(1)}\right)^T --- \\ --- \left(x^{(2)}\right)^T --- \\ \cdots \\ --- \left(x^{(n)}\right)^T --- \end{pmatrix} \in \mathbb{R}^{n \times (d+1)}$$

- Construct an n-dimensional vector $\mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \cdots \\ y^{(n)} \end{pmatrix}$
- **Solution:** $\mathbf{w} = (X^T X)^{-1} X^T \mathbf{y}$ (Derivation details can be found from extra materials; omitted here)

Method comparison

Gradient descent	Normal equation
Need to choose α Need many iterations	No need to choose α Don't need to iterate
Works well even when *n* is large	Need to compute $(X^T X)^{-1} \in \mathbb{R}^{n \times n}$ (cost roughly $O(n^3)$) Slow if n is large (e.g., $n \geq 1000$)
Need feature scaling	No need to do feature scaling