00 Linear Regression via Least Squares

June 2, 2021

Information: *Basic concepts and simple examples of linear regression via least squares*

Written by: *Zihao Xu*

Last update date: *06.02.2021*

1 Basic concepts

1.1 Regression

- **Regression** refers to a set of methods for modeling the relationship between one or more independent variables and a dependent variable.
  - In the natural sciences and social sciences, the purpose of regression is most often to **characterize** the relationship between the inputs and outputs
  - In machine learning, it is most often concerned with **prediction**
- Regression problems pop up whenever a prediction for a **numerical value** is wanted.
  - Predicting prices
  - Predicting length of stay
  - Demand forecasting
  - ...
- Mathematical Representation
  - Given n observations consisting of

    $$\mathbf{X}_{1:n} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$$

    which is usually known as *inputs*, *features*, etc. and

    $$\mathbf{y}_{1:n} = \{y_1, y_2, \cdots, y_n\}$$

    which is **continuous** and is usually known as *outputs*, *targets*, etc.
  - The Regression Problem is to use the data to learn the **map** between $\mathbf{x}$ and y

1.2 Linear Regression

- May be both the **simplest** and most **popular** among the standard tools to regression
- A kind of traditional **supervised machine learning**
- **Assumptions**:
  - The relationship between the independent variables $\mathbf{x}$ and the dependent variable y is **linear**
    - That is to say, y can be expressed as a **weighted sum** of the elements in x

    * Or in other words, *Linear regression* models the output as a line ($\dim(\mathbf{x}) = 1$) or hyperplane ($\dim(\mathbf{x} > 1)$)
  - There exist some noise on the observations and any noise is **well-behaved** (following a Gaussian distribution)
- **Linear Regression Model**:
  - The linear regression model is defined by the coefficients (or **parameters**) for each feature
  - For $\mathbf{x} \in \mathbb{R}^d, \mathbf{x} = [x_1, x_2, \cdots, x_d]^T$, denote the parameters to the θ:

$$\hat{y} = f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_d x_d$$

    Let $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \cdots, \theta_d]^T$ and augmented $\tilde{\mathbf{x}} = [1, x_1, x_2, \cdots, x_d]^T$ The the model can be written as

$$\hat{y} = f(x) = \boldsymbol{\theta}^T \tilde{\mathbf{x}}$$

  - This is known as **parametric model**
- **Goal of Linear Regression**:
  - Notice that in **assumption**, we assume that there exist some noises in the observations following a Gaussian distribution. Therefore, we are **not** going to directly solve the equation and expect to get a result $\boldsymbol{\theta}$ that

$$y_i = \boldsymbol{\theta} \tilde{\mathbf{x}}_i \text{ for } 1 \leq i \leq n$$

  - Even when we are confident that the underlying relationship is linear, the noise term should be taken into consideration
  - The goal of linear regression is to find the parameters $\boldsymbol{\theta}$ that **minimize the prediction error**

1.3 Loss function

The most *popular* loss function in regression problems is the **squared error**. - When the prediction for an example i is $\hat{y}_i$ and the corresponding true label is y_i, the squared error is given by

$$l_i(\boldsymbol{\theta}) = (\hat{y}_i - y_i)^2 = \left(\boldsymbol{\theta}^T \tilde{\mathbf{x}}_i - y_i\right)^2$$

- Note: In some notations, there is a $\frac{1}{2}$ term for the convenience of notating derivatives, but it makes no difference as optimization - To measure the quality of a model on the entire dataset of n examples, we simply sum (or **equivalently**, average) the losses on dataset

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} l_i(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(\boldsymbol{\theta}^T \tilde{\mathbf{x}}_i - y_i\right)^2$$

In matrix notation:

$$L(\boldsymbol{\theta}) = \|\mathbf{y} - \tilde{\mathbf{X}}\boldsymbol{\theta}\|_2^2$$

Where $\mathbf{y} = [y_1, y_2, \cdots, y_n]^T \in \mathbb{R}^n$, $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \cdots, \theta_d]^T \in \mathbb{R}^{d+1}$, $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_n]^T \in \mathbb{R}^{n \times (d+1)}$, $\tilde{\mathbf{x}} = [1, x_1, x_2, \cdots, x_d]^T$ - When averaging the losses on dataset, it's called **Mean Squared Error** - Known as **Ordinary Least Squares (OLS)**

1.4 Closed-form solution for OLS

Calculate the gradient of OLS

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} \|\mathbf{y} - \tilde{\mathbf{X}}\boldsymbol{\theta}\|_2^2 &= \nabla_{\boldsymbol{\theta}} \left[\left(\mathbf{y} - \tilde{\mathbf{X}}\boldsymbol{\theta} \right)^T \left(\mathbf{y} - \tilde{\mathbf{X}}\boldsymbol{\theta} \right) \right] \\
&= \left[2 \left(\mathbf{y} - \tilde{\mathbf{X}}\boldsymbol{\theta} \right)^T \cdot \nabla_{\boldsymbol{\theta}} \left[\mathbf{y} - \tilde{\mathbf{X}}\boldsymbol{\theta} \right] \right]^T \\
&= 2 \left(\mathbf{y} - \tilde{\mathbf{X}}\boldsymbol{\theta} \right) \cdot \left(-\tilde{\mathbf{X}} \right)^T \\
&= 2 \left(-\tilde{\mathbf{X}}^T \mathbf{y} + \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \boldsymbol{\theta} \right)
\end{aligned}
$$

Set the gradient to zero *(first-order optimization)* and solve:

$$
\begin{aligned}
2 \left(-\tilde{\mathbf{X}}^T \mathbf{y} + \tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \boldsymbol{\theta}^* \right) &= 0 \\
\tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \boldsymbol{\theta}^* &= \tilde{\mathbf{X}}^T \mathbf{y} \\
\boldsymbol{\theta}^* &= \left(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}} \right)^{-1} \tilde{\mathbf{X}}^T \mathbf{y}
\end{aligned}
$$

- This is known as **normal equation**, finds the regression coefficients **analytically**. - It's an one-step learning algorithm (as opposed to Gradient Descent)

1.5 Normal Equation vs Gradient Descent

1.5.1 Gradient Descent

- Needs to choose GD-based algorithms and set appropriate parameters
- Needs to do a lot of iterations
- Works well with large d (dimension of input data)

1.5.2 Normal Equation

- Gets rid of setting parameters
- Does not need to iterate - compute in one step
- Slow if d is large ($n \leq 10^4$)
- Needs to compute inverse of $\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}$, which is very slow
  - Sometimes use math tricks like *QR factorization* to speed up computation
- Leads to problems if $\tilde{\mathbf{X}}^T \tilde{\mathbf{X}}$ is not invertible

```
[ ]:
```