APL 405: Machine Learning in Mechanics

Lecture 5: Linear regression

by

Rajdip Nayek

Assistant Professor,
Applied Mechanics Department,
IIT Delhi

Instructor email: rajdipn@am.iitd.ac.in

Introduction to basic parametric models

- We introduced the supervised machine learning problem as well as two basic non-parametric methods
  - kNN and Decision Trees
  - Non-parametric methods don't have a fixed set of parameters

- Now we will look at some basic parametric modelling techniques, particularly
  - Linear regression
  - Logistic regression

- **Parametric** model
  - Models that have a **certain defined form** and have **a fixed set of parameters** $\boldsymbol{\theta}$ which are learned from training data
  - Once the parameters are learned, the training data can be discarded, and predictions depend only on $\boldsymbol{\theta}$

Linear Regression

- In both regression and classification settings, we seek a function $f(\mathbf{x}_*)$ that maps the test input $\mathbf{x}_*$ to a prediction

- Regression → learn relationships between some input variables $\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_p]^T$ and a numerical output y
- The inputs can be either categorical or numerical, but let's consider that all p inputs are numerical
- Mathematically, regression is about learning a *model* f that maps the input to the output

$$y = f(\mathbf{x}) + \epsilon$$

- ϵ is an error term that describes everything about the input-output relationship that cannot be captured by the model
- From a statistical perspective, ϵ is considered as a random variable and referred to as **noise**, that is independent of $\mathbf{x}$ and has zero mean

- **Linear regression model**: Output y (a scalar) is an affine combination of p input variables $x_1, x_2, \ldots, x_p$ plus a noise term

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p + \epsilon$$

- $\theta_0, \ \theta_1, \ \theta_2, \ldots, \theta_p$ are called the *parameters* of the model

Linear Regression

- **Linear regression model**: Output y (a scalar) is an affine combination of $p+1$ input variables $1, x_1, x_2, \dots, x_p$ plus a noise term

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_p x_p + \epsilon = \begin{bmatrix} 1 & x_1 & \dots & x_p \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_p \end{bmatrix} + \epsilon = \mathbf{x}^T \boldsymbol{\theta} + \epsilon$$

- $\theta_0, \ \theta_1, \ \theta_2, \dots, \theta_p$ are called the *parameters* of the model

- Symbol $\mathbf{x}$ is used both for the $p+1$ and p-dimensional versions of the input vector, with or without the constant one in the leading position, respectively

- The linear regression model is a **parametric** function of the form $f(\mathbf{x}) = \mathbf{x}^T \boldsymbol{\theta} + \epsilon$

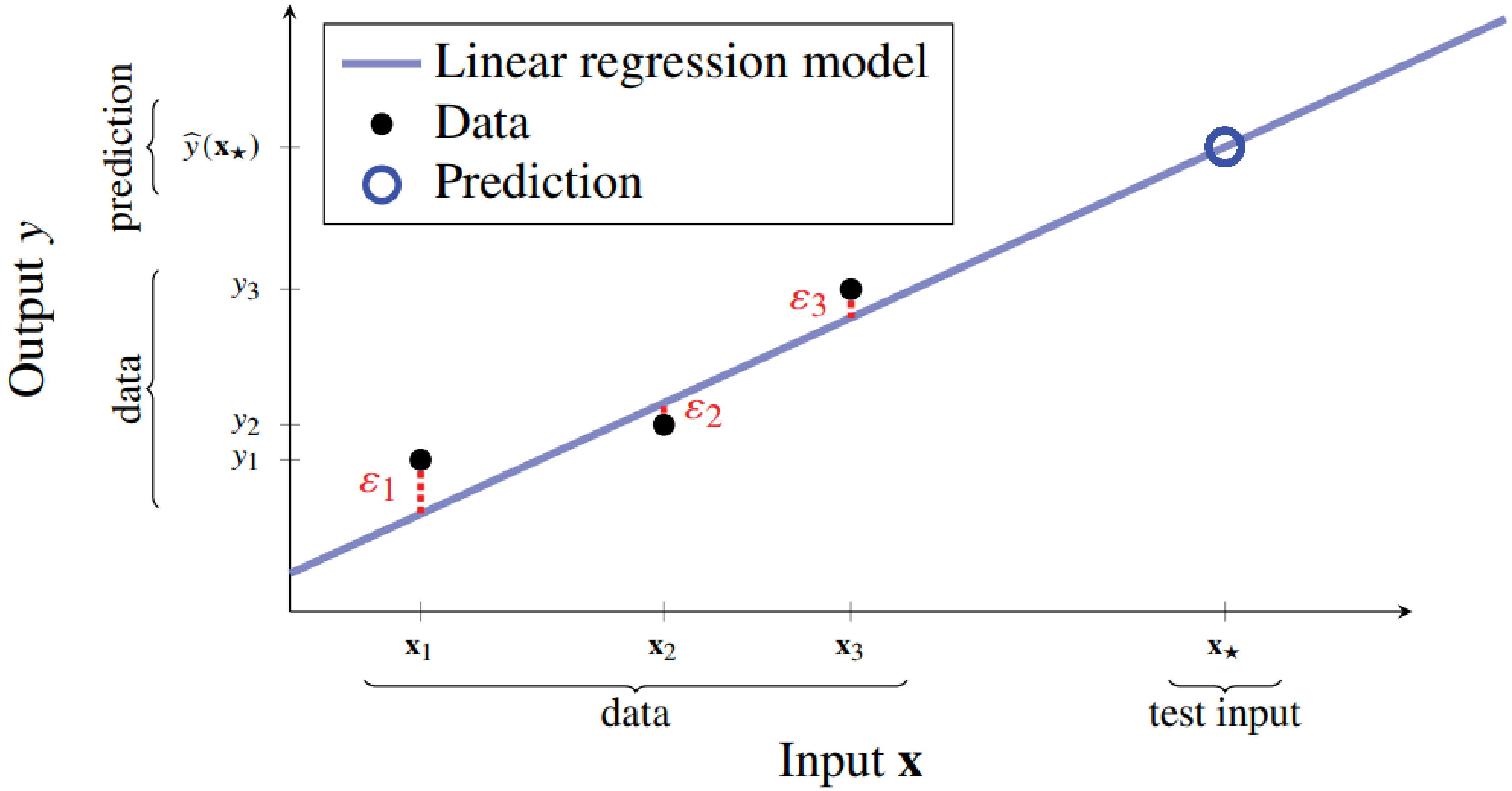
- The parameters $\boldsymbol{\theta}$ can take arbitrary values, and the actual values that we assign to them will control the input–output relationship described by the model

- **Learning of the model** → finding suitable values for $\boldsymbol{\theta}$ based on observed training data

How to predict on test set?

- How to make predictions $\hat{y}(\mathbf{x}_*)$ for new previously unseen test input $\mathbf{x}_* = [1 \quad x_{*,1} \quad x_{*,2} \quad \dots \quad x_{*,p}]^T$?
- Let $\widehat{\boldsymbol{\theta}}$ be the learned parameter value for the linear regression model
- Since the noise term ϵ is random with zero mean and independent of all observed variables, we replace ϵ with 0 in the prediction
- Prediction takes form:

$$\hat{y}(\mathbf{x}_*) = \mathbf{x}_*^T \widehat{\boldsymbol{\theta}}$$



Training a linear regression model from training data

- Training data: $\mathcal{T} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$

$$\boldsymbol{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}\,, \qquad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}, \quad \boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_{*,1} \\ x_{*,2} \\ \vdots \\ x_{*,p} \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

- Here, $\boldsymbol{\epsilon}$ is the vector of noise terms

- Predicted outputs for training data, $\hat{\boldsymbol{y}} = [\hat{y}(\mathbf{x}_1) \quad \hat{y}(\mathbf{x}_2) \quad \cdots \quad \hat{y}(\mathbf{x}_N)]^T$

$$\hat{\boldsymbol{y}} = \mathbf{X}\boldsymbol{\theta}$$

- Learning the unknown parameters $\boldsymbol{\theta}$ amounts to finding their values such that $\hat{\boldsymbol{y}}$ is “similar” to $\boldsymbol{y}$
  - “Similar” → finding $\boldsymbol{\theta}$ such that $\hat{\boldsymbol{y}} - \boldsymbol{y} = \boldsymbol{\epsilon}$ is small

- Formulate a loss function, which gives a mathematical meaning to “similarity” between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$

How to define the problem of learning model parameters?

- Use loss function $L(y, \hat{y}) \rightarrow$ measures the closeness of the model's prediction $\hat{y}$ to the observed data y
  - Smaller the loss, better the model fits the data, and vice versa

- Define average loss (or cost function) function, $J(\boldsymbol{\theta})$, as the average loss over the training data

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} L\big(y_i, \hat{y}(\mathbf{x}_i; \boldsymbol{\theta})\big)$$

- Training a model $\rightarrow$ finding the model parameters $\boldsymbol{\theta}$ that minimize the average training loss

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N} L\big(y_i, \hat{y}(\mathbf{x}_i; \boldsymbol{\theta})\big)$$

- $\hat{y}(\mathbf{x}_i; \boldsymbol{\theta})$ is the model prediction for the $\mathbf{x}_i$ training input and y_i is the corresponding training output
  - The parameter $\boldsymbol{\theta}$ has been put as an argument to denote the dependence of the prediction on it

- The operator $\underset{\boldsymbol{\theta}}{\operatorname{argmin}}$ means 'the value of $\boldsymbol{\theta}$ for which the averaged loss function attains it minimum'

Least squares problem

- For regression, a commonly used loss function is the *squared error* loss

$$L\big(y, \hat{y}(\mathbf{x};\boldsymbol{\theta})\big) = \big(y - \hat{y}(\mathbf{x};\boldsymbol{\theta})\big)^2$$

- This loss function grows quadratically fast as the difference $(y - \hat{y}(\mathbf{x};\boldsymbol{\theta}))$ increases

- The corresponding average loss function (or cost function)

$$J(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\big(y_i - \hat{y}(\mathbf{x}_i;\boldsymbol{\theta})\big)^2 = \frac{1}{N}\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 = \frac{1}{N}\|\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 = \frac{1}{N}\|\boldsymbol{\epsilon}\|_2^2$$

- Here, $\|\cdot\|_2^2$ denotes the square of the Euclidean norm. Due to the square, it is called the least squares cost function

- In linear regression, the learning problem effectively finds the best parameter estimate

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{x}_i{}^T\boldsymbol{\theta})^2 = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N}\|\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2$$

- Closed-form solution exists → $\widehat{\boldsymbol{\theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\boldsymbol{y}$ if $\mathbf{X}^T\mathbf{X}$ is invertible (will be an exercise in HW)

Linear regression algorithm

- Linear regression with squared error loss is very common in practice, due to its closed-form solution
- Other loss functions lead to optimization problems and often lack closed-form solutions

Training using linear regression model

Training Data: $\mathcal{T} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$
Result: Learned parameter vector $\widehat{\boldsymbol{\theta}}$

1. Construct matrix of input features $\mathbf{X}$ and output vector $\boldsymbol{y}$
2. Compute $\widehat{\boldsymbol{\theta}}$ by solving $(\mathbf{X}^T\mathbf{X})\widehat{\boldsymbol{\theta}} = \mathbf{X}^T\boldsymbol{y}$

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}, \quad \boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_{*,1} \\ x_{*,2} \\ \vdots \\ x_{*,p} \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

Testing using linear regression model

Data: Learned parameter vector $\widehat{\boldsymbol{\theta}}$
Result: Prediction $\hat{y}(\mathbf{x}_*)$

1. Compute $\hat{y}(\mathbf{x}_*) = \mathbf{x}_*{}^T\, \widehat{\boldsymbol{\theta}}$

A maximum likelihood perspective of least squares

- **"Likelihood"** refers to a statistical concept of a certain function which describes how likely is that a certain value of $\boldsymbol{\theta}$ has generated the measurements $\boldsymbol{y}$

- Instead of selecting a loss function, one could start with the problem

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, p(\boldsymbol{y}|\mathbf{X};\boldsymbol{\theta})$$

- $p(\boldsymbol{y}|\mathbf{X};\boldsymbol{\theta})$ is the probability density of all observed outputs $\boldsymbol{y}$ in the training data, given all inputs $\mathbf{X}$ and parameters $\boldsymbol{\theta}$

- $p(\boldsymbol{y}|\mathbf{X};\boldsymbol{\theta})$ determines mathematically what 'likely' means

A maximum likelihood perspective of least squares

- **Common assumption**: Noise terms are independent and identically distributed (i.i.d.), each with a Gaussian distribution (also known as a normal distribution) with mean zero and variance σ_ϵ^2

$$\epsilon \sim \mathcal{N}(\epsilon; 0, \sigma_\epsilon^2)$$

- Implies that all observed training data points are independent, and $p(\boldsymbol{y}|\mathbf{X}; \boldsymbol{\theta})$ factorizes out as (prove it)

$$p(\boldsymbol{y}|\mathbf{X}; \boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_i; \boldsymbol{\theta})$$

- The linear regression model, $y = \mathbf{x}^T\boldsymbol{\theta} + \epsilon$, together with i.i.d. Gaussian noise assumption leads to

$$p(y_i|\mathbf{x}_i; \boldsymbol{\theta}) = \mathcal{N}(y_i; \mathbf{x}_i{}^T\boldsymbol{\theta}, \sigma_\epsilon^2) = \frac{1}{\sqrt{2\pi\sigma_\epsilon^2}} \exp\left(-\frac{1}{2\sigma_\epsilon^2}(y_i - \mathbf{x}_i{}^T\boldsymbol{\theta})^2\right)$$

- Recall, we want to maximize the likelihood w.r.t. the parameter $\boldsymbol{\theta}$
- Better to work with logarithm of the likelihood (log-likelihood) to prevent numerical overflow

$$\ln p(\boldsymbol{y}|\mathbf{X}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \ln\big(p(y_i|\mathbf{x}_i; \boldsymbol{\theta})\big)$$

A maximum likelihood perspective of least squares

- Better to work with logarithm of the likelihood (log-likelihood) to prevent numerical overflow

$$\ln p(\boldsymbol{y}|\mathbf{X};\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln\big(p(y_i|\mathbf{x}_i;\boldsymbol{\theta})\big)$$

- Logarithm is a monotonically increasing function, maximizing the loglikelihood is equivalent to maximizing the likelihood

- The linear regression model, $y = \mathbf{x}^T\boldsymbol{\theta} + \epsilon$, together with i.i.d. Gaussian noise assumption leads to

$$\ln p(\boldsymbol{y}|\mathbf{X};\boldsymbol{\theta}) = -\frac{N}{2}\ln(2\pi\sigma_\epsilon^2) - \frac{1}{2\sigma_\epsilon^2}\sum_{i=1}^{N}(y_i - \mathbf{x}_i{}^T\boldsymbol{\theta})^2$$

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \ln p(\boldsymbol{y}|\mathbf{X};\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left(-\sum_{i=1}^{N}(y_i - \mathbf{x}_i{}^T\boldsymbol{\theta})^2\right) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{x}_i{}^T\boldsymbol{\theta})^2$$

- Recall the same estimate is also obtained from linear regression with the least squares cost
- Using squared error loss is equivalent to assuming a Gaussian noise distribution in maximum likelihood formulation
- Other assumptions on ϵ lead to other loss functions (will discuss later)

How to handle categorical input variables?

- We had mentioned earlier that input variables $\mathbf{x}$ can be numerical, categorical, or mixed

- Assume that an input variable is categorical and takes only two classes, say $\mathbf{A}$ and $\mathbf{B}$

- We can represent such an input variable x using 1 and 0

$$x = \begin{cases} 0, & \text{if } \mathbf{A} \\ 1, & \text{if } \mathbf{B} \end{cases}$$

- For linear regression, the model effectively looks like

$$y = \theta_0 + \theta_1 x + \epsilon = \begin{cases} \theta_0 + \epsilon, & \text{if } \mathbf{A} \\ \theta_0 + \theta_1 + \epsilon, & \text{if } \mathbf{B} \end{cases}$$

- If the input is a categorical variable with more than two classes, let's say $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$, use one-hot encoding

$$\mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{if } \mathbf{A}, \quad \mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \text{if } \mathbf{B}, \quad \mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{if } \mathbf{C}, \quad \mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \text{if } \mathbf{D}$$