KNTU_IPM Machine learning

a. Translate 10 real-world problem into ML concepts

   (create code for the model)

b. Use proper mathematical notation for those concepts

c. Justify your answers ;

   supervised or unsupervised?

   Regression or classification?

   Learning to explain or learning to predict?

d. Include cost function minima and gradient descent to the problems at hand.

Send your answers to:
tel : @elysianv13
email : armingh186@gmail.com

Example :

House Price Prediction

Problem Description:

The task is to predict the price of a house based on features such as the number of bedrooms, square footage, and location. This is a common real-world problem in real estate, often used in ML tutorials.

ML Concept Translation:

This fits supervised learning, as historical data with known prices (labels) is used to train the model. It is a regression problem, as the output (price) is continuous.

Mathematical Notation:
The model can be represented as

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n + \epsilon$$

, where y is the predicted price, xi are features, θi are parameters, and ϵ is the error term. This is standard for linear regression.

Supervised or Unsupervised:
Supervised, as we have input-output pairs (features and prices).

Regression or Classification:
Regression, given the continuous nature of prices.

Learning to Explain or Predict:
Both. The coefficients θi can explain how each feature affects the price (e.g., more bedrooms increase price), and the model predicts new house prices for unseen data.

Justification:
The use of labeled data aligns with supervised learning, and the continuous output (price) is characteristic of regression. The dual purpose of explanation and prediction is common in such models, where feature importance can guide real estate strategies.

Cost Function and Gradient Descent:
The cost function is typically the mean squared error (MSE):

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

where hθ(x) is the hypothesis. Gradient descent minimizes this by updating parameters:

good luck 😊

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$