COMP4211
Machine Learning

Notes

Insturctor: Dit Yan, Yeung

Fall 2025

Edited by

ThunderDora

The Hong Kong University of Science and Technology

Last Edited: ***July 21, 2025***

Contents

1 Machine Learning

1.1 What is Machine Learning?

> ★ **Definition**
> Machine Learning is the science of **making computer artifacts improve their performance** with respect to a certain performance criterion using example data without requring humans to explicitly program the rules for improvement.

Machine Learning problems can be categorized into three main types:

- **Supervised Learning**: Algorithm that learns from a labeled dataset, where the input data corresponds to the accurate output.
- **Unsupervised Learning**: Model that is developed using an unlabeled dataset, in which the input data lacks associated output labels.
- **Reinforcement Learning**: Algorithm that acquires knowledge through engagement with an environment, obtaining feedback as rewards or penalties depending on its actions.

1.2 What is Supervised Learning?

The fundamental steps of supervised learning is outlined below:

1. Consider a training set $S = \{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{N}$ consisting of N labeled pairs of inputs and outputs.
2. Identify a function $f(\cdot)$ with the training set S so that $f(\mathbf{x}^{(l)}) \approx \mathbf{y}^{(l)}$ holds for all $l = 1, \ldots, N$, and that $f(\mathbf{x})$ for new examples $\mathbf{x}$ also results in $f(\mathbf{x}) \approx \mathbf{y}$ from the same distribution.
3. In the testing stage, examples without labels containing only the input $\mathbf{x}$ are given, and the model determines the output $\mathbf{y}$ by applying the learned function $f(\mathbf{x})$.

Supervised learning is generally applied to address two kinds of issues:

- **Classification**: The output $\mathbf{y}$ is a **categorical** value, like a label class.
- **Regression**: The output $\mathbf{y}$ is a **continuous** quantity, like a real number.

1.3 What is Unsupervised Learning?

The basic steps of unsupervised learning are detailed below:

1. Take a training set $S = \{\mathbf{x}^{(l)}\}_{l=1}^{N}$ that contains N inputs without labels.
2. Determine a function $f(\cdot)$ using the training set S such that $f(\mathbf{x}^{(l)})$ represents the essential pattern of the data

Unsupervised learning is typically utilized to tackle four types of problems:

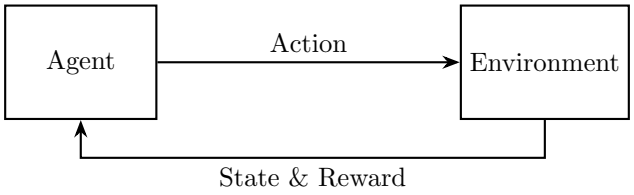
- **Clustering**: Method that organizes the data into clusters by their similarities.
- **Dimensionality Reduction**: Algorithm that minimizes the number of attributes while maintaining crucial information.
- **Anomaly Detection**: Algorithm that detects atypical patterns that deviate from anticipated behavior.
- **Density Estimation**: Method that calculates the likelihood distribution of the data.

1.4 What is Reinforcement Learning?

Reinforcement learning is a form of machine learning in which an agent learns to make choices by performing actions in an environment to optimize total rewards. The essential elements of reinforcement learning are:

- **Agent**: Entity making decisions or learning by engaging with the environment.
- **Environment**: Outside system that the agent engages with.
- **State**: Depiction of the existing circumstances of the environment.
- **Action**: Decision made by the agent that influences the condition of the environment.
- **Reward**: Feedback signal obtained by the agent following an action, showing how effective or ineffective that action was.

The reinforcement learning process entails the agent noticing the existing state of the environment, choosing an action guided by a policy, obtaining a reward, and refining its policy to enhance future actions.



2 Linear Regression

2.1 What is Regression?

★ Definition
Regression is a statistical modeling approach used to estimate the **relationship between a dependent variable and one or several independent variables that may contain errors**. The aim of regression is to identify the most suitable line or curve that represents the connection between the variables.

The fundamental concept of regression can be illustrated through the following steps:

1. Consider a training set $S = \{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{N}$ which comprises N pairs of inputs and corresponding outputs, with $\mathbf{x}^{(l)}$ representing the input and $\mathbf{y}^{(l)}$ denoting the output.
2. The input $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ represents a vector in d dimensions, with d denoting the count of features or attributes.
3. **Regression Function** $f(\cdot; \mathbf{w})$ employs S to ensure the predicted outcome $f(\mathbf{x}^{(l)}; \mathbf{w})$ is as near as feasible to the true output $\mathbf{y}^{(l)}$ for every $l = 1, \ldots, N$, with $\mathbf{w}$ representing the parameter vector of the regression function.
4. When the output $\mathbf{y}$ consists of a multivariate vector, it represents a **multi-output regression** issue. For a univariate result, it constitutes a **single-output regression** issue.

2.2 Linear Regression Function

If the regression function $f(\mathbf{x}; \mathbf{w})$ is **linear**, it can be expressed as:

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_d x_d = \begin{bmatrix} w_0 & w_1 & w_2 & \ldots & w_d \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \mathbf{w}^T \tilde{\mathbf{x}} = \tilde{\mathbf{x}}^T \mathbf{w} \tag{2.1}$$

- Weight w_0 represents the **bias** term, which is a fixed value that acts as an adjustment for the regression function.
- Learning problem involves identifying the optimal parameter vector $\mathbf{w} = (w_0, w_1, \ldots, w_d)$ that reduces the discrepancy between the predicted output $f(\mathbf{x}^{(l)}; \mathbf{w})$ and the actual output $\mathbf{y}^{(l)}$ for every $l = 1, \ldots, N$.

2.3 Squared Loss Function

To determine the parameter vector $\mathbf{w}$ of the linear regression function $f(\mathbf{x}; \mathbf{w})$, it is essential to establish a loss function $L(\mathbf{w}; S)$ that measures the discrepancy between the predicted output and the true output. The loss function most frequently utilized for linear regression is the **squared loss function**, defined as:

$$L(\mathbf{w}; S) = \sum_{l=1}^{N} \left(f(\mathbf{x}^{(l)}; \mathbf{w}) - \mathbf{y}^{(l)} \right)^2 = \sum_{l=1}^{N} \left(w_0 + w_1 x_1^{(l)} + w_2 x_2^{(l)} + \ldots + w_d x_d^{(l)} - \mathbf{y}^{(l)} \right)^2 \tag{2.2}$$

The **mean squared error** (MSE) can be defined as the average of the squared loss function across all N training samples:

$$\text{MSE}(\mathbf{w}; S) = \frac{1}{N} L(\mathbf{w}; S) = \frac{1}{N} \sum_{l=1}^{N} \left(f(\mathbf{x}^{(l)}; \mathbf{w}) - \mathbf{y}^{(l)} \right)^2 \tag{2.3}$$

The squared loss function has two scenarios: $d = 1$ and $d > 1$, with d representing the count of features in the input $\mathbf{x}$.

2.3.1 Special Case: Single-Output Regression ($d = 1$)

Considering the squared loss function for single-output regression, the loss function can be represented as:

$$\boxed{L(w_0, w_1; S) = \sum_{l=1}^{N} \left(w_0 + w_1 x^{(l)} - y^{(l)}\right)^2} \tag{2.4}$$

The distinctive optimal solution $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$ can be obtained by minimizing the loss function $L(w_0, w_1; S)$ through the least squares technique.

To find the optimal solution, one can calculate the partial derivatives of the loss function concerning w_0 and w_1, equate them to zero, and resolve the resulting equations:

$$\frac{\partial L}{\partial w_0} = 2\sum_{l=1}^{N} \left(w_0 + w_1 x^{(l)} - y^{(l)}\right) \cdot 1 = 0 \implies \sum_{l=1}^{N} \left(w_0 + w_1 x^{(l)}\right) = \sum_{l=1}^{N} y^{(l)} \implies N w_0 + w_1 \sum_{l=1}^{N} x^{(l)} = \sum_{l=1}^{N} y^{(l)}$$

$$\frac{\partial L}{\partial w_1} = 2\sum_{l=1}^{N} \left(w_0 + w_1 x^{(l)} - y^{(l)}\right) x^{(l)} = 0 \implies w_0 \sum_{l=1}^{N} x^{(l)} + w_1 \sum_{l=1}^{N} x^{(l)^2} = \sum_{l=1}^{N} y^{(l)} x^{(l)}$$

We have a system of linear equations consisting of two equations and two unknown variables, which can be represented in matrix notation as:

$$\boxed{\mathbf{A}\mathbf{w} = \begin{bmatrix} N & \sum_{l=1}^{N} x^{(l)} \\ \sum_{l=1}^{N} x^{(l)} & \sum_{l=1}^{N} x^{(l)^2} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} \sum_{l=1}^{N} y^{(l)} \\ \sum_{l=1}^{N} y^{(l)} x^{(l)} \end{bmatrix} = \mathbf{b}} \tag{2.5}$$

Assuming the matrix $\mathbf{A}$ is invertible, the optimal solution can be found by multiplying both sides of the equation by $\mathbf{A}^{-1}$:

$$\boxed{\mathbf{w} = \mathbf{A}^{-1}\mathbf{b}} \tag{2.6}$$

2.3.2 General Case: Multi-Output Regression ($d > 1$)

In multi-output regression, two methods can be used to identify the best solution.

The initial strategy is to utilize the **least squares method**. Initially, we represent the input and output sections of the N examples as:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times (d+1)} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} y_1^{(1)} \\ y_2^{(1)} \\ \vdots \\ y_d^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times d}$$

We obtain a system of $d + 1$ linear equations involving $d + 1$ unknowns, which can be represented in matrix form as:

$$\boxed{\mathbf{A}\mathbf{w} = (\mathbf{X}^T\mathbf{X})\mathbf{w} = \mathbf{X}^T\mathbf{Y} = \mathbf{b}} \tag{2.7}$$

Assuming the matrix $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ is invertible, one can determine the optimal solution by multiplying both sides of the equation by $\mathbf{A}^{-1}$:

$$\boxed{\mathbf{w} = \mathbf{A}^{-1}\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}} \tag{2.8}$$

An alternative method is to differentiate using **Multivariable Calculus**. Initially, we can represent $\mathbf{Xw} - \mathbf{Y}$ as:

$$\mathbf{Xw} - \mathbf{Y} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} = \begin{bmatrix} w_0 + w_1 x_1^{(1)} + w_2 x_2^{(1)} + \ldots + w_d x_d^{(1)} - y^{(1)} \\ w_0 + w_1 x_1^{(2)} + w_2 x_2^{(2)} + \ldots + w_d x_d^{(2)} - y^{(2)} \\ \vdots \\ w_0 + w_1 x_1^{(N)} + w_2 x_2^{(N)} + \ldots + w_d x_d^{(N)} - y^{(N)} \end{bmatrix}$$

The squared loss function can be represented as the **squared L-2 norm of the vector** $\mathbf{Xw} - \mathbf{Y}$:

$$\boxed{L(\mathbf{w}; S) = \|\mathbf{Xw} - \mathbf{Y}\|^2} \tag{2.9}$$

Then we can express the squared loss function as:

$$L(\mathbf{w}; S) = (\mathbf{Xw} - \mathbf{Y})^T (\mathbf{Xw} - \mathbf{Y}) = \mathbf{w}^T \mathbf{X}^T \mathbf{Xw} - 2\mathbf{Y}^T \mathbf{Xw} + \mathbf{Y}^T \mathbf{Y}$$

To reduce the squared loss function, we can calculate the gradient of $L(\mathbf{w}; S)$ concerning $\mathbf{w}$ and equate it to zero:

$$\begin{aligned} \nabla_{\mathbf{w}} L(\mathbf{w}; S) = 2\mathbf{X}^T \mathbf{Xw} - 2\mathbf{X}^T \mathbf{Y} = 0 &\implies \mathbf{X}^T \mathbf{Xw} = \mathbf{X}^T \mathbf{Y} \\ &\implies \mathbf{w} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{Y} \end{aligned}$$

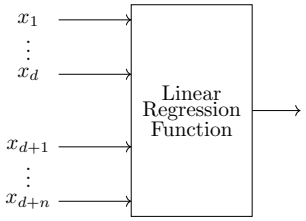
To find the optimal solution $\mathbf{w}$, we must compute the inverse of the matrix $\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{(d+1)\times(d+1)}$. We can apply the **LeGall algorithm**, recognized as the quickest identified method for matrix multiplication, exhibiting a **time complexity of** $O(n^{2.3728596})$ for an $n \times n$ matrix, in contrast to the straightforward $O(n^3)$ approach used in Cholesky decomposition, LU decomposition, or Gaussian elimination.

2.4 Nonlinear Extension of Linear Regression

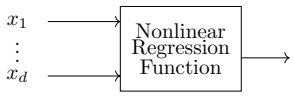
Nonlinear regression functions are crucial because more complex issues cannot be addressed by linear regression. To tackle this, we can modify the linear regression function $f(\mathbf{x}; \mathbf{w})$ into a nonlinear format by adding nonlinear transformations of the input features. Three distinct methods exist for nonlinear expansions to accomplish this:

1. **Feature Engineering**: Intentionally generate additional input dimensions (nonlinearly related to the original input) and utilize linear regression on the newly created input dimensions. For instance, if the original input is $\mathbf{x} = (x_1, x_2)$, we can generate additional features like x_1^2, x_2^2, and $x_1 x_2$ to create an updated input $\tilde{\mathbf{x}} = (1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)$.

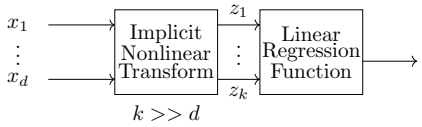
   The key benefit is that linear regression remains applicable, and the weights in the model are **highly interpretable** (the greater the weight's magnitude, the more significant the feature). The primary drawback is that it necessitates domain expertise to develop new features and can be costly in terms of computation if the feature count is high.



2. **Explicit Mapping**: Apply an explicitly defined **nonlinear regression function** $f(\mathbf{x}; \mathbf{w})$ that is nonlinear in the input $\mathbf{x}$, such as polynomial regression, radial basis function (RBF) regression, or neural networks. The model learns the parameters $\mathbf{w}$ of the nonlinear function directly from the training data.



3. **Implicit Mapping**: Implement an **implicitly defined nonlinear transformation** on the original input, followed by utilizing a linear regression function on the modified input. This method employs kernel techniques or neural networks with latent layers to automatically acquire valuable feature representations without their explicit specification. For instance, kernel techniques such as Support Vector Machines (SVM) implicitly transform the input into a higher-dimensional space via kernel functions.



2.5 Polynomial Regression

Polynomial Regression introduces higher-order polynomial terms as the additional input features in the linear regression function. The polynomial regression function can be expressed as:

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x + \ldots + w_m x^m = \begin{bmatrix} w_0 & w_1 & \ldots & w_m \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \vdots \\ x^m \end{bmatrix} = \mathbf{w}^T \tilde{\mathbf{x}} = \tilde{\mathbf{x}}^T \mathbf{w} \tag{2.10}$$

- The parameter vector $\mathbf{w} = (w_0, w_1, \ldots, w_m)$ consists of $m + 1$ parameters, where m is the degree of the polynomial.
- The input $\tilde{\mathbf{x}} = (1, x, x^2, \ldots, x^m)$ represents the polynomial features derived from the original input x.