

---

---

# COMP4211

## Machine Learning

NOTES

---

---

INSTURCTOR: DIT YAN, YEUNG  
FALL 2025

EDITED BY  
THUNDERDORA

*The Hong Kong University of Science and Technology*

*Last Edited: August 5, 2025*

## Contents

<b>1</b>	<b>Machine Learning</b>	<b>2</b>
1.1	What is Machine Learning? . . . . .	2
1.2	What is Supervised Learning? . . . . .	2
1.3	What is Unsupervised Learning? . . . . .	2
1.4	What is Reinforcement Learning? . . . . .	3
<b>2</b>	<b>Linear Regression</b>	<b>4</b>
2.1	What is Regression? . . . . .	4
2.2	Linear Regression Function . . . . .	4
2.3	Squared Loss Function . . . . .	4
2.3.1	Special Case: Single-Output Regression ( $d = 1$ ) . . . . .	5
2.3.2	General Case: Multi-Output Regression ( $d > 1$ ) . . . . .	5
2.4	Nonlinear Extension of Linear Regression . . . . .	6
2.5	Polynomial Regression . . . . .	7
2.6	Regularization . . . . .	7
2.6.1	Choice of Regularization Parameter $\lambda$ . . . . .	8
2.6.2	Other Regularization Techniques . . . . .	9
2.7	Common Performace Metrics for Regression . . . . .	9
2.7.1	Mean Squared Error (MSE) . . . . .	9
2.7.2	R-squared ( $R^2$ ) Score . . . . .	10

# 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 requiring 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  $l = 1, \dots, N$ , 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:

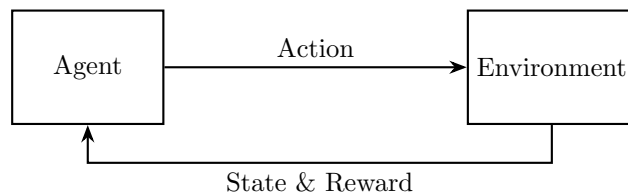
- **Clustering:** Method that organizes the data into clusters by their similarities.
- **Dimensionality Reduction:** Algorithm that minimizes the number of attributes while maintaining crucial information.
- **Anomaly Detection:** Algorithm that detects atypical patterns that deviate from anticipated 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  $\mathbf{x} = (x_1, x_2, \dots, x_d)$  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  $l = 1, \dots, N$ , 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:

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1x_1 + w_2x_2 + \dots + w_dx_d = \begin{bmatrix} w_0 & w_1 & w_2 & \dots & w_d \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \mathbf{w}^T \tilde{\mathbf{x}} = \tilde{\mathbf{x}}^T \mathbf{w} \quad (2.1)$$

- 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  $\mathbf{w} = (w_0, w_1, \dots, w_d)$  that reduces the discrepancy between the predicted output  $f(\mathbf{x}^{(l)}; \mathbf{w})$  and the actual output  $\mathbf{y}^{(l)}$  for every  $l = 1, \dots, N$ .

### 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:

$$L(\mathbf{w}; S) = \sum_{l=1}^N \left( f(\mathbf{x}^{(l)}; \mathbf{w}) - \mathbf{y}^{(l)} \right)^2 = \sum_{l=1}^N \left( w_0 + w_1x_1^{(l)} + w_2x_2^{(l)} + \dots + w_dx_d^{(l)} - \mathbf{y}^{(l)} \right)^2 \quad (2.2)$$

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

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

## COMP4211 - Machine Learning

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:

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

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:

$$\begin{aligned} \frac{\partial L}{\partial w_0} &= 2 \sum_{l=1}^N \left( w_0 + w_1 x^{(l)} - y^{(l)} \right) \cdot 1 = 0 \implies \sum_{l=1}^N \left( w_0 + w_1 x^{(l)} \right) = \sum_{l=1}^N y^{(l)} \implies Nw_0 + w_1 \sum_{l=1}^N x^{(l)} = \sum_{l=1}^N y^{(l)} \\ \frac{\partial L}{\partial w_1} &= 2 \sum_{l=1}^N \left( w_0 + w_1 x^{(l)} - y^{(l)} \right) x^{(l)} = 0 \implies w_0 \sum_{l=1}^N x^{(l)} + w_1 \sum_{l=1}^N x^{(l)^2} = \sum_{l=1}^N y^{(l)} x^{(l)} \end{aligned}$$

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

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

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

$$\mathbf{w} = \mathbf{A}^{-1}\mathbf{b} \quad (2.6)$$

### 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:

$$\mathbf{A}\mathbf{w} = (\mathbf{X}^T \mathbf{X})\mathbf{w} = \mathbf{X}^T \mathbf{y} = \mathbf{b} \quad (2.7)$$

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  $\mathbf{A}^{-1}$ :

$$\mathbf{w} = \mathbf{A}^{-1}\mathbf{b} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (2.8)$$

## COMP4211 - Machine Learning

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

$$\mathbf{X}\mathbf{w} - \mathbf{Y} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} - \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix} = \begin{bmatrix} w_0 + w_1x_1^{(1)} + w_2x_2^{(1)} + \dots + w_dx_d^{(1)} - y^{(1)} \\ w_0 + w_1x_1^{(2)} + w_2x_2^{(2)} + \dots + w_dx_d^{(2)} - y^{(2)} \\ \vdots \\ w_0 + w_1x_1^{(N)} + w_2x_2^{(N)} + \dots + w_dx_d^{(N)} - y^{(N)} \end{bmatrix}$$

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

$$L(\mathbf{w}; S) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 \quad (2.9)$$

Then we can express the squared loss function as:

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

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

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

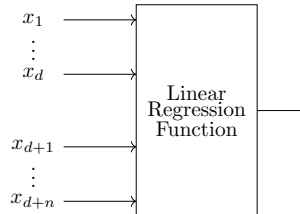
To find the optimal solution  $\mathbf{w}$ , we must compute the inverse of the matrix  $\mathbf{X}^T \mathbf{X} \in \mathbb{R}^{(d+1) \times (d+1)}$ . We can apply the **LeGall algorithm**, recognized as the quickest identified method for matrix multiplication, exhibiting a **time complexity of  $O(n^{2.3728596})$**  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

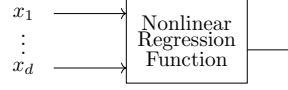
Nonlinear regression functions are crucial because more complex issues cannot be addressed by linear regression. To tackle this, we can modify the linear regression function  $f(\mathbf{x}; \mathbf{w})$  into a nonlinear format by adding nonlinear transformations of the input features. Three distinct 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  $x_1x_2$  to create an updated input  $\tilde{\mathbf{x}} = (1, x_1, x_2, x_1^2, x_2^2, x_1x_2)$ .

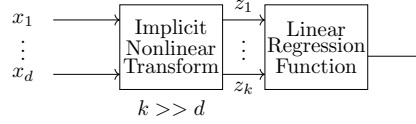
The key benefit is that linear regression remains applicable, and the weights in the model are **highly interpretable** (the greater the weight's magnitude, the more significant the feature). The primary drawback is that it necessitates domain expertise to develop new features and can be costly in terms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

$$f(\mathbf{x}; \mathbf{w}) = w_0 + w_1x + \dots + w_mx^m = \begin{bmatrix} w_0 & w_1 & \dots & w_m \end{bmatrix} \begin{bmatrix} 1 \\ x \\ \vdots \\ x^m \end{bmatrix} = \mathbf{w}^T \tilde{\mathbf{x}} = \tilde{\mathbf{x}}^T \mathbf{w} \quad (2.10)$$

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

Although  $f(\mathbf{x}; \mathbf{w})$  is nonlinear with respect to the input  $x$ , it stays linear concerning the parameters  $\mathbf{w}$ . Consequently, we can utilize the same least squares technique to determine the ideal parameter vector  $\mathbf{w}$  by minimizing the squared loss function.

Additionally, we can employ **broader transformations** of the initial input dimensions to develop polynomial features. For example, feature engineering can specify the features tailored to the application that are used on the input data, including interaction terms or transformations specific to the domain. Using body weight and body height as an illustration, the body mass index (BMI) will be the supplementary characteristic specified. The least squares method for linear regression remains applicable to determine the optimal parameter vector  $\mathbf{w}$  for acquiring the **closed form solution**.

## 2.6 Regularization

Regularization is a technique that alters the original loss function by incorporating one or more penalty terms known as **regularizers** that discourage large parameter values to avoid overfitting due to excessively high parameter magnitudes.



**Overfitting** arises when the **training dataset**  $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^N$  is **limited in size relative to the number of parameters in the linear regression model**  $f(\mathbf{x}; \mathbf{w})$ . In this scenario, the model could discover significant values in certain parameters, as an expanded search space enables the model to align more closely with the training data, which may result in inadequate generalization on new data.

Regularization helps tackle this issue by constraining the magnitude of the parameters, promoting simpler models that are less likely to overfit the training data. The **loss function with regularization** employing  $L_2$  **regularization / Tikhonov regularization** can be represented as:

$$L_\lambda(\mathbf{w}; S) = L(\mathbf{w}; S) + \lambda \|\mathbf{w}\|^2 = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda \|\mathbf{w}\|^2 = \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{y}^T \mathbf{X} \mathbf{w} + \mathbf{y}^T \mathbf{y} + \lambda \mathbf{w}^T \mathbf{w} \quad (2.11)$$

- The parameter  $\lambda > 0$  serves as a **regularization hyperparameter** that regulates the intensity of the regularization, established during the validation phase.  
A greater  $\lambda$  produces more intense regularization, whereas a lesser  $\lambda$  causes weaker regularization.
- The notation  $\|\mathbf{w}\|^2$  signifies the squared L2 norm of the parameter vector  $\mathbf{w}$ , imposing a penalty on large parameter values.

In practice, the bias  $w_0$  is frequently left out of the regularization term since it merely acts as an **offset** and does not add to the model's complexity. The loss function with regularization can be optimized through techniques like gradient descent or by utilizing **closed-form solutions**.

To find the closed-form solution with  $L_2$  regularization, we can differentiate the loss function  $L_\lambda(\mathbf{w}; S)$  with respect to  $\mathbf{w}$  and set it to zero:

$$2\mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{X}^T \mathbf{y} + 2\lambda \mathbf{w} = 0 \implies (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{w} = \mathbf{X}^T \mathbf{y} \quad \text{where } \mathbf{I} \text{ is the identity matrix}$$

The closed-form solution for the least squares estimate of the parameter vector  $\mathbf{w}$  with  $L_2$  regularization can be defined as:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y} \quad \text{where } \mathbf{X}^T \mathbf{X} + \lambda \mathbf{I} \text{ is invertible for } \lambda > 0 \quad (2.12)$$

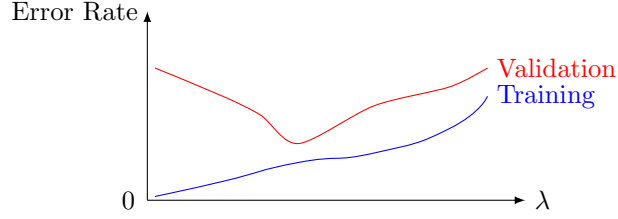
This regularized method is referred to as **Ridge Regression**. It minimizes overfitting and enhances the model's ability to generalize to new data by imposing penalties on large parameter values. Crucially, when the regularization parameter  $\lambda$  is equal to zero, ridge Regression reduces to standard linear regression, since no penalty is enforced.

## 2.6.1 Choice of Regularization Parameter $\lambda$

The regularization coefficient  $\lambda$  is an essential hyperparameter that dictates the intensity of the regularization imposed on the model. The selection of  $\lambda$  can greatly influence the performance of the model and is usually established via a validation process. A frequent method for determining the best value of  $\lambda$  is to employ **cross-validation**.

Cross-validation involves **splitting the training dataset into multiple subsets (folds) and training the model on one subset** while evaluating it on the remaining data. Subsequently, the trained model is **evaluated on a separate validation set** to resemble the test set. The procedure is carried out for different  $\lambda$  values, and the model's effectiveness is assessed utilizing a specific metric (e.g., mean squared error, accuracy). The best performance on the validation set is achieved by choosing the optimal regularization parameter as the value of  $\lambda$  with the smallest validation error.

The standard training and validation error curves for various  $\lambda$  values are illustrated below.



The training error usually rises with an increase in  $\lambda$ , whereas the validation error first drops and then rises, suggesting that **a low  $\lambda$  causes overfitting whereas a high  $\lambda$  leads to underfitting**.

## 2.6.2 Other Regularization Techniques

In addition to  $L_2$  regularization, other approaches can also be used to prevent overfitting in machine learning models. A general framework for these methods is provided by the  $L_p$  norm, which generalizes both the  $L_1$  and  $L_2$  norms. By varying the value of  $p$ , a family of regularization strategies can be defined:

$$\|\mathbf{v}\|_p = \left( \sum_{i=1}^n |v_i|^p \right)^{1/p} \quad \text{for } p \geq 1 \quad (2.13)$$

Among these,  $L_1$  regularization, known as **LASSO (Least Absolute Shrinkage and Selection Operator)**, is especially popular. LASSO adds a penalty term based on the  $L_1$  norm of the parameter vector  $\mathbf{w}$  to the loss function, which encourages sparsity by driving many coefficients to exactly zero:

$$L_\lambda(\mathbf{w}; S) = L(\mathbf{w}; S) + \lambda \|\mathbf{w}\|_1 = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda \sum_{i=0}^d |w_i| \quad (2.14)$$

While  $L_1$  regularization leads to a convex optimization problem, it does not admit a closed-form solution. Instead, iterative optimization algorithms such as coordinate descent or subgradient descent are typically used to find the optimal parameter vector  $\mathbf{w}$ .

## 2.7 Common Performance Metrics for Regression

To evaluate the performance of regression models, several metrics are commonly used. These metrics help assess how well the model predicts the target variable and can guide improvements in model design. Two of the most widely used metrics are **Mean Squared Error (MSE)** and **R-squared Scores**.

### 2.7.1 Mean Squared Error (MSE)

Once a regression model has been trained, it is crucial to quantitatively evaluate how accurately the model's predictions align with the real target values. A commonly utilized metric for this aim is the **Mean Squared Error (MSE)**. MSE computes the **mean of the squared deviations between the predicted values and the actual target values for all samples**. It is described as:

$$\text{MSE} = \frac{1}{N} \sum_{l=1}^N \left( f(\mathbf{x}^{(l)}; \mathbf{w}) - \mathbf{y}^{(l)} \right)^2 = \frac{1}{N} L(\mathbf{w}; S) \quad (2.15)$$

- $N$  is the total number of samples in the dataset,  $f(\mathbf{x}^{(l)}; \mathbf{w})$  is the predicted value for the  $l$ -th sample, and  $\mathbf{y}^{(l)}$  is the corresponding true value.

MSE can be calculated on the training set, validation set, or test set to assess model performance and generalization.

Nonetheless, since MSE is represented in the squared units of the target variable, interpreting it directly can be challenging. To tackle this, the **Root Mean Squared Error (RMSE)** is commonly employed, as it calculates the square root of the MSE to revert the error to the initial scale of the target variable:

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{N} \sum_{l=1}^N (f(\mathbf{x}^{(l)}; \mathbf{w}) - \mathbf{y}^{(l)})^2} \quad (2.16)$$

Consequently, RMSE is easier to understand and gives a clear indication of the usual size of prediction errors in the same units as the target variable.

## 2.7.2 R-squared ( $R^2$ ) Score

Though MSE and RMSE offer an absolute assessment of prediction error, having a relative metric that shows how effectively the model accounts for the variability in the target variable against a basic baseline is often beneficial. The **R-squared ( $R^2$ ) value**, often referred to as the **coefficient of determination**, fulfills this role.

$R^2$  measures the fraction of the overall variability in the target variable that the model's predictions explain. It is characterized as:

$$R^2 = 1 - \frac{\sum_{l=1}^N (f(\mathbf{x}^{(l)}; \mathbf{w}) - \mathbf{y}^{(l)})^2}{\sum_{l=1}^N (\mathbf{y}^{(l)} - \bar{\mathbf{y}})^2} = 1 - \frac{\text{MSE}}{\text{Var}(\mathbf{y})} \quad \text{where } \bar{\mathbf{y}} = \frac{1}{N} \sum_{l=1}^N \mathbf{y}^{(l)} \text{ is the mean of the target variable} \quad (2.17)$$

- The numerator is the residual sum of squares (RSS), indicating the variance that remains unexplained (the discrepancy between predicted and actual values).
- The denominator is the total sum of squares (TSS), which indicates the overall variance in the target variable concerning its average.

$R^2$  varies between 0 and 1, with **1 indicating that the model completely accounts for all variability** in the target variable, and **0 signifying no explanation** at all (similar to predicting the average consistently). Negative  $R^2$  values may arise when the model is less effective than merely predicting the average, signifying inadequate model fit