Supervised Learning: Regression Analysis

Machine Learning Tutorial

May 24, 2025

Contents

1 Introduction to Supervised Learning

Supervised learning is a fundamental paradigm in machine learning where algorithms learn from labeled training data to make predictions on new, unseen data. The goal is to find a mapping function $f : X \to Y$ that maps input features X to target outputs Y.

Definition 1 (Supervised Learning)**.** Given a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$ represents the input features and y_i represents the corresponding target values, supervised learning aims to learn a function $h : \mathbb{R}^d \to \mathbb{R}$ that minimizes the prediction error on unseen data.

Supervised learning problems are typically categorized into two main types:

- **Classification**: When the target variable y is categorical (discrete classes)
- **Regression**: When the target variable y is continuous (real-valued)

This document focuses specifically on regression problems.

2 Regression Analysis

2.1 Definition and Objective

Definition 2 (Regression Problem)**.** In a regression problem, we aim to predict a continuous target variable $y \in \mathbb{R}$ given input features $x \in \mathbb{R}^d$. The goal is to learn a function $f(x)$ that approximates the true relationship between inputs and outputs.

The general form of a regression model can be expressed as:

$$y = f(x) + \epsilon \tag{1}$$

where $f(x)$ is the true underlying function and ϵ is the noise term.

2.2 Loss Functions

To evaluate the quality of our predictions, we define loss functions that measure the difference between predicted and actual values.

2.2.1 Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2}$$

2.2.2 Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{3}$$

2.2.3 Root Mean Squared Error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4}$$

3 Linear Regression

3.1 Simple Linear Regression

Simple linear regression models the relationship between a single independent variable x and a dependent variable y using a linear equation:

$$y = \beta_0 + \beta_1 x + \epsilon \tag{5}$$

where:

- β_0 is the y-intercept
- β_1 is the slope coefficient
- ϵ is the error term

Example 1 (House Price Prediction)**.** Consider predicting house prices based on square footage:

$$\text{Price} = 50000 + 100 \times \text{Square Feet} + \epsilon \tag{6}$$

This suggests that each additional square foot increases the price by \$100, with a base price of \$50,000.

3.2 Multiple Linear Regression

Multiple linear regression extends simple linear regression to multiple independent variables:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon \tag{7}$$

In matrix form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{8}$$

where:

- $\mathbf{y} \in \mathbb{R}^n$ is the response vector
- $\mathbf{X} \in \mathbb{R}^{n \times (p+1)}$ is the design matrix
- $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ is the parameter vector
- $\boldsymbol{\epsilon} \in \mathbb{R}^n$ is the error vector

3.3 Parameter Estimation

3.3.1 Ordinary Least Squares (OLS)

The OLS method finds parameters that minimize the sum of squared residuals:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \tag{9}$$

The closed-form solution is:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{10}$$

Algorithm 1 Ordinary Least Squares

procedure OLS($\mathbf{X}, \mathbf{y}$)
    Compute $\mathbf{X}^T\mathbf{X}$
    Compute $\mathbf{X}^T\mathbf{y}$
    Solve $(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y}$
    return $\boldsymbol{\beta}$
end procedure

3.3.2 Gradient Descent

For large datasets, gradient descent provides an iterative solution:

Algorithm 2 Gradient Descent for Linear Regression

procedure GradientDescent($\mathbf{X}, \mathbf{y}, \alpha, \text{max\_iter}$)
    Initialize $\boldsymbol{\beta} \leftarrow \mathbf{0}$
    for $t = 1$ to max_iter **do**
        $\mathbf{h} \leftarrow \mathbf{X}\boldsymbol{\beta}$ ▷ Predictions
        $\mathbf{e} \leftarrow \mathbf{h} - \mathbf{y}$ ▷ Errors
        $\boldsymbol{\nabla} \leftarrow \frac{2}{n}\mathbf{X}^T\mathbf{e}$ ▷ Gradient
        $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \alpha\boldsymbol{\nabla}$ ▷ Update
    end for
    return $\boldsymbol{\beta}$
end procedure

4 Regularized Regression

Regularization techniques prevent overfitting by adding penalty terms to the loss function.

4.1 Ridge Regression (L2 Regularization)

Ridge regression adds an L2 penalty term:

$$\mathcal{L}_{\text{Ridge}} = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|^2 \tag{11}$$

The closed-form solution is:

$$\hat{\boldsymbol{\beta}}_{\text{Ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{12}$$

where $\lambda > 0$ is the regularization parameter.

4.2 Lasso Regression (L1 Regularization)

Lasso regression uses an L1 penalty:

$$\mathcal{L}_{\text{Lasso}} = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{13}$$

Lasso performs automatic feature selection by setting some coefficients to zero.

4.3 Elastic Net

Elastic Net combines both L1 and L2 penalties:

$$\mathcal{L}_{\text{ElasticNet}} = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1\|\boldsymbol{\beta}\|_1 + \lambda_2\|\boldsymbol{\beta}\|^2 \tag{14}$$

Algorithm 3 Coordinate Descent for Lasso

procedure LassoCoordinateDescent($\mathbf{X}, \mathbf{y}, \lambda, \text{max\_iter}$)
  Initialize $\boldsymbol{\beta} \leftarrow \mathbf{0}$
  for $t = 1$ to max_iter **do**
    for $j = 1$ to p **do**
      $r_j \leftarrow \sum_{i=1}^{n} x_{ij}(y_i - \sum_{k \neq j} x_{ik}\beta_k)$
      $\beta_j \leftarrow \text{SoftThreshold}(r_j, \lambda) / \sum_{i=1}^{n} x_{ij}^2$
    end for
  end for
  return $\boldsymbol{\beta}$
end procedure

5 Polynomial Regression

Polynomial regression extends linear regression by including polynomial terms:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \epsilon \tag{15}$$

This can still be solved using linear regression techniques by treating $x, x^2, \ldots, x^d$ as separate features.

Example 2 (Polynomial Regression)**.** Consider fitting a quadratic model to data:

$$y = 2 + 3x - 0.5x^2 + \epsilon \tag{16}$$

The design matrix becomes:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \tag{17}$$

6 Non-linear Regression Methods

6.1 Decision Tree Regression

Decision trees partition the feature space into regions and predict the average target value in each region.

Algorithm 4 Decision Tree Regression (Simplified)

procedure BUILDTREE($\mathcal{D}$, depth)
  if stopping criterion met **then**
    return leaf with mean($\{y_i : (x_i, y_i) \in \mathcal{D}\}$)
  end if
  Find best split (j, s) that minimizes MSE
  $\mathcal{D}_L \leftarrow \{(x_i, y_i) \in \mathcal{D} : x_{i,j} \leq s\}$
  $\mathcal{D}_R \leftarrow \{(x_i, y_i) \in \mathcal{D} : x_{i,j} > s\}$
  left $\leftarrow$ BuildTree($\mathcal{D}_L$, depth + 1)
  right $\leftarrow$ BuildTree($\mathcal{D}_R$, depth + 1)
  return internal node with split (j, s), left, right
end procedure

6.2 Random Forest Regression

Random Forest combines multiple decision trees:

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} T_b(x) \tag{18}$$

where T_b is the b-th tree trained on a bootstrap sample.

6.3 Support Vector Regression (SVR)

SVR finds a function that deviates from targets by at most ϵ:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{19}$$

subject to:

$$y_i - \mathbf{w}^T\phi(x_i) - b \leq \epsilon + \xi_i \tag{20}$$

$$\mathbf{w}^T\phi(x_i) + b - y_i \leq \epsilon + \xi_i \tag{21}$$

$$\xi_i \geq 0 \tag{22}$$

7 Model Evaluation and Validation

7.1 Cross-Validation

K-fold cross-validation partitions data into k folds and uses each fold as validation set:

Algorithm 5 K-Fold Cross-Validation

```
procedure KFoldCV(𝒟, k, model)
    Partition 𝒟 into k folds: 𝒟_1, ..., 𝒟_k
    scores ← []
    for i = 1 to k do
        𝒟_train ← 𝒟 \ 𝒟_i
        𝒟_val ← 𝒟_i
        Train model on 𝒟_train
        Evaluate model on 𝒟_val
        Append score to scores
    end for
    return mean(scores), std(scores)
end procedure
```

7.2 Model Selection Metrics

Beyond MSE, several metrics help evaluate regression models:

7.2.1 R-squared (Coefficient of Determination)

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{23}$$

7.2.2 Adjusted R-squared

$$R^2_{\text{adj}} = 1 - \frac{(1 - R^2)(n-1)}{n - p - 1} \tag{24}$$

7.2.3 Akaike Information Criterion (AIC)

$$\text{AIC} = 2p - 2\ln(\mathcal{L}) \tag{25}$$

8 Practical Examples

Example 3 (Boston Housing Dataset)**.** Consider predicting median home values using features like crime rate, average rooms, etc.

Features: x_1 (crime rate), x_2 (average rooms), x_3 (distance to employment centers)
Model: Price $= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$
Using OLS estimation:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{26}$$

Example 4 (Polynomial Feature Engineering)**.** For a dataset with non-linear relationships, we might transform:

$$x \to [1, x, x^2, x^3, \sin(x), \cos(x)] \tag{27}$$

$$\text{Model:} \quad y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 \sin(x) + \beta_5 \cos(x) \tag{28}$$

9 Implementation Considerations

9.1 Feature Scaling

Standardization ensures features have similar scales:

$$x_{\text{scaled}} = \frac{x - \mu}{\sigma} \tag{29}$$

9.2 Handling Multicollinearity

When features are highly correlated, consider:

- Ridge regression to handle correlated features
- Principal Component Analysis (PCA) for dimensionality reduction
- Variance Inflation Factor (VIF) analysis

9.3 Outlier Detection

Common methods include:

- Cook's distance for influential points
- Residual analysis
- Robust regression techniques

10 Conclusion

Supervised learning regression encompasses a wide range of techniques for predicting continuous variables. The choice of method depends on factors such as:

- Dataset size and dimensionality
- Linearity of relationships
- Interpretability requirements
- Computational constraints
- Presence of noise and outliers

Linear regression remains fundamental due to its simplicity and interpretability, while regularized methods like Ridge and Lasso help prevent overfitting. Non-linear methods such as decision trees and SVR can capture complex patterns but may sacrifice interpretability.

Proper model evaluation through cross-validation and appropriate metrics ensures robust performance assessment and helps guide model selection for specific applications.