MATH5836: Data and Machine Learning

Week 1: Data and Linear Regression

Sarat Moka *UNSW, Sydney*

Key Topics

Books:

(A) *Pattern Recognition and Machine Learning* by Christopher Bishop (2006).

(B) *Mathematical Engineering of Deep Learning* book by Liquet, Moka, and Nazarathy (2024).

1.1 Data Mining vs Machine Learning

What is Data Mining?

Data mining extracts actionable knowledge from large datasets by applying statistical, machine learning, and database techniques. It identifies hidden patterns, relationships, and trends to support decision-making, prediction, and process optimization.

1. **Data Preparation**: Clean and preprocess raw data through missing value handling, outlier removal, and transformation for analysis-ready quality.
2. **Exploratory Analysis (EDA)**: Investigate data structure and relationships using statistical summaries, visualization, and correlation analysis.
3. **Feature Engineering**: Select relevant predictors and create new features to enhance model performance.
4. **Model Development**: Implement and validate techniques like classification, regression, clustering, association rules, and anomaly detection using appropriate evaluation metrics.
5. **Insight Communication**: Interpret results through visualizations (charts, graphs, dashboards) and actionable recommendations.

Data mining drives value across industries: financial fraud detection, customer segmentation in marketing, treatment optimization in healthcare, and operational efficiency in supply chains. With advancing big data technologies, it remains essential for transforming complex datasets into strategic assets.

What is Machine Learning?

Machine learning creates adaptive systems that *automate decision-making through statistical pattern recognition* from data, rather than relying on explicit programming. ML models iteratively improve their performance by extracting meaningful relationships from structured or unstructured inputs.

1. **Data-Driven Learning**: Models infer patterns from training datasets, where performance scales with data quality and representativeness. Critical considerations include feature relevance and sample diversity.
2. **Generalization Capacity**: Effective models maintain predictive accuracy on unseen data through techniques like regularization and cross-validation, preventing overfitting to training artifacts.
3. **Learning Paradigms**:
    - *Supervised*: Learns input-output relationships using labeled data (e.g., classification, regression)

- *Unsupervised*: Discovers latent structures in unlabeled data (e.g., clustering, dimensionality reduction)
- *Reinforcement*: Maximizes cumulative rewards through environment interactions (e.g., Markov decision processes)

4. **Algorithmic Approaches**: Implements mathematical frameworks including:
   - Parametric models (linear regression, neural networks)
   - Non-parametric methods (decision trees, k-nearest neighbors)
   - Ensemble techniques (gradient boosting, random forests)
5. **Strategic Applications**:
   - Core ML domains: Natural language processing, computer vision, speech recognition
   - Industry impact: Medical diagnostics, algorithmic trading, autonomous navigation, recommender systems
   - Emerging frontiers: Generative AI, reinforcement learning agents, federated learning

Machine learning provides systematic methodologies for building predictive and prescriptive analytics tools that adapt to evolving data environments. Its mathematical foundations enable solutions where traditional algorithmic approaches become intractable.

Data Mining vs. Machine Learning: Relationship & Synergies

Core Distinctions:

- **Objective**:
  - *DM*: Pattern discovery through EDA, clustering (association rules, anomaly detection)
  - *ML*: Predictive model development with generalization guarantees
- **Focus**:
  - *DM*: Insight generation for human decision-making (descriptive analytics)
  - *ML*: Automated decision systems (prescriptive/predictive analytics)
- **Data Utilization**:
  - *DM*: Emphasis on structured transactional data + feature engineering
  - *ML*: Agnostic to data morphology (raw signals → structured inputs), i.e., ML architectures increasingly process raw, unstructured data directly (text, pixels, sensor streams) through automated feature learning, rather than requiring manual feature engineering.

Key Synergies:

1. **Algorithmic Cross-Pollination**:
   - DM techniques (Apriori, DBSCAN) formalized as ML implementations
   - ML methods (SVMs, deep neural nets) enhance traditional mining tasks
2. **Methodological Overlap**:
   - Shared statistical foundations: Bayesian inference, entropy measures
   - Common optimization frameworks: Gradient descent, expectation-maximization
3. **Modern Convergence**:
   - Scalable implementations via Spark MLlib, TensorFlow Extended (TFX)
   - Integrated pipelines: Feature engineering (DM) $\rightarrow$ Model training (ML) $\rightarrow$ Deployment
   - Emerging fusion: Deep learning for pattern mining (e.g., graph neural networks in recommendation systems)
4. **Evolutionary Perspective**: The DM-ML division is increasingly obsolete as:
   - Modern ML subsumes many mining tasks through automated feature learning
   - DM provides critical preprocessing/validation frameworks for ML pipelines
   - Both fields converge in causal inference and explainable AI research

The distinction now primarily exists in *intent* rather than *methodology* - DM as application-focused knowledge discovery vs. ML as general-purpose learning framework.

1.2 Supervised vs. Unsupervised Learning

Definitions:

- **Supervised Learning**: Uses labeled data (X, Y) to learn a function $f : X \rightarrow Y$ mapping inputs to known outputs
- **Unsupervised Learning**: Finds patterns in unlabeled data X by modeling its probability distribution $P(X)$ or discovering latent structures

Key Distinctions:

Aspect	**Supervised**	**Unsupervised**
Data	Labeled (X, Y) pairs	Raw observations X
Goal	Predict Y from X	Discover structure in X
Evaluation	Accuracy, F1-score, MSE	Silhouette score, reconstruction error
Methods	Regression, classification	Clustering, PCA

Examples:

- **Supervised**:
  - Email spam detection (X=email text, Y=spam/not spam)
  - House price prediction (X=features, Y=price)
- **Unsupervised**:
  - Customer segmentation (X=purchase history)
  - Image compression via K-means (reduce color palette)

Core Idea: Supervised requires explicit labels ("teacher"), while unsupervised discovers inherent patterns autonomously.

1.3 Types of Supervised Learning

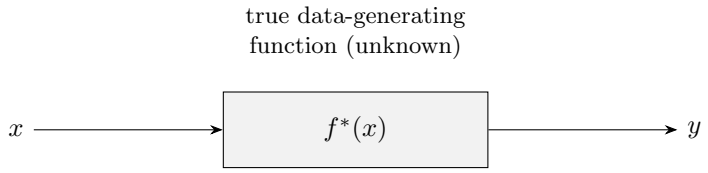


Figure 1.1: Real-world data generation: the observed output y arises via the true mapping $f^*(x)$—which is latent in practice—and possibly additional noise.

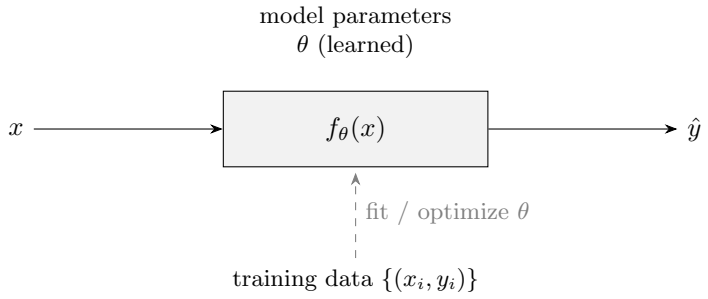


Figure 1.2: Parametric model in supervised learning. We use a labeled dataset $\{(x_i, y_i)\}$ to optimize the unknown parameter vector θ, yielding a predictor $\hat{y} = f_\theta(x)$.

- Figure 1.1 illustrates how real-world outputs y arise from an unknown "black-box" mapping f^*.
- An input x passes through the true data-generating mechanism $y = f^*(x)$.
- In practice, f^* (and any system or measurement noise) is latent and cannot be inspected.

- This motivates learning an approximate model $\hat{f}$ to emulate f^* for prediction.
- Figure 1.2 shows a parametric hypothesis $f_\theta(x)$ with unknown parameters θ.
- A labeled training set $\{(x_i, y_i)\}$ (dashed arrow) is used to fit θ.
- Once optimized, the model computes predictions $\hat{y} = f_\theta(x)$ for new inputs.
- This highlights the two stages of supervised learning:
  - *Learning*: estimating θ from data
  - *Inference*: applying f_θ to produce $\hat{y}$

Regression vs. Classification

Supervised learning tasks can be grouped into two main categories:

- **Regression**
  - *Goal*: Predict a **continuous** response $y \in \mathbb{R}$ from inputs $x \in \mathbb{R}^d$.
  - *Common Models*: Linear regression, ridge/lasso, support vector regression, neural nets.
  - *Examples*:
    - House-price estimation
    - Temperature forecasting
    - Demand prediction
- **Classification**
  - *Goal*: Assign a **discrete** label $y \in \{1, \ldots, C\}$ to each input x.
  - *Common Models*: Logistic regression, softmax regression, SVM, decision trees, random forests, deep neural networks.
  - *Examples*:
    - Email spam detection
    - Image digit recognition
    - Medical diagnosis (benign vs. malignant)

1.4 Error Measurements in Supervised Learning

Measurement of errors play a crucial role in both regression and classification tasks. Below are the definitions of various types of measurement errors and commonly used loss functions for evaluating the performance of models. These loss functions provide different ways to measure the discrepancy between the predicted and actual outcomes.

Regression

In regression tasks, the goal is to predict continuous outcomes. As usual, let y_i's are the true response variables and $\hat{y}_i$'s are the corresponding predictions. Then the following are some commonly used loss functions.

- **Mean Squared Error (MSE):** Measures the average squared difference between the predicted and actual values:
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2.$$
- **Root Mean Squared Error (RMSE):** It is the square roor of MSE, that is,
$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}.$$
- **Mean Absolute Error (MAE):** Measures the average absolute difference between the predicted and actual values:
$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|.$$
- **Huber Loss:** A combination of MSE and MAE that is less sensitive to outliers:
$$\text{Huber Loss} = \frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \hat{y}_i),$$
where
$$\ell(y_i, \hat{y}_i) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| \le \delta \\ \delta(|y_i - \hat{y}_i| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}.$$
- **Log-Cosh Loss:** Approximates the logarithm of the hyperbolic cosine of the prediction error.
$$\text{Log-Cosh Loss} = \frac{1}{n}\sum_{i=1}^{n}\log(\cosh(y_i - \hat{y}_i))$$
- **Quantile Loss:** Generalization of MAE that allows for different penalties for overestimation and underestimation.
$$\text{Quantile Loss}_\tau = \frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \hat{y}_i),$$
where
$$\ell(y_i, \hat{y}_i) = \begin{cases} \tau(y_i - \hat{y}_i) & \text{if } y_i - \hat{y}_i \ge 0 \\ (\tau - 1)(y_i - \hat{y}_i) & \text{otherwise} \end{cases}.$$

Remark 1.4.1

To see the relationship between MSE, MAE and Huber loss, refer to Figure 1.3 below.

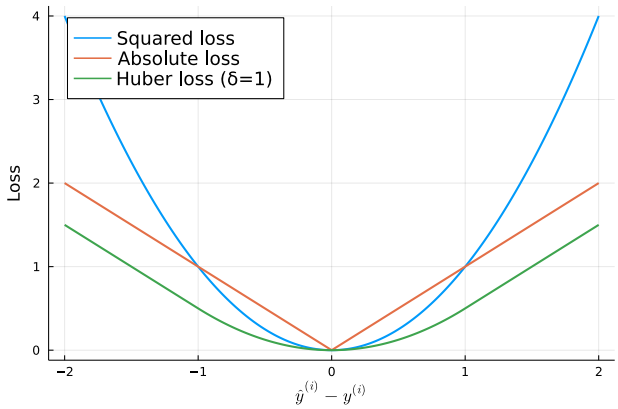


Figure 1.3: Loss function and error distribution alternatives. (a) Squared, absolute, and Huber loss functions.

Classification

In classification tasks, the goal is to predict categorical outcomes. In classification, for each sample i, we have a true label y_i and a vector of prediction probabilities $\hat{\boldsymbol{p}}_i = (\hat{p}_{i,1}, \ldots, \hat{p}_{i,K})$ with K denoting that number of classes and $\hat{p}_{i,k} = \mathbb{P}(y_i = k)$. For binary classification, we take $\hat{\boldsymbol{p}}_i = (\hat{p}_i, 1 - \hat{p}_i)$ with $\hat{p}_i = \mathbb{P}(y_i = 1)$. The following are some commonly used loss functions.

- **Log Loss (Cross-Entropy Loss):** Applies to binary classification where the response variables y_i takes values in $\{0, 1\}$. Measures the difference between the predicted and actual class probabilities.

$$\text{Log Loss} = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)\right),$$

  with $\hat{p}_i$ denoting the predictions (probabilities).

- **Focal Loss:** A modification of cross-entropy loss that focuses on hard-to-classify examples.

$$\text{Focal Loss}_\gamma = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i(1 - \hat{p}_i)^\gamma \log(\hat{p}_i) + (1 - y_i)\hat{p}_i^\gamma \log(1 - \hat{p}_i)\right).$$

- **Exponential Loss:** Penalizes misclassifications exponentially.

$$\text{Exponential Loss} = \frac{1}{n}\sum_{i=1}^{n} e^{-y_i \hat{p}_i}$$

- **Hinge Loss:** Used in binary classification tasks with support vector machines (SVMs).

$$\text{Hinge Loss} = \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i \hat{p}_i)$$

- **KL Divergence (Relative Entropy):** Can be applied for multiclass classification. Measures the difference between the predicted and actual probability distributions.

$$\text{KL Divergence} = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} I(y_i = k)\log\left(\frac{I(y_i = k)}{\hat{p}_{i,k}}\right),$$

  with the convention that $0 \log 0 = 0$, where $\hat{p}_{i,k}$ is the predicted probability of i-th sample is of class k.

1.5 Gradient descent

Gradient descent is a popular optimization algorithm used to minimize a function by iteratively moving in the direction of steepest descent. It is widely employed in machine learning for training models and in various optimization problems. In this section, we present the basic formulation of gradient descent.

Formulation

Suppose we have an error measurement function $C : \mathbb{R}^p \to \mathbb{R}$ that we want to minimize, where $\theta = (\theta_1, \ldots, \theta_p)$ is a vector of parameters. That is, our goal is solve the optimization,

$$\min_{\theta} C(\theta).$$

Remark 1.5.1

For example, $C(\theta)$ can be MSE in regression or KL divergence for classification. It is a function of the learnable parameters θ as the prediction $\hat{y}$ depends on θ.

The gradient descent algorithm starts with an initial guess $\theta^{(0)}$ and updates the parameters iteratively according to the rule:

$$\theta^{(n+1)} = \theta^{(n)} - \alpha \nabla C(\theta^{(n)})$$

where α is the learning rate and $\nabla C(\theta^{(n)})$ is the gradient of the function C evaluated at $\theta^{(n)}$. The learning rate determines the step size of each iteration. Here, the gradient $\nabla C(\theta)$ at θ is defined as a vector of partial derivatives given by

$$\nabla C(\theta) = \left[\frac{\partial f(\theta)}{\partial \theta_1}, \cdots, \frac{\partial C(\theta)}{\partial \theta_p}\right]^\top.$$

Intuitively, $\frac{\partial C(\theta)}{\partial \theta_i}$ denotes the rate of change in the function value in the direction of θ_i at θ.

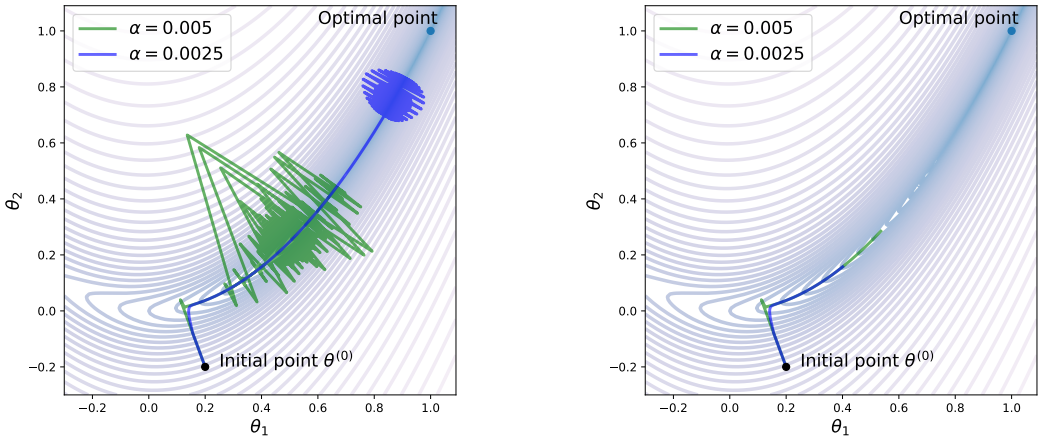
Algorithm

The gradient descent algorithm can be summarized as follows:

1. Initialize the parameters $\theta^{(0)}$.
2. Repeat until convergence or a maximum number of iterations:
   (a) Compute the gradient $\nabla C(\theta^{(n)})$.
   (b) Update the parameters: $\theta^{(n+1)} = \theta^{(n)} - \alpha \nabla C(\theta^{(n)})$.
3. Return the final θ.

Convergence

The convergence of gradient descent to the optimal minimum point θ^* depends on various factors, including the choice of the learning rate α, the properties of the objective function $C(\theta)$, and the initial point $\theta^{(0)}$. In general, if the learning rate is too large, the algorithm may overshoot the minimum or oscillate around it. On the other hand, if the learning rate is too small, the convergence may be slow.



(a) (b)

Figure 1.4: Attempting minimization of a Rosenbrock function via basic gradient descent for two values of α with 10^5 iterations and $\theta^{(0)} = (0.2, -0.2)$. (a) Fixed learning rate: $\alpha^{(t)} = \alpha$. (b) Exponentially decaying learning rate: $\alpha^{(t)} = \alpha\, 0.99^t$.

Example 1.5.1

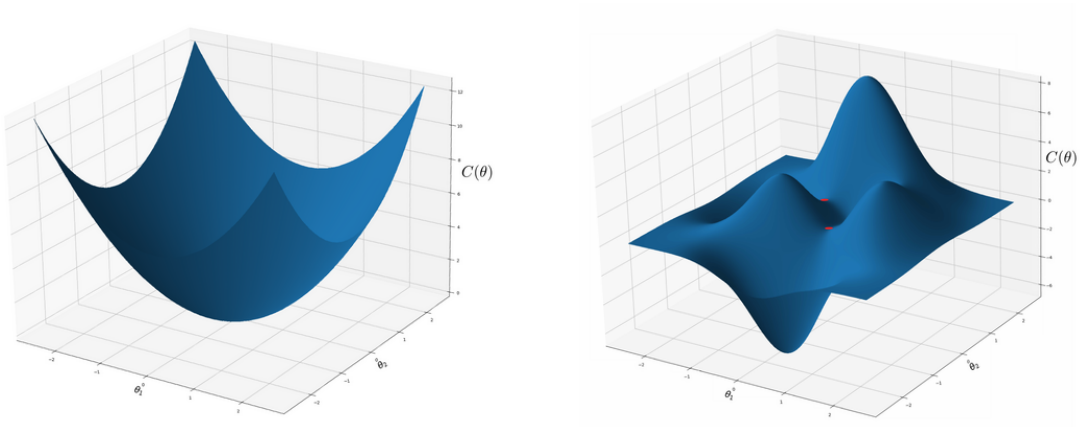
Consider the following 2-dimensional Rosenbrock function:

$$C(\theta) = (1-\theta_1)^2 + 100(\theta_2 - \theta_1^2)^2, \quad \theta = (\theta_1, \theta_2) \in \mathbb{R}^2. \tag{1.1}$$

This function has unique minimum at $\theta^* = (1, 1)$. We applied gradient descent on this function. Figure 1.4 illustrates how the learning rate effects the convergence.

Example 1.5.2

Figure 1.5 shows examples of convex and non-convex functions. For convex functions, we can often guarantee converges of gradient descent to a (global) minimum point for an appropriate choice of learning rate, starting at any initial point. Such a convergence cannot be guaranteed for non-convex function. Note that in deep learning, we mostly deal with minimization of non-convex functions.



(a) (b)

Figure 1.5: Two dimensional smooth functions. (a) The function $C(\theta) = \theta_1^2 + \theta_2^2$ is convex with a unique global minimum. (b) A well-known function called the *peaks function* is an example non-convex function with several local minima, local maxima, and saddle points. Around each local minimum, the function is locally convex. On the figure, two *saddle points* are marked with red dots.

Variants

Several variants of gradient descent exist, including stochastic gradient descent (SGD), mini-batch gradient descent, and momentum-based methods such as Adam optimizer and Nesterov momentum methods. These variants introduce modifications to the basic gradient descent algorithm to improve convergence speed or handle large datasets efficiently. We will cover some of these concepts later in this course[1].

[1]For complete details, read Chapter 4 of the book *Mathematical Engineering of Deep Learning*.

1.6 Linear regression

Linear regression is a fundamental statistical method used to model the relationship between a dependent variable and one or more independent variables. It is widely used in various fields for prediction, inference, and understanding the relationship between variables. In this section, we present the basic formulation of linear regression.

Formulation for Simple Case

We first focus on a simple case where independent variable has only one feature. Consider a dataset $\{(x_i, y_i)\}_{i=1}^n$, where x_i represents the independent variable and y_i represents the dependent variable. Linear regression seeks to find a linear relationship between x and y of the form:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i,$$

where β_0 and β_1 are the intercept and slope parameters, respectively, and ϵ_i is the error term representing the deviation of the observed data points from the regression line.

Least Squares Estimation: The most common method for estimating the parameters β_0 and β_1 in linear regression is the least squares method. It seeks to minimize the sum of squared residuals (differences between the observed and predicted values). In particular, with $\beta = (\beta_0, \beta_1)$, let

$$C(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2,$$

which is the mean squared error. Then, in linear regression, we aim to solve,

$$\min_{\beta} C(\beta).$$

A solution to the above problem, denoted as $\hat{\beta}_0$ and $\hat{\beta}_1$, is the least squares estimates of β_0 and β_1. In this simple case, we can obtain the least squares estimates by solving the normal equations:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of the independent and dependent variables, respectively.

Example 1.6.1

We consider Boston housing dataset[a], has 506 observation where each observation is associated with a suburb or town in the Boston Massachusetts area. Figure 1.6 shows modelling of median house prices as a function of average number of rooms per dwelling. Refer to Section 2.1 of Chapter 2 in the book *Mathematical Engineering of Deep Learning.*

[a] This dataset is originally published by D. Harrison Jr and D. L. Rubinfeld. *Hedonic Housing Prices and the Demand for Clean Air.* Journal of Environmental Economics and Management, 1978.

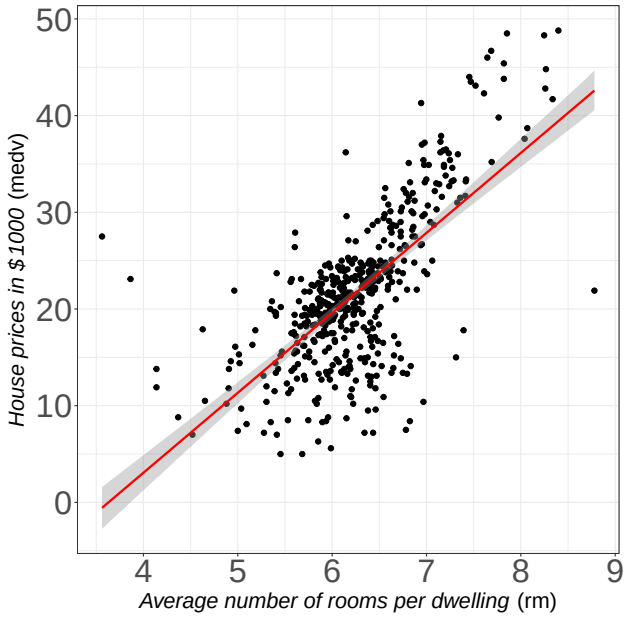


Figure 1.6: Example of simple linear model. Median house prices per locality (`medv`) as a function of average number of rooms per dwelling (`rm`) is described via a simple linear (affine) relationship.

Formulation for General Case

We now focus on case case where independent variable is a vector with p features. Consider a dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x}_i = (x_{i,1}, \ldots, x_{i,p}) \in \mathbb{R}^p$ represents the independent variable and y_i represents the corresponding dependent variable. Linear regression seeks to find a linear relationship between independent variable and dependent variable of the form:

$$y_i = \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p} + \epsilon_i, \quad i = 1, 2, \ldots, n, \tag{1.2}$$

where $\theta = \beta = (\beta_0, \beta_1, \ldots, \beta_p)$ is the vector of unknown parameters, with β_0 denoting the intercept, and ϵ_i is the error term representing the deviation of the observed data points from the regression line.

It is often easier to represent the above linear relationship using matrices and vectors. In particular, let $\boldsymbol{y} = (y_1, \ldots, y_n)$, $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)$, and

$$X = \begin{bmatrix} 1 & \boldsymbol{x}_1^\top \\ 1 & \boldsymbol{x}_2^\top \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_n^\top \end{bmatrix} = \begin{bmatrix} 1 & x_{1,1} & \ldots, & x_{1,p} \\ 1 & x_{2,1} & \ldots, & x_{2,p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n,1} & \ldots, & x_{n,p} \end{bmatrix},$$

which is an $n \times (p+1)$ matrix. Then, the linear model (1.2) can be written as

$$\boldsymbol{y} = X\beta + \boldsymbol{\epsilon},$$

Least Squares Estimation: Again, the most common method for estimating the parameter evctor β in linear regression is the least squares method. In this general setting, let

$$C(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_i))^2,$$

which is the mean squared error. Then, in linear regression, we aim to solve,

$$\min_{\beta} C(\beta).$$

A solution to the above problem, denoted as $\hat{\beta}$, is the least squares estimates of β.

Remark 1.6.1

Unlike the simple case with scalar independent variable, for the general case with vector independent variable, there is no unique solution for $\hat{\beta}$ except when $X^\top X$ is invertible. In that case,

$$\hat{\beta} = (X^\top X)^{-1}(X^\top \boldsymbol{y}).$$

When we $X^\top X$ is not invertible, we replace $(X^\top X)^{-1}$ with the *pseudo-inverse* of $X^\top X$ to get a solution.

Model Evaluation

After obtaining the least squares estimates, it is essential to evaluate the goodness of fit of the linear regression model. Common metrics for evaluation include the coefficient of determination R^2 (defined later in the lecture), which measures the proportion of variance explained by the model, and the standard error of the regression, which measures the average deviation of the observed values from the regression line.

Extensions

Linear regression can be extended to handle more complex relationships by including higher-order terms or interactions between variables. Additionally, various techniques exist to deal with issues such as multicollinearity, heteroscedasticity, and outliers.

Example 1.6.2

Re-condisder the Boston housing price dataset. This time assume that the independent variable x_i is lower status of the population in %. As shown in Figure 1.7, the relationship between x_i and the median housing price y_i is linear. However, we get a better fit by assume

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i, \quad i = 1, \ldots, n. \tag{1.3}$$

Again, refer to Section 2.1 of Chapter 2 in the book *Mathematical Engineering of Deep Learning*.

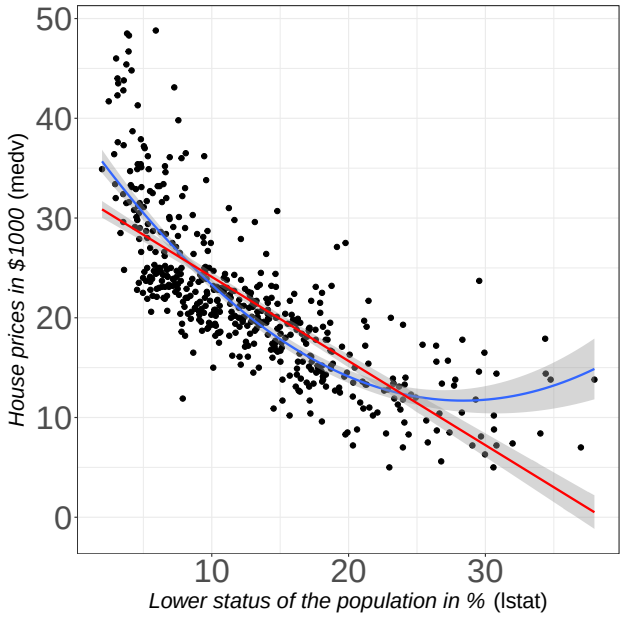


Figure 1.7: Example of an extended simple linear model. House prices as a function of lower status of the population in % (`lstat`) is not described well with a linear relationship (red), but by introducing an additional quadratic engineered feature it is described well via a three parameter linear model (1.3) resulting in a quadratic fit (blue).

1.7 Correlation coefficient and R^2 Score

The correlation coefficient and R^2 score are important measures used to quantify the strength and direction of the relationship between two variables in statistics. In this section, we present the definitions and interpretations of these measures.

Correlation Coefficient

The *correlation coefficient*, also known as *sample Pearson correlation coefficient*, denoted as r, measures the strength and direction of a linear relationship between two variables. For a sample of n observations (x_i, y_i), the correlation coefficient is given by:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of the two variables x and y, respectively. The correlation coefficient r ranges from -1 to 1, where:

- $r = 1$ indicates a perfect positive linear relationship.
- $r = -1$ indicates a perfect negative linear relationship.

- $r = 0$ indicates no linear relationship.

Further information about interpreting r is given below:

- Exactly –1. A perfect downhill (negative) linear relationship
- –0.70. A strong downhill (negative) linear relationship
- –0.50. A moderate downhill (negative) relationship
- –0.30. A weak downhill (negative) linear relationship
- 0.00. No linear relationship
- +0.30. A weak uphill (positive) linear relationship
- +0.50. A moderate uphill (positive) relationship
- +0.70. A strong uphill (positive) linear relationship
- Exactly +1. A perfect uphill (positive) linear relationship

R^2 Score

The *coefficient of determination*, denoted as R^2, is a measure of how well the independent variable(s) explain the variability of the dependent variable. It is defined as the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

For a regression model with predicted values $\hat{y}_i$ and observed values y_i, the R^2 score is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\bar{y}$ is the mean of the observed values y_i. The R^2 score ranges from 0 to 1, where:

- $R^2 = 1$ indicates that the regression model perfectly predicts the dependent variable.
- $R^2 = 0$ indicates that the regression model does not explain any of the variability of the dependent variable.

Interpretation

Both the correlation coefficient and R^2 score provide insights into the relationship between variables. While the correlation coefficient measures the strength and direction of a linear relationship, the R^2 score quantifies the proportion of variance explained by a regression model.