ZZSC5836: Data Mining and Machine Learning

Week 1: Introduction to learning algorithms and data mining

Dr Sarat Moka *Hexamester 3, 2024*

Key subtopics

- Data Mining vs Machine Learning
- Gradient descent
- Linear regression
- Correlation coefficient and R^2 Score
- Logistic regression
- Measurements of error
- Bias and variance
- Cross validation
- Confusion matrix
- Receiver Operating Characteristics (ROC) curve and Area Under The Curve (AUC)
- Regularisation
- One hot encoding

Books:

(A) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd Ed.) by Aurélien Géron (2019).

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

1.1 Data Mining vs Machine Learning

What is data mining (DM)?

Data mining is the process of *discovering patterns, trends, and insights from large datasets* using various techniques from statistics, machine learning, and database systems. It involves analyzing large volumes of data to uncover hidden patterns, relationships, and knowledge that can be used to make informed decisions, predict future trends, or optimize processes.

Key aspects of data mining include:

1. **Data Preparation:** This involves collecting, cleaning, and preprocessing raw data to ensure its quality and suitability for analysis. Data preprocessing may include tasks such as handling missing values, removing outliers, and transforming variables.
2. **Exploratory Data Analysis (EDA):** EDA involves exploring the data visually and statistically to gain insights into its structure, distribution, and relationships between variables. Techniques such as summary statistics, data visualization, and correlation analysis are commonly used in EDA.
3. **Feature Selection and Engineering:** Feature selection involves identifying the most relevant variables or features that are predictive of the target variable. Feature engineering involves creating new features from existing ones to improve model performance.
4. **Model Building and Evaluation:** This involves selecting appropriate machine learning or statistical models, training them on the data, and evaluating their performance using various metrics. Common data mining techniques include classification, regression, clustering, association rule mining, and anomaly detection.
5. **Interpretation and Visualization:** After building and evaluating models, it's essential to interpret the results and communicate insights effectively. Visualization techniques such as charts, graphs, and dashboards are often used to present findings in a clear and understandable manner.

Data mining is widely used across various industries and domains, including finance, healthcare, marketing, retail, and telecommunications. It helps organizations make data-driven decisions, improve business processes, detect fraud, identify customer segments, and optimize operations. With the increasing availability of big data and advanced analytics tools, data mining continues to play a crucial role in extracting actionable insights from complex datasets.

What is machine learning (ML)?

Machine learning focuses on the *development of algorithms and statistical models that enable computers to learn from and make predictions or decisions based on data*, without being explicitly programmed to perform specific tasks. In essence, machine learning algorithms learn patterns and relationships from data to improve their performance over time.

Key aspects of machine learning include:

1. **Learning from Data:** Machine learning algorithms learn from example data, known as training data, to identify patterns, trends, and relationships that can be used to make predictions or decisions on new, unseen data. The quality and quantity of the training data are crucial factors in the performance of machine learning models.

2. **Generalization:** Machine learning algorithms aim to generalize patterns learned from the training data to make accurate predictions or decisions on new, unseen data. Generalization ensures that the model can perform well on data it has not encountered during training.

3. **Types of Learning:** Machine learning can be broadly categorized into three main types of learning:

   – *Supervised Learning:* In supervised learning, the algorithm learns from labeled data, where each example is associated with a target output. The goal is to learn a mapping from input variables to output variables.

   – *Unsupervised Learning:* In unsupervised learning, the algorithm learns from unlabeled data, where the goal is to discover hidden patterns or structures in the data without explicit guidance.

   – *Reinforcement Learning:* In reinforcement learning, the algorithm learns to make sequential decisions by interacting with an environment and receiving feedback in the form of rewards or penalties.

4. **Algorithms and Models:** Machine learning algorithms encompass a wide range of techniques and models, including decision trees, random forests, support vector machines, neural networks, and clustering algorithms, among others. Each algorithm is suited to different types of tasks and data.

5. **Applications:** Machine learning has applications across various domains, including but not limited to:

   - Natural Language Processing (NLP)

   - Computer Vision

   - Speech Recognition

- Recommender Systems
- Predictive Analytics
- Healthcare
- Finance
- Autonomous Vehicles

In summary, machine learning enables computers to learn from data and make predictions or decisions in complex and uncertain environments, making it a powerful tool for solving a wide range of real-world problems.

What is the relationship between DM and ML?

Data mining and machine learning are closely related fields that share many common techniques and objectives, but they also have some key differences.

1. **Objective:**

   - *Data Mining:* The primary objective of data mining is to extract useful information and insights from large datasets. It focuses on discovering patterns, trends, and relationships in data to help businesses make better decisions and gain a competitive advantage.

   - *Machine Learning:* Machine learning aims to develop algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed. It focuses on building predictive models that can generalize from training data to new, unseen data.

2. **Techniques:**

   - *Data Mining:* Data mining techniques include exploratory data analysis, clustering, association rule mining, anomaly detection, and visualization. These techniques are used to uncover hidden patterns and structures in data.

   - *Machine Learning:* Machine learning techniques include supervised learning, unsupervised learning, reinforcement learning, and semi-supervised learning. These techniques involve training algorithms on labeled or unlabeled data to learn patterns and make predictions or decisions.

3. **Data Types:**

   - *Data Mining:* Data mining often deals with structured and unstructured data from various sources, including databases, text documents, images, and sensor data.

- *Machine Learning:* Machine learning algorithms can handle various types of data, including numerical data, categorical data, text data, and image data. The choice of algorithm depends on the type and characteristics of the data.

4. **Applications:**

   - *Data Mining:* Data mining is used in various industries and domains, including marketing, finance, healthcare, retail, telecommunications, and manufacturing. It helps businesses analyze customer behavior, detect fraud, optimize processes, and improve decision-making.

   - *Machine Learning:* Machine learning has a wide range of applications, including natural language processing, computer vision, speech recognition, recommender systems, predictive analytics, autonomous vehicles, and robotics.

5. **Overlap:**

   - There is significant overlap between data mining and machine learning, as both fields use statistical and computational techniques to analyze and extract knowledge from data.

   - Many machine learning algorithms, such as decision trees, support vector machines, neural networks, and k-means clustering, are also used in data mining.

   - Conversely, data mining techniques, such as association rule mining and clustering, can be considered as specific applications of machine learning.

In conclusion, data mining and machine learning are closely related fields that share common objectives and techniques but have distinct focuses and applications. They complement each other and are often used together to extract valuable insights and knowledge from data.

Further details on the overlap between DM and ML

While data mining and machine learning have traditionally been viewed as distinct fields with different focuses and objectives, the boundaries between them have become increasingly blurred in recent years. As a result, it can be challenging to distinguish between these two areas, and they are often used interchangeably or in conjunction with each other in practice. Here are some reasons why:

1. **Convergence of Techniques:** Many techniques and algorithms originally developed in one field are now widely used in the other. For example, machine learning algorithms such as decision trees, neural networks, and support vector machines are commonly used in data mining applications. Similarly, data mining techniques such as clustering and association rule mining are used in various machine learning tasks.

2. **Integration of Tools and Platforms:** Modern data analysis tools and platforms often integrate both data mining and machine learning capabilities. For example, popular tools like Python's scikit-learn library and R's caret package provide a wide range of algorithms for both data mining and machine learning tasks. Similarly, cloud-based platforms such as Google Cloud Platform and Microsoft Azure offer integrated services for data analysis, including both data mining and machine learning functionalities.

3. **Interdisciplinary Nature:** Both data mining and machine learning draw from a wide range of disciplines, including statistics, computer science, mathematics, and domain-specific areas such as biology, finance, and healthcare. As a result, researchers and practitioners in both fields often collaborate and leverage techniques from each other's domains to solve complex problems.

4. **Common Objectives:** Despite their historical differences, data mining and machine learning share common objectives, such as extracting useful insights from data, building predictive models, and making data-driven decisions. As a result, the distinctions between these areas have become less pronounced, and they are often viewed as complementary approaches to data analysis rather than separate disciplines.

While there may still be some conceptual distinctions between data mining and machine learning, these boundaries are becoming increasingly fluid as both fields continue to evolve and converge. In practice, researchers and practitioners often draw on techniques and methodologies from both areas to address real-world data analysis challenges effectively.

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

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ C(\boldsymbol{\theta}).$$

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

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

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

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

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

Algorithm

The gradient descent algorithm can be summarized as follows:

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

Convergence

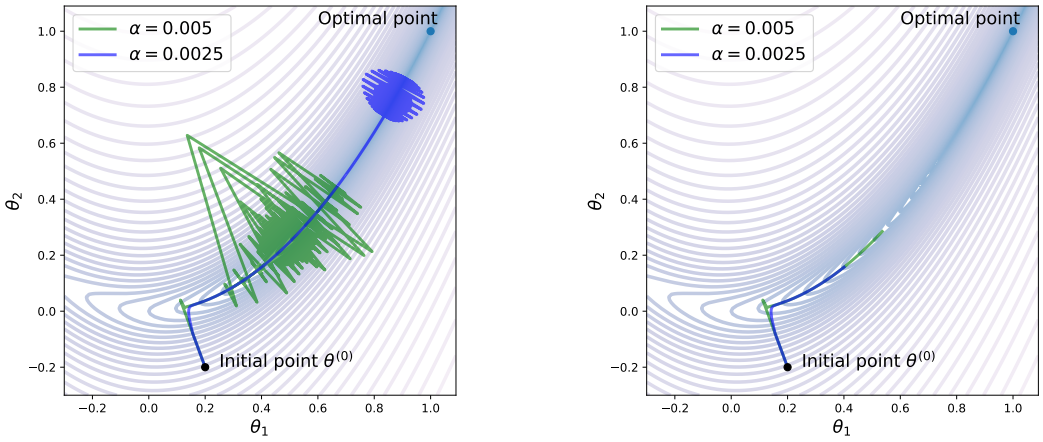
The convergence of gradient descent to the optimal minimum point $\boldsymbol{\theta}^*$ depends on various factors, including the choice of the learning rate, the properties of the objective function C, and the initial point $\boldsymbol{\theta}^{(0)}$. In general, if the learning rate is too large, 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.

Example

Consider the following 2-dimensional Rosenbrock function:

$$C(\boldsymbol{\theta}) = (1-\theta_1)^2 + 100(\theta_2 - \theta_1^2)^2. \tag{1.1}$$

This function has unique minimum at $\boldsymbol{\theta}^* = (1,1)$. We applied gradient descent on this function. Figure 1.1 illustrates how the learning rate effects the convergence. Python code for generating these figures can be found here.

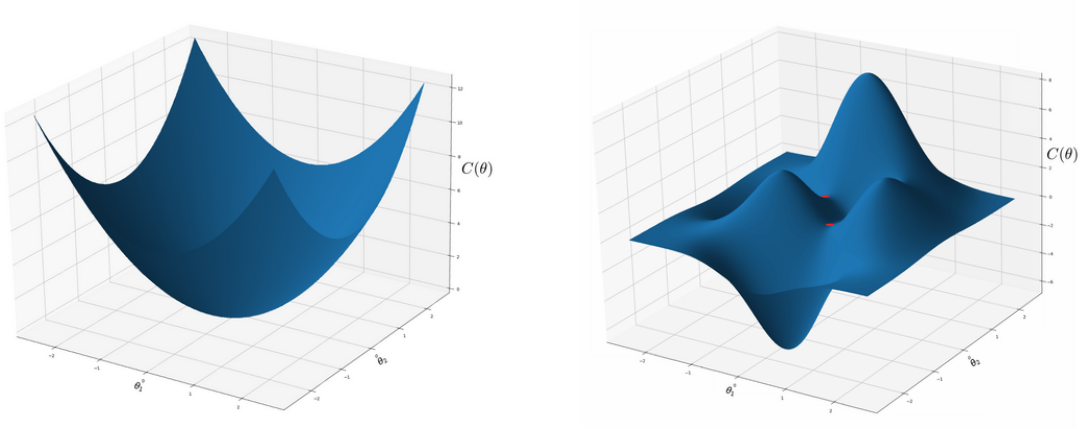


(a) (b)

Figure 1.1: Attempting minimization of a Rosenbrock function via basic gradient descent for two values of α with 10^5 iterations and $\boldsymbol{\theta}^{(0)} = (0.2, -0.2)$. (a) Fixed learning rate: $\alpha^{(t)} = \alpha$. (b) Exponentially decaying learning rate: $\alpha^{(t)} = \alpha\, 0.99^t$.

Example

Figure shows examples of convex and non-convex functions. For convex functions, we can often guarantee converges of gradient descent to a 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 1.2: Two dimensional smooth functions. (a) The function $C(\boldsymbol{\theta}) = \boldsymbol{\theta}_1^2 + \boldsymbol{\theta}_2^2$ 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. For complete details, read Chapter 4 of the book *Mathematical Engineering of Deep Learning*.

1.3 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 $\boldsymbol{\beta} = (\beta_0, \beta_1)$, let

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

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

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \; C(\boldsymbol{\beta}).$$

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

We consider Boston housing dataset which is originally published in [1], has 506 observation where each observation is associated with a suburb or town in the Boston Massachusetts area. Figure 1.3 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*.

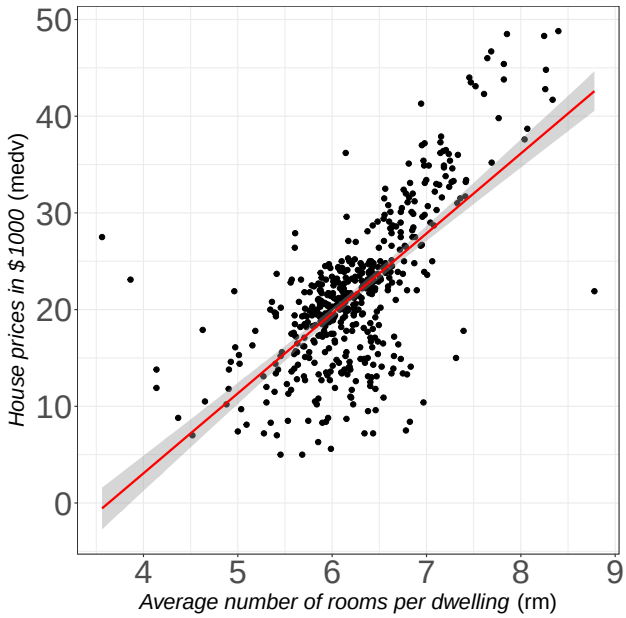


Figure 1.3: Example of simple linear model. Median house prices per locality (`medv`) as a function of average number of rooms per dwelling (`rm`) is described via a simple linear (affine) relationship.

Formulation for General Case

We now focus on case case where independent variable is a vector with p features. Consider a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,p}) \in \mathbb{R}^p$ 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 $\boldsymbol{\theta} = \boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p)$ 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

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

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

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

Least Squares Estimation: Again, the most common method for estimating the parameter evctor $\boldsymbol{\beta}$ in linear regression is the least squares method. In this general setting, let

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

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

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \;\; C(\boldsymbol{\beta}).$$

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

Remark

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

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

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

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

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 1.4, 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*.

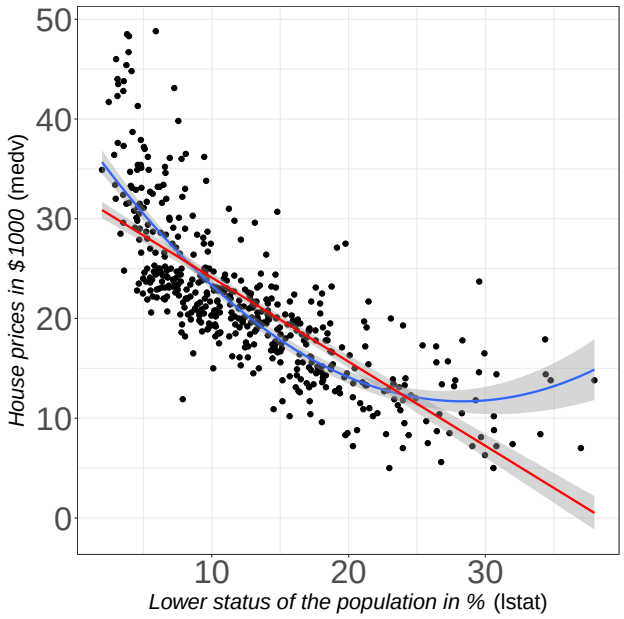


Figure 1.4: Example of an extended simple linear model. House prices as a function of lower status of the population in % (`lstat`) is not described well with a linear relationship (red), but by introducing an additional quadratic engineered feature it is described well via a three parameter linear model (1.3) resulting in a quadratic fit (blue).

1.4 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.

1.5 Logistic regression

Logistic regression is a statistical method used for binary classification tasks. It models the probability of a binary outcome (such as success/failure or 0/1) given one or more independent variables. In this section, we present the basic formulation and interpretation of logistic regression.

Formulation

Let $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,p})$ be a vector of independent variables in the i-th sample and y_i be the corresponding binary dependent variable. The logistic regression model assumes a linear relationship between the independent variables and the log odds of the binary outcome:

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots + \beta_p x_{i,p}, \quad i = 1, \ldots, n.$$

where p_i is the probability of the positive outcome, and $\beta_0, \beta_1, \ldots, \beta_p$ are the coefficients to be estimated.

In other words, for each vector of independent variable $\mathbf{x}_i = (x_{1,1}, \ldots, x_{i,p})$, the corresponding depedent variable y_i takes values 0 or 1 with

$$\mathbb{P}(y_i = 1) = p_i, \quad \text{and} \quad \mathbb{P}(y_i = 0) = (1 - p_i).$$

Logistic Function: The *logistic function*, also known as the *sigmoid function*, is used to transform the linear combination of independent variables into a probability between 0 and 1:

$$p_i = \frac{1}{1 + \exp(-(\beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \ldots + \beta_p x_{i,p}))}$$

where exp is the exponential function.

Interpretation The coefficients $\beta_1, \beta_2, \ldots, \beta_p$ represent the change in the log odds of the binary outcome for a one-unit change in the corresponding independent variable, holding all other variables constant.

Model Training

To estimate the coefficients of the logistic regression model, various optimization techniques such as maximum likelihood estimation or gradient descent are used. The goal is to find the coefficients that maximize the likelihood of the observed data given the model.

Model Evaluation

Once the logistic regression model is trained, it is evaluated using metrics such as accuracy, precision, recall, F1 score, and the receiver operating characteristic (ROC) curve. These metrics assess the performance of the model in predicting the binary outcome.

Extensions

Logistic regression can be extended to handle multi-class classification tasks using techniques such as one-vs-rest or multinomial logistic regression. Additionally, regularization techniques such as L1 and L2 regularization can be applied to prevent overfitting; more details later.

Remark

For complete details on the logistic regression, refer to Section 3.1 in Chapter 3 of the book *Mathematical Engineering of Deep Learning*.

Logistic Regression as a Shallow Neural Network

Let us first represent the logistic regression model as

$$\hat{y} = p = \sigma \underbrace{\left(\overbrace{b + \boldsymbol{w}^\top \mathbf{x}}^{z} \right)}_{a}, \tag{1.4}$$

with $b = \beta_0$, $\boldsymbol{w} = (w_1, \ldots, w_p) = (\beta_1, \ldots, \beta_p)$, and σ denoting the sigmoid function, which we refer to as a scalar *activation function*. In this form, logistic regression model is a neural network; see Figure 1.5.

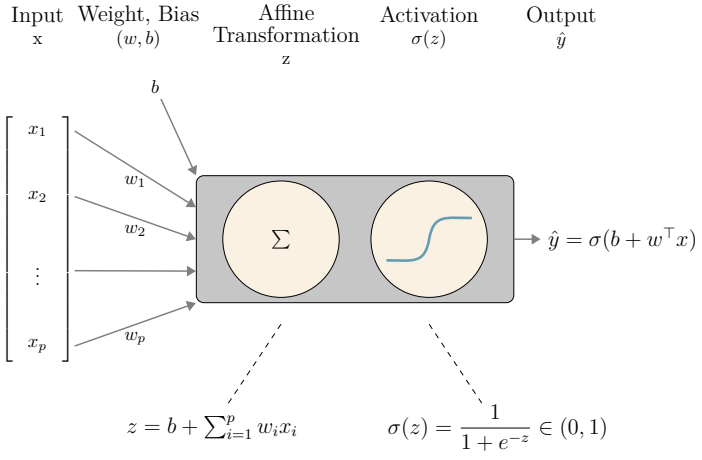


Figure 1.5: Logistic regression represented with neural network terminology as a shallow neural network. The gray box represents an artificial neuron composed of an affine transformation to create z and an activation $\sigma(z)$.

Remark

A trivial alternative activation function to the logistic function is the *identity activation function* $\sigma(z) = z$. With this identity activation function, the model in (1.4) is clearly just the linear model $\hat{y} = b + w^\top x$, which we studied in linear regression.

1.6 Measurements of error

Measurement errors play a crucial role in both regression and classification tasks. In this section, we provide definitions of various types of measurement errors and commonly used loss functions for evaluating the performance of models.

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:

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

- **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

$$\ell(y_i, \hat{y}_i) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| \leq \delta \\ \delta(|y_i - \hat{y}_i| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}.$$

- **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

$$\ell(y_i, \hat{y}_i) = \begin{cases} \tau(y_i - \hat{y}_i) & \text{if } y_i - \hat{y}_i \geq 0 \\ (\tau - 1)(y_i - \hat{y}_i) & \text{otherwise} \end{cases}.$$

Remark

To see the relationship between MSE, MAE and Huber loss, refer to Figure 1.6.

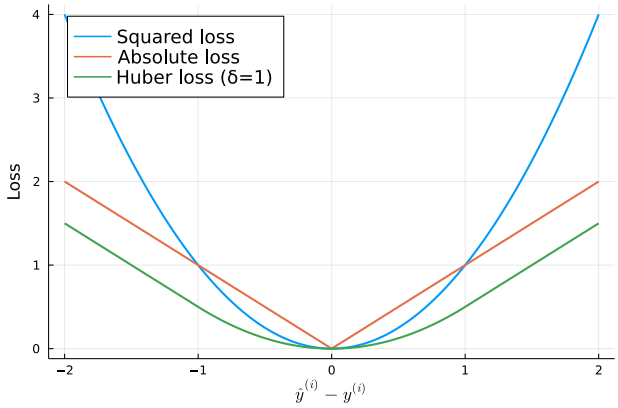


Figure 1.6: Loss function and error distribution alternatives. (a) Squared, absolute, and Huber loss functions.

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 $\hat{\boldsymbol{p}}_i = (\hat{p}_{i,1}, \ldots, \hat{p}_{i,C})$ with C denoting that number of classes and $\hat{p}_{i,c} = \mathbb{P}(y_i = c)$. 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.

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

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

These loss functions provide different ways to measure the discrepancy between the predicted and actual outcomes in both regression and classification tasks.

1.7 Bias and variance

Bias and variance are two fundamental concepts in machine learning and statistics that describe the behavior of models in terms of their prediction errors. In this section, we discuss bias, variance, and the bias-variance tradeoff.

Bias: Bias refers to the error introduced by approximating a real-world problem with a simplified model. A model with high bias makes strong assumptions about the underlying data and may fail to capture complex patterns. Common examples of high bias models include linear regression with insufficient features and decision trees with shallow depths.

Mathematically, bias is defined as the difference between the expected prediction of the model $\hat{y} = \hat{f}(\mathbf{x})$ and the true value $y = f(\mathbf{x})$ for an unseen data point $(\mathbf{x}, y)$:

$$\text{Bias}(\hat{y}) = \mathbb{E}[\hat{y}] - y$$

where $\mathbb{E}[\hat{y}]$ is the expected value of the predicted value.

Variance: Variance measures the variability of the model's predictions for a given data point. A model with high variance is sensitive to small fluctuations in the training data and may overfit by capturing noise instead of true patterns. Examples of high variance models include decision trees with deep depths and complex neural networks.

Mathematically, variance is defined as the expected squared difference between the predicted value of the model and the expected prediction:

$$\text{Var}(\hat{y}) = \mathbb{E}[(\hat{y} - \mathbb{E}[\hat{y}])^2] = \mathbb{E}[\hat{y}^2] - (\mathbb{E}[\hat{y}])^2$$

Bias-Variance Tradeoff: The bias-variance tradeoff refers to the balance between bias and variance when designing machine learning models. In general, increasing the complexity of a model reduces bias but increases variance, and vice versa. The goal is to find the optimal balance that minimizes the overall prediction error.

- **High Bias, Low Variance:** Simple models with high bias and low variance tend to underfit the data by oversimplifying the underlying patterns.
- **Low Bias, High Variance:** Complex models with low bias and high variance tend to overfit the data by capturing noise instead of true patterns.
- **Optimal Tradeoff:** The optimal tradeoff between bias and variance depends on the specific problem and the amount of available data. Techniques such as regularization and cross-validation can help find the optimal balance.

Understanding bias and variance is essential for designing effective machine learning models that generalize well to unseen data.

Generally, as model complexity increases, expected training performance improves (decreases) since complex structured models can explain the training data better. At high extremes this is overfitting. Similarly an opposite phenomenon is that models with low complexity are not able to describe the data well. The tradeoff between these two regimes is obtained at the minimum of the expected generalized performance on an unseen data, marked by the vertical dashed line in Figure 1.7.

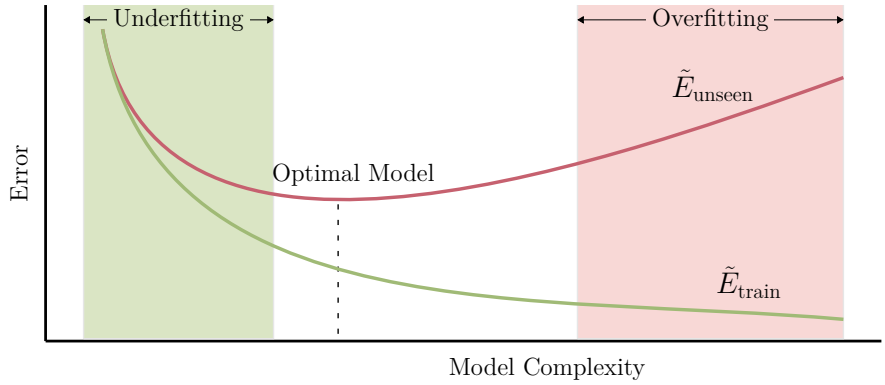


Figure 1.7: Typical behaviour of expected generalization performance on unseen data (red curve) and expected training performance on training data (blue) as a function of model complexity

Example

As one simple illustrative example capturing the tradeoffs of model complexity, let us consider linear models with polynomial features applied to synthetic univariate ($p = 1$) data. The model

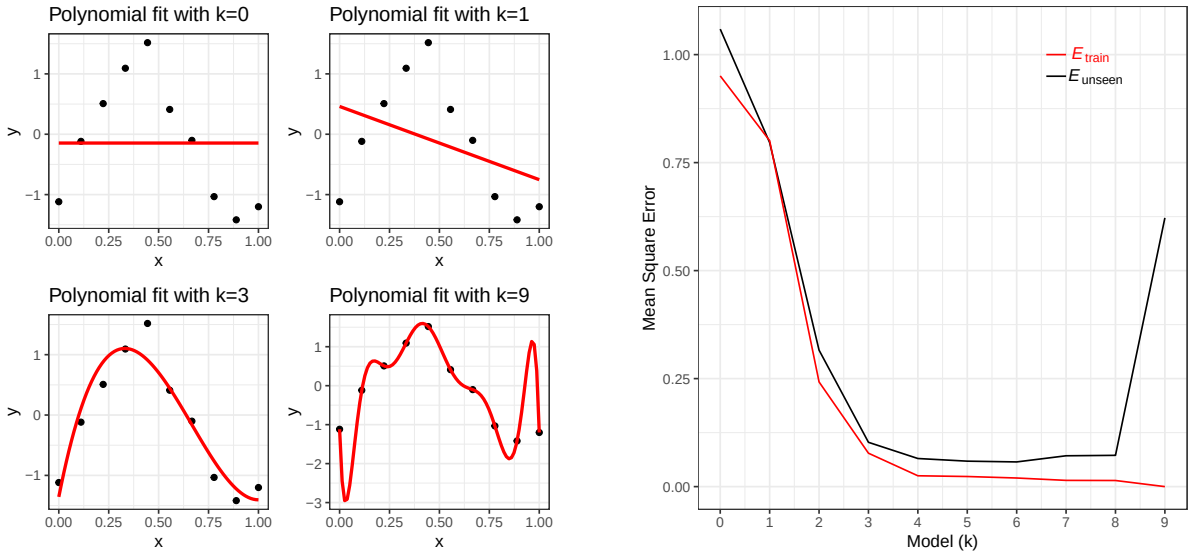
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \ldots + \beta_k x^k + \epsilon, \tag{1.5}$$

where k is the order of the polynomial. Hence, the model with $k = 0$ is the constant model, the model with $k = 1$ is the simple linear model, the model with $k = 2$ is the quadratic model, and so on. In this framework, model complexity corresponds to the degree of the polynomial model is illustrated in Figure 1.8.

1.8 Cross validation

Cross-validation is a widely used technique in machine learning for assessing the generalized performance of a predictive model. It helps to estimate how well the model will generalize to unseen data. In this section, we discuss the concept of cross-validation and various commonly used cross-validation techniques.

The main idea behind cross-validation is to partition the available data into multiple subsets, called folds. The model is trained on a subset of the data and evaluated on the remaining data. This process is repeated multiple times, with different subsets used for training and evaluation each time. The performance metrics obtained from each iteration are then averaged to obtain a more reliable estimate of the model's performance.



(a) (b)

Figure 1.8: Increasing model complexity illustrated via linear models with polynomial features where k, the order of the polynomial, captures the complexity. (a) Fitting several models to a single realization with $n = 10$ data-points. (b) The training performance in red and simulation estimates of the generalization performance in black.

K-Fold Cross-Validation

K-fold cross-validation is one of the most commonly used cross-validation techniques. In K-fold cross-validation, the given data is divided into K subsets of approximately equal size. The model is trained K times, each time using K-1 folds for training and the remaining fold for evaluation. The performance metrics obtained from each iteration are then averaged to obtain the final performance estimate which provides an estimation of the expected generalized performance; see Figure 1.9 for an illustration.

Leave-One-Out Cross-Validation (LOOCV)

Leave-One-Out Cross-Validation (LOOCV) is a special case of K-fold cross-validation where K equals the number of data points in the dataset. In LOOCV, the model is trained K times, each time using all but one data point for training and the remaining data point for evaluation. This process is repeated for each data point in the dataset. LOOCV provides an unbiased estimate of the model's performance but can be computationally expensive, especially for large datasets.

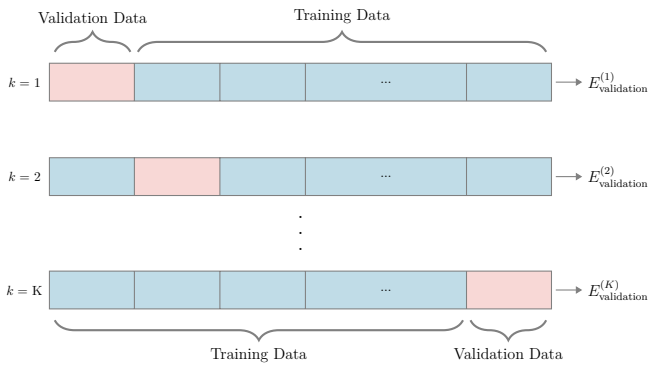


Figure 1.9: K-fold cross validation. For each $k = 1, \ldots, K$ the data is split into training data and validation data differently. This yields K estimates for performance and these estimates can be averaged.

Stratified Cross-Validation

Stratified cross-validation is used when the dataset is imbalanced, i.e., the distribution of classes is uneven. In stratified cross-validation, the data is divided into folds such that each fold contains approximately the same proportion of samples from each class as the original dataset. This ensures that the model is trained and evaluated on representative samples from each class.

Nested Cross-Validation

Nested cross-validation is used for model selection and hyperparameter tuning. In nested cross-validation, the data is divided into an outer loop and an inner loop. The outer loop is used for model evaluation, while the inner loop is used for hyperparameter tuning. Nested cross-validation provides an unbiased estimate of the model's performance and helps to prevent overfitting during model selection.

Conclusion

Cross-validation is a powerful technique for estimating the performance of predictive models and selecting the best model for a given task. By using cross-validation, researchers and practitioners can ensure that their models generalize well to unseen data and make reliable predictions in real-world scenarios.

1.9 Confusion matrix

A confusion matrix is a performance measurement tool used in classification tasks to evaluate the performance of a predictive model. It provides a summary of the predicted and actual class labels for a given dataset. In this section, we discuss the components of a confusion matrix and how it is used to assess the performance of a classification model.

Components of a Confusion Matrix

A confusion matrix is typically represented as a square matrix with rows and columns corresponding to the actual and predicted class labels, respectively. To define the components of this matrix, suppose for a classifier is constructed via a decision rule based on a threshold τ, with the predicted output being,

$$\hat{Y}_i = \begin{cases} 1, & \text{if } \hat{p}_i \geq \tau, \\ 0, & \text{if } \hat{p}_i < \tau, \end{cases} \tag{1.6}$$

Then the main components of a confusion matrix are as follows:

- **True Positives (TP):** The number of instances that belong to the positive class (actual positive) and are correctly predicted as positive. That is, the number of instances i where $\hat{Y}_i = 1$ and $y_i = 1$.
- **False Positives (FP):** The number of instances that belong to the negative class (actual negative) but are incorrectly predicted as positive. That is, the number of instances i where $\hat{Y}_i = 1$ for $y_i = 0$.
- **True Negatives (TN):** The number of instances that belong to the negative class (actual negative) and are correctly predicted as negative. That is, the number of instances i where $\hat{Y}_i = 0$ and $y_i = 0$.
- **False Negatives (FN):** The number of instances that belong to the positive class (actual positive) but are incorrectly predicted as negative. That is, the number of instances i where $\hat{Y}_i = 0$ for $y_i = 1$.

Interpretation

The confusion matrix provides a detailed breakdown of the model's performance in terms of true positives, false positives, true negatives, and false negatives. It allows us to calculate various performance metrics, such as accuracy, precision, recall, and F1 score, which provide insights into different aspects of the model's performance.

Example

Consider a binary classification task where we are predicting whether an email is spam (positive class) or not spam (negative class). A confusion matrix for this task might look like the following:

$$\begin{bmatrix} \text{TN} & \text{FP} \\ \text{FN} & \text{TP} \end{bmatrix} = \begin{bmatrix} 950 & 20 \\ 30 & 1000 \end{bmatrix}$$

In this example, there are 950 true negatives (non-spam emails correctly classified), 20 false positives (non-spam emails incorrectly classified as spam), 30 false negatives (spam emails incorrectly classified as non-spam), and 1000 true positives (spam emails correctly classified).

Conclusion

The confusion matrix is a valuable tool for evaluating the performance of classification models and gaining insights into their strengths and weaknesses. By analyzing the components of the confusion matrix, we can identify areas for improvement and make informed decisions about model selection and optimization.

1.10 Receiver Operating Characteristics (ROC) curve

The Receiver Operating Characteristics (ROC) curve is a graphical tool widely used in binary classification to assess the performance of a classifier across different decision thresholds. Let's explore the mathematical formulation and interpretation of the ROC curve in detail.

Mathematical Concept

The ROC curve is a plot of the true positive rate (TPR) (sensitivity) against the false positive rate (FPR) (1 - specificity) for various decision thresholds τ. To define formally, recall the binary classifier given in (1.6) and define,

$$\text{TPR}(\tau) = \frac{TP}{TP + FN}, \quad \text{and} \quad \text{FPR}(\tau) = \frac{FP}{TN + FP}.$$

Then, ROC is the curve of TPR(τ) vs FPR(τ)).

Interpretation

The ROC curve provides a visual representation of a classifier's trade-off between sensitivity and specificity at different decision thresholds. A curve that hugs the upper left corner indicates superior performance, as it achieves high sensitivity with low false positive rate. Conversely, a curve closer to the diagonal line suggests poorer discrimination ability.

Example

Consider a logistic regression set-up with logistic function being

$$p_i = \frac{1}{1+\exp(-(\beta_0+\beta_1 x_1+\beta_2 x_1^2+\cdots+\beta_k x_i^k))},$$

or, equivalently,

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0+\beta_1 x_1+\beta_2 x_1^2+\cdots+\beta_k x_i^k$$

is a polynomial of degree k. Compare this with (1.5).
Figure 1.10 illustrates ROC for an example dataset. For more details on ROC and the dataset used in the figure, refer to Section 2.2 in Chapter 2 of the book *Mathematical Engineering of Deep Learning*.

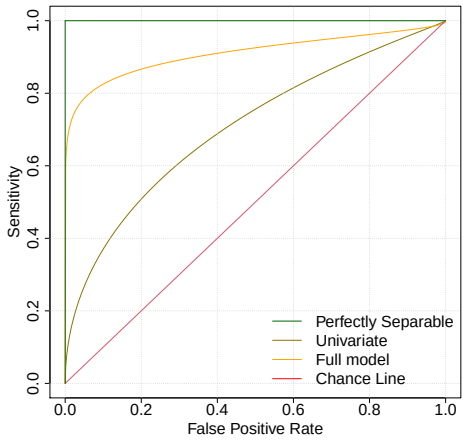


Figure 1.10: Receiver operating characteristic (ROC) curves for the breast cancer data. One model is a univariate model ($k = 1$), and the other is with $k = 30$. A chance line (guessing model) and a perfectly separable line (ideal model) are also plotted. For each model, the ROC captures the tradeoff between the sensitivity and the false positive rate (one minus specificity).

Advantages

The ROC curve is advantageous as it provides a comprehensive visualization of a classifier's performance across different decision thresholds. It allows for easy comparison between different classifiers and facilitates the selection of an appropriate operating point based on the specific application requirements.

Area Under The Curve (AUC)

ROC curves allow us to asses the quality of models taking all possible threshold parameters into account. A related measure that tries to quantify the quality of a curve into a single number is the *area under the curve*(AUC) measure. For a classifier with an ROC curves that achieves perfect sensitivity under any level of specificity this measure is at 1 and corresponds to the perfectly separable green curve in Figure 1.10. However for classifiers that just choose a random class, this measure is at 0.5 corresponding to the chance line, red line in Figure 1.10. In the case of the breast cancer data we see that on the test set the AUC for the $k = 1$ model is 0.70 and for the $k = 30$ model it is at 0.92. This may give an indication that the additional features in the richer model help obtain a better predictor.

Conclusion

The Receiver Operating Characteristics (ROC) curve is a valuable tool for evaluating the performance of binary classifiers. Its mathematical foundation and interpretation offer insights into the trade-off between sensitivity and specificity, aiding in informed decision-making in classification tasks.

1.11 Regularisation

Regularization is a technique used in machine learning and statistics to prevent overfitting by adding a penalty term to the loss function. It discourages overly complex models by penalizing large parameter values. In this section, we discuss three commonly used regularization techniques: Lasso, Ridge, and ElasticNet.

A common way to keep model parameters at bay is to augment the optimization objective $\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$ with an additional *regularization term* $R_\lambda(\boldsymbol{\theta})$. The revised objective is then,

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \;\; C(\boldsymbol{\theta}) + R_\lambda(\boldsymbol{\theta}). \tag{1.7}$$

The regularization term $R_\lambda(\theta)$ (also called penalty term) depends on a *regularization parameter* λ, which is often a scalar in the range $[0, \infty)$ but also sometimes a vector. This

hyper-parameter allows us to optimize the bias and variance tradeoff.

Lasso (L1 Regularization)

Lasso, short for Least Absolute Shrinkage and Selection Operator, adds a penalty term proportional to the absolute value of the coefficients to the loss function. Mathematically, the Lasso regularization term is defined as:

$$R_\lambda(\boldsymbol{\theta}) = \lambda \sum_{j=1}^{p} |\theta_j|.$$

where λ is the regularization parameter and β_j are the model coefficients. Lasso encourages sparsity in the model, leading to feature selection by shrinking some coefficients to zero.

Ridge (L2 Regularization)

Ridge regularization adds a penalty term proportional to the square of the coefficients to the loss function. Mathematically, the Ridge regularization term is defined as:

$$R_\lambda(\boldsymbol{\theta}) = \lambda \sum_{j=1}^{p} \theta_j^2.$$

Ridge regularization penalizes large coefficients while maintaining all features in the model. It tends to shrink the coefficients towards zero without completely eliminating them.

ElasticNet Regularization

ElasticNet regularization combines Lasso and Ridge penalties by adding both L1 and L2 regularization terms to the loss function. The ElasticNet penalty is defined as a weighted combination of the Lasso and Ridge penalties:

$$R_\lambda(\boldsymbol{\theta}) = \lambda_1 \sum_{j=1}^{p} |\theta_j| + \lambda_2 \sum_{j=1}^{p} \theta_j^2.$$

where λ_1 and λ_2 are the regularization parameters controlling the strength of L1 and L2 regularization, respectively.

Choice of Regularization Parameter

The choice of the regularization parameter (λ for Lasso and Ridge, λ_1 and λ_2 for ElasticNet) is crucial and is typically determined using techniques such as cross-validation or grid search.

Conclusion

Regularization techniques such as Lasso, Ridge, and ElasticNet are powerful tools for preventing overfitting and improving the generalization of machine learning models. By adding penalty terms to the loss function, these techniques encourage simpler models and feature selection, leading to better performance on unseen data.

1.12 One Hot Encoding

In deep learning, especially in tasks like natural language processing (NLP) and classification, one-hot encoding is a common technique used to represent categorical variables numerically. It involves converting categorical variables into binary vectors, where each vector has a length equal to the number of categories and contains a single '1' value at the index corresponding to the category, while all other elements are '0'.

- Denote a categorical variable with C categories as $c_1, c_2, \ldots, c_C$.
- The one-hot encoding of this variable would result in binary vectors $x_1, x_2, \ldots, x_C$, where:

$$x_i = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

  where j is the index of the category c_j in the original categorical variable.
- Mathematically, we can represent one-hot encoding using the Kronecker delta function. The Kronecker delta, denoted as δ_{ij}, is defined as:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

- Then, the ith element of the one-hot encoded vector x_i can be expressed as:

$$x_i = \delta_{ij}$$

  This means that the one-hot encoding of the category c_i results in a vector where the ith element is '1' and all other elements are '0'.
- In deep learning applications, one-hot encoding is commonly used as input representations for categorical variables in neural networks, particularly in tasks such as multi-class classification, where the output layer typically employs a softmax activation function. This encoding ensures that the neural network can effectively learn to predict probabilities across multiple classes.

- Despite its simplicity, one-hot encoding has some drawbacks, such as high dimensionality when dealing with a large number of categories and the inability to capture ordinal relationships between categories. However, it remains a fundamental technique in deep learning for handling categorical variables efficiently.

References

[1] D. Harrison Jr and D. L. Rubinfeld. Hedonic Housing Prices and the Demand for Clean Air. *Journal of Environmental Economics and Management*, 1978.