Chapter Title: The Learning Problem

1 Introduction

1.1 What is Machine Learning?

Machine learning is a broad subject that spans from abstract theory to practical rules of thumb. It is useful because it provides a conceptual framework and practical tools to deal with real learning systems. The inclusion of a topic in a machine learning course depends on its relevance to the field itself, balancing mathematical foundations with practical applications.

1.2 Course Storyline

This chapter, and the course, will follow a story line to understand the fundamentals of learning, starting with the basic questions and progressing to more advanced concepts. The storyline is as follows:

- What is learning?
- Can we learn?
- How to do it?
- How to do it well?
- Take-home lessons.

While the topics are presented in a logical sequence, some practical tools will be introduced early on to allow for experimentation with theoretical concepts, even before the complete theoretical framework is fully developed. The course will begin by building conceptual and theoretical foundations, and then transition towards more practical aspects of machine learning.

2 The Essence of Machine Learning: A Movie Rating Example

2.1 Predicting Movie Ratings

Consider the problem of predicting how a viewer would rate a movie. This is a pertinent problem for movie rental companies like Netflix, who famously offered a $1 million prize for a mere 10% improvement in their recommendation system. This improvement is valuable because better recommendations lead to increased user engagement and movie rentals, significantly boosting revenue. This example encapsulates the core of machine learning and its high-stakes applications in various fields, such as financial forecasting where even minute improvements can yield substantial financial gains.

2.2 Three Components of Machine Learning

The problem of movie rating effectively illustrates the three essential components that define a machine learning problem:

1. **A pattern exists**: There is an underlying pattern in how viewers rate movies. A person's rating is related to their past ratings and how others have rated the same movie.
2. **Mathematically intractable pattern**: This pattern, while real, cannot be easily described or formulated mathematically, for instance, through a simple polynomial equation.
3. **Data availability**: There must be data from which to learn this pattern. Without data, there is nothing to learn from.

The reliance on data to uncover patterns that are too complex to define mathematically is what necessitates and empowers machine learning. Without these three components, especially the presence of data, applying machine learning techniques becomes impractical.

2.3 A Proposed Solution

One approach to movie rating prediction involves describing both viewers and movies as vectors of factors. These factors could represent various aspects such as comedy content, action content, preference for blockbusters, or even preferences regarding lead actors.

For a **viewer**, the vector would represent their taste profile:

- Does the viewer like comedies?
- Does the viewer like action movies?
- Does the viewer prefer blockbusters or indie films?
- Does the viewer like specific actors?
- ... (and so on for hundreds of potential factors)

Similarly, for a **movie**, the vector would represent its content profile:

- Does the movie have comedy content?
- Does the movie have action?
- Is it a blockbuster-style movie?
- Does it star specific actors?
- ... (corresponding factors to the viewer profile)

The prediction is made by matching the movie and viewer factors. If there is a high degree of match between a viewer's preferences and a movie's content across these factors, the predicted rating would be high. Conversely, mismatches would lead to lower predicted ratings. The process involves:

1. Matching movie and viewer factors.
2. Adding the contributions from each factor match or mismatch.
3. Outputting a predicted rating based on the aggregated contributions.

However, this approach requires manual analysis of movies and viewers to determine their factor profiles, making it less of a 'learning' approach and more of an analytical one.

2.4 A Learning Approach: Reverse Engineering from Ratings

A true machine learning approach to movie ratings involves reverse-engineering the factor profiles directly from user ratings. Instead of pre-defining and analyzing movie and viewer factors, the learning process starts with observed ratings and works backward to infer these factors.

Initially, we can assume random factors for both viewers and movies. For each viewer and movie, assign random numerical values across all factors. When a viewer rates a movie, the system compares the predicted rating (derived from the inner product of their random factor vectors) with the actual rating. If there is a discrepancy, the system slightly adjusts the factors to move the predicted rating closer to the actual rating.

This adjustment process is repeated over a large dataset of ratings—potentially millions. By iteratively refining these random factors based on actual ratings, the system gradually learns meaningful and consistent factor vectors for viewers and movies.

After this learning process, when a new viewer who has not rated a particular movie is considered, the system can use the learned factor vectors. By calculating the inner product of the viewer's and the movie's factor vectors, the system can predict a rating that is likely to be consistent with how the viewer would actually rate the movie. This method, which emerged as a winning solution in the Netflix Prize competition, exemplifies how machine learning can automatically discover hidden patterns and relationships from data without explicit programming of rules or factor definitions.

3 Formalizing the Learning Problem: Credit Approval Metaphor

3.1 Credit Approval Scenario

To formalize the concept of learning, let's consider a different application: credit approval. Banks need to decide whether to grant credit to an applicant based on their information. Similar to movie ratings, there's no explicit formula to determine creditworthiness. Banks rely on historical data to make these decisions.

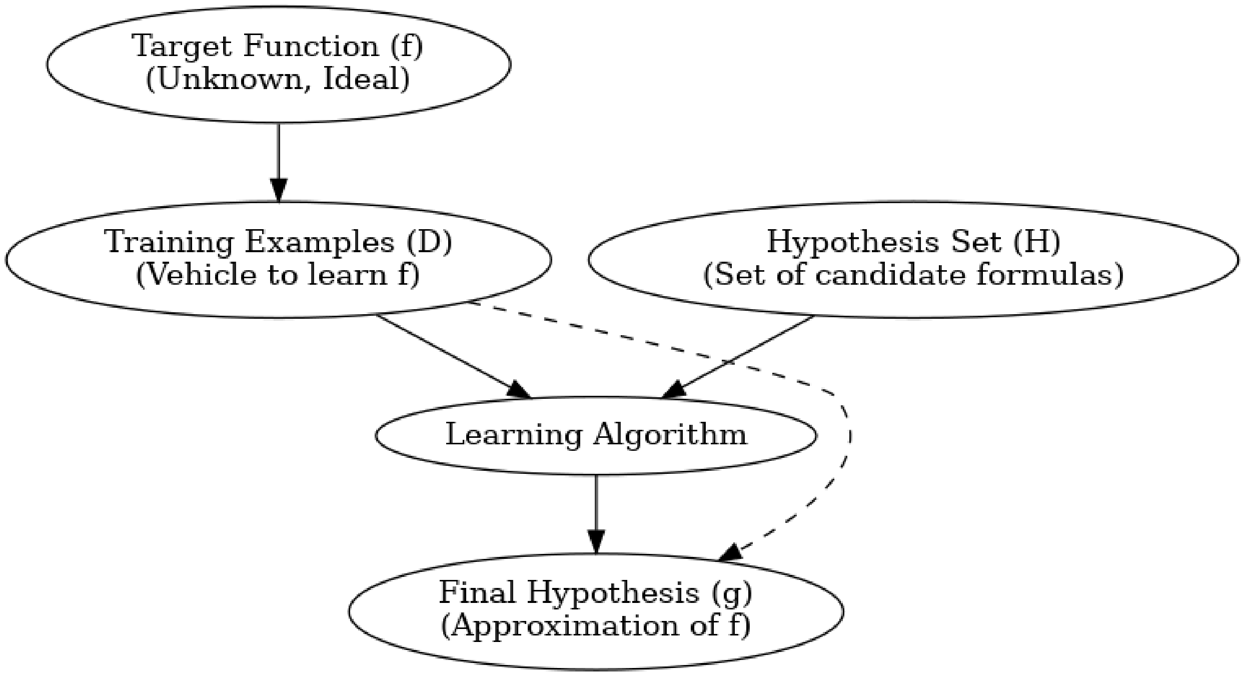
3.2 Components and Notations

In a credit approval scenario, we can identify key components of a learning problem and formalize them using mathematical notations.

1. **Input** ($\mathbf{x}$): This is the information provided by the applicant, such as age, income, debt, etc. Mathematically, this is represented as a d-dimensional vector $x = (x_1, x_2, ..., x_d)$, where each dimension corresponds to an attribute. The set of all possible input vectors is denoted by X.
2. **Output** ($\mathbf{y}$): This is the decision made by the bank, which is binary: approve credit (+1) or deny credit (-1). The set of possible outputs is $Y = \{+1, -1\}$.
3. **Target Function** ($\mathbf{f}$): This is the ideal credit approval formula, which is unknown. It's a function $f : X \to Y$ that perfectly determines if an applicant is creditworthy. In reality, this function is what we want to learn or approximate.
4. **Data** ($\mathbf{D}$): Historical records of previous customers. Each record consists of an applicant's information (input x) and their credit behavior in hindsight (output y). The dataset D consists of N such examples: $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$.
5. **Hypothesis** ($\mathbf{g}$): This is our approximation of the target function f. After learning from the data D, we aim to find a hypothesis $g : X \to Y$ that is as close to f as possible. Ideally, $g(x) \approx f(x)$ for most inputs x.

3.3 The Learning Diagram

We can visualize the learning process using a diagram:



Learning Diagram

The target function f is the actual relationship we want to learn, but it remains unknown. We only observe it through training examples D. A hypothesis set H is a collection of possible functions that our learning algorithm can choose from. The learning algorithm uses the training data D to select a final hypothesis g from H that best approximates the target function f. The hypothesis set is a crucial component as it defines the type of functions the learning algorithm will consider. Choosing an appropriate hypothesis set is important for effective learning.

4 The Perceptron Model: A Simple Learning Algorithm

4.1 The Perceptron Hypothesis Set

The perceptron model is a simple yet fundamental model in machine learning, particularly for binary classification problems like credit approval.

Given a d-dimensional input vector $x = (x_1, x_2, ..., x_d)$, the perceptron model uses a linear formula to make a decision. It calculates a weighted sum of the input features and compares it to a threshold. The formula is:

$$h(x) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right)$$

where:

- $x = (x_1, x_2, ..., x_d)$ is the input vector (customer application).
- $w = (w_1, w_2, ..., w_d)$ is the weight vector, where each w_i represents the importance of the i-th feature.
- threshold is a value against which the weighted sum is compared.
- $\text{sign}(z)$ is the sign function, which returns +1 if $z \geq 0$ and -1 if $z < 0$.

The weights w_i and the threshold are the parameters that define a specific perceptron hypothesis. The **hypothesis set** H for the perceptron model consists of all possible functions $h(x)$ that can be formed by varying the weights w and the threshold. Each choice of weights and threshold gives a different hypothesis $h \in H$. The learning algorithm's goal is to find the optimal weights and threshold based on the training data, resulting in a final hypothesis g that approximates the unknown target function f.

4.2 Visualizing Linear Separability

The perceptron model works best when the data is *linearly separable.* Consider a simplified 2-dimensional case where we want to classify customers as 'good' (+1) or 'bad' (-1) based on two features. Linearly separable data means we can draw a straight line (in 2D) or a hyperplane (in higher dimensions) to perfectly separate the 'good' customers from the 'bad' customers.

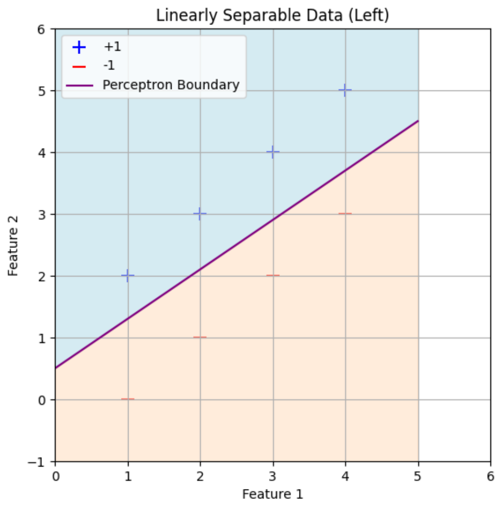


Figure 1: Linearly Separable Data - Initial Random Boundary

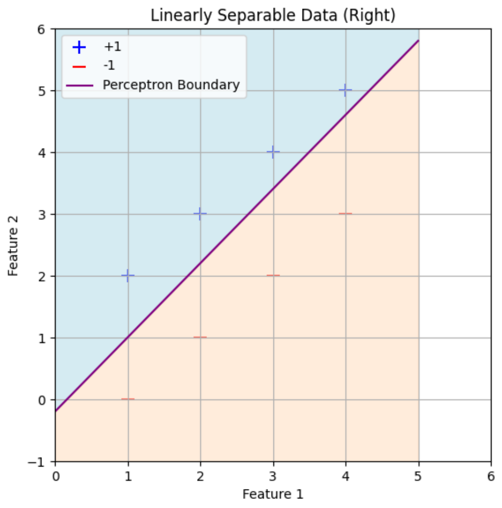


Figure 2: Linearly Separable Data - Learned Boundary

In these figures, blue '+' symbols represent 'good' customers (+1), and red '-' symbols represent 'bad' customers (-1). The purple line represents a perceptron hypothesis, acting as a decision boundary. On one side of the line (light blue region), customers are classified as 'good', and on the other side (light peach region), as 'bad'. Initially, with random weights, the perceptron boundary (purple line in the left plot) might not correctly separate the data. The learning algorithm adjusts the weights to move this line, aiming to achieve perfect separation as shown in the right plot.

4.3 Simplified Perceptron Hypothesis Notation

To simplify the notation, we can incorporate the threshold into the weighted sum. We introduce an artificial feature $x_0 = +1$ and a corresponding weight $w_0 = -\text{threshold}$. The perceptron formula then becomes:

$$h(x) = \text{sign}\left(\sum_{i=0}^{d} w_i x_i\right)$$

where $x = (x_0, x_1, ..., x_d)$ and $w = (w_0, w_1, ..., w_d)$. This can be further expressed in vector notation as:

$$h(x) = \text{sign}(w^T x)$$

where w and x are now column vectors, and w^T denotes the transpose of w. This form represents the hypothesis as the sign of the inner product of the weight vector w and the input vector x.

4.4 Perceptron Learning Algorithm (PLA)

The Perceptron Learning Algorithm (PLA) is an iterative algorithm used to find the weight vector w that correctly classifies all training examples, assuming the data is linearly separable.

Algorithm Steps:

1. **Initialization**: Start with an initial weight vector, e.g., $w = 0$.
2. **Iteration**:
   (a) **Find a misclassified point**: Iterate through the training dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$. Look for a point (x_n, y_n) that is misclassified by the current weight vector w. A point is misclassified if $\text{sign}(w^T x_n) \neq y_n$.

(b) **Update weight vector**: If a misclassified point (x_n, y_n) is found, update the weight vector using the rule:

$$w \leftarrow w + y_n x_n$$

3. **Repeat**: Repeat step 2 until no misclassified points are found in the dataset.

The update rule intuitively adjusts the weight vector w in the direction to correctly classify the misclassified point x_n. If $y_n = +1$ and $w^T x_n < 0$ (misclassified as -1), adding $y_n x_n = x_n$ to w tends to increase $w^T x_n$, making it more positive and thus more likely to be correctly classified as +1 in the next iteration. Similarly, if $y_n = -1$ and $w^T x_n > 0$ (misclassified as +1), adding $y_n x_n = -x_n$ tends to decrease $w^T x_n$, making it more negative and more likely to be correctly classified as -1.

4.5 Iterations of PLA

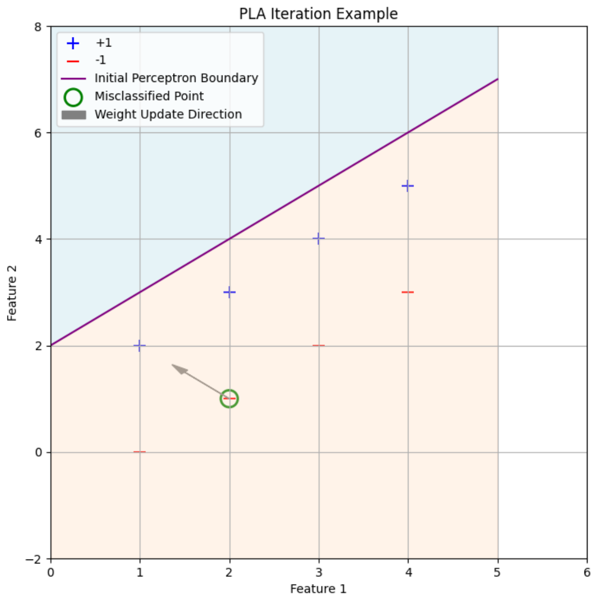


Figure 3: One Iteration of PLA - Adjusting the Boundary

In each iteration of PLA, the algorithm picks a misclassified point and adjusts the perceptron boundary (purple line) to correctly classify this point. As illustrated, if a blue '+' point falls into the peach puff region (incorrectly classified), or a red '-' point falls into the light blue region (incorrectly classified), the PLA update rule moves the purple line in a direction that reduces this misclassification. This iterative adjustment, when applied repeatedly to misclassified points, eventually leads to a boundary that correctly separates all points, provided the data is linearly separable.

4.6 Convergence of PLA

A crucial property of the PLA is its convergence guarantee. If the training data is indeed linearly separable, the PLA is guaranteed to converge to a weight vector w that correctly classifies all training examples in a finite number of iterations. This means the algorithm will eventually find a perceptron hypothesis that perfectly separates the data. However, if the data is not linearly separable, the PLA will not converge, and may keep adjusting the weights indefinitely, or oscillate.

5 Types of Learning

5.1 Premise of Learning

At its core, learning, in a broad sense, is about using a set of observations to uncover an underlying process. This premise is not unique to machine learning; it is shared across various disciplines, including statistics. In statistics, for example, the underlying process is often a probability distribution, and observations are samples drawn from that distribution. The goal is to infer the distribution from the samples.

5.2 Categorizing Learning Types

Different types of learning emerge when this fundamental premise is applied in different contexts, often necessitating distinct mathematical tools and algorithms. Here are three major types of learning:

1. **Supervised Learning**
2. **Unsupervised Learning**
3. **Reinforcement Learning**

5.3 Supervised Learning

In **supervised learning**, the training data is "supervised" because for each input example, the correct output or target value is explicitly given. The algorithm learns to map inputs to outputs based on these labeled examples.

5.3.1 Example: Supervised Coin Recognition

Consider a vending machine that needs to recognize different types of coins. We can measure physical attributes of coins, such as size and mass. In supervised learning, we would have a dataset where each coin is described by its size and mass, and is also labeled with its denomination (e.g., 25 cents, 10 cents, 5 cents, 1 cent).

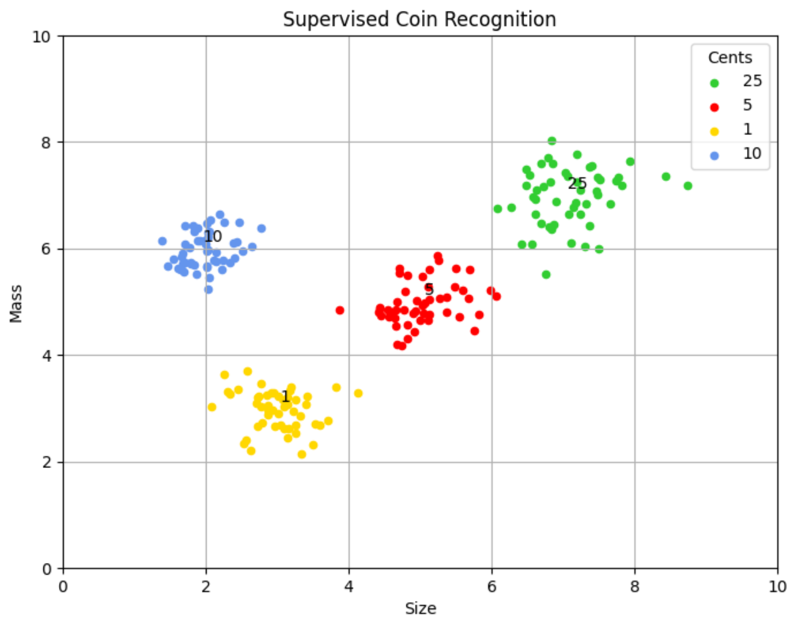


Figure 4: Supervised Coin Recognition Data

In the figure above, each cluster of points represents a type of coin, color-coded and labeled with its value. The supervised learning algorithm uses this labeled data to learn boundaries that can classify future, unlabeled coins into their correct denominations. Once trained, the system can recognize new coins based on their size and mass measurements by determining which region they fall into, delimited by learned boundaries.

5.4 Unsupervised Learning

In **unsupervised learning**, the training data is unlabeled. The algorithm only receives input data without corresponding correct outputs. The goal is to find patterns, structures, or groupings within the data itself, without explicit guidance.

5.4.1 Example: Unsupervised Coin Recognition

Imagine the same coin recognition problem, but now the vending machine only measures the size and mass of coins without knowing their denominations. The data is just a collection of measurements, without labels indicating coin values.

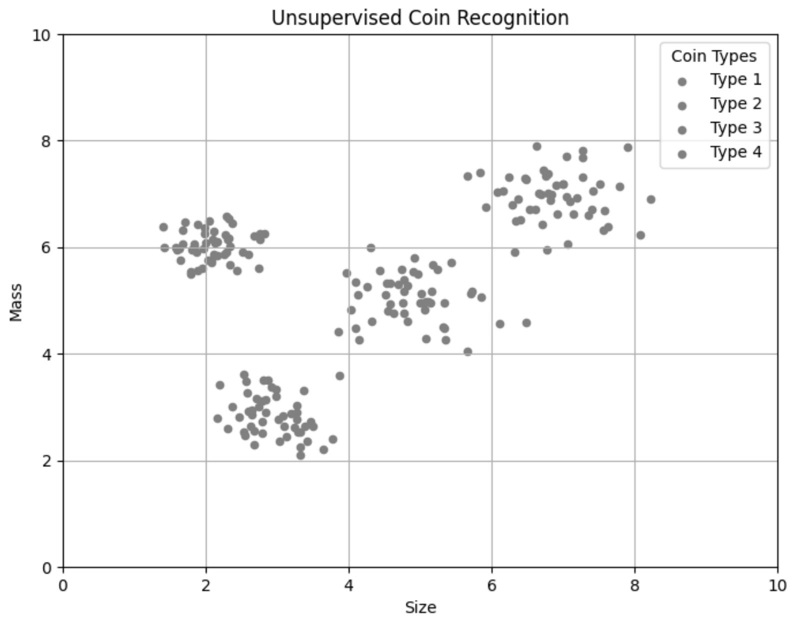


Figure 5: Unsupervised Coin Recognition Data

As shown in the figure, without labels, the data points still cluster into distinct groups. An unsupervised learning algorithm can identify these clusters, effectively categorizing the coins into different types based on their physical attributes. Although the algorithm doesn't know that "Type 1" is a quarter or "Type 4" is a dime, it successfully groups similar coins together. This form of learning is useful for exploratory data analysis, clustering, dimensionality reduction, and discovering hidden structures in data.

5.5 Reinforcement Learning

In **reinforcement learning**, the algorithm interacts with an environment to learn optimal behavior. It receives feedback in the form of rewards or penalties based on its actions. There are no explicit correct outputs provided for inputs; instead, the learning is driven by maximizing cumulative rewards.

5.5.1 Example: Toddler Learning about Hot Tea

Consider a toddler learning not to touch a hot cup of tea. The input is the sight of a tea cup. Initially curious, the toddler might reach out to touch it. The action of touching results in a negative reward (pain, "Ouch!"). This negative experience reinforces the association between touching a steaming cup and pain. Over time, through repeated experiences and feedback (rewards or penalties), the toddler learns to avoid touching hot cups. The learning is not from explicit instructions but from the consequences of actions in an environment. Reinforcement learning is about learning to make sequences of decisions to achieve a long-term goal, guided by feedback from the environment.

5.5.2 Application: Game Playing

Reinforcement learning is prominently used in training AI agents to play games. For example, a computer learning to play backgammon might make a move (action) based on the current game state (input). The outcome of the game (win or loss) serves as a reward signal. The algorithm learns to make moves that maximize its chances of winning over many games, without being explicitly told the optimal move in each situation.

6 A Learning Puzzle: The Limits of Learning

6.1 The Puzzle

Consider the following learning puzzle. We are given training examples in the form of 3x3 grids of black and white squares, each labeled with either -1 or +1. The task is to learn a function that predicts the label for a new, unlabeled 3x3 grid.

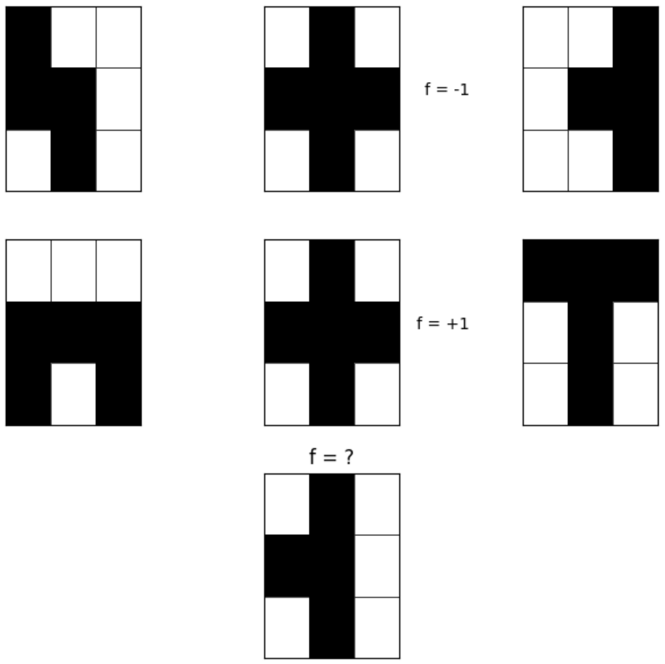


Figure 6: Learning Puzzle Patterns

The top row of grids are labeled with $f = -1$, and the middle row are labeled with $f = +1$. The bottom grid is the test point for which we want to predict the label, $f =?$.

6.2 The Ambiguity of Learning

Upon examining the puzzle, one might notice different potential patterns. For example, one could hypothesize that the function is +1 if the pattern is symmetric, or -1 if the top-left square is black. However, different individuals might perceive different patterns, leading to varying predictions for the test point (+1 or -1).

This divergence in answers highlights a fundamental issue: with a finite training set, there can be multiple functions that are consistent with the given data but differ on unseen points. Since the *true* target function is unknown, and we only have a limited set of examples, we cannot definitively determine the correct label for the test point based solely on the provided data. The function could indeed be anything.

6.3 Implications for Learning

The learning puzzle illustrates a critical concept: **learning from data is inherently ambiguous**. If the target function is truly unknown, and we only observe its values at a finite number of points, we cannot definitively extrapolate its behavior to unobserved points. There are infinitely many functions that could explain the given training data.

Does this mean learning is impossible? Not necessarily. While we cannot be certain about predictions on unseen data, machine learning is still possible and incredibly useful. The key is to understand under what conditions and to what extent we can reliably generalize from training data to unseen data. The next lecture will delve into the question of when and why learning is feasible, despite the inherent ambiguity illustrated by this puzzle.

7 Conclusion

This chapter introduced the fundamental problem of learning and the key components involved. We explored the essence of machine learning through the movie rating example, formalized the learning problem using the credit approval metaphor, and introduced the perceptron model as a simple learning algorithm. We also categorized different types of learning and concluded with a puzzle demonstrating the inherent ambiguity and challenges in learning from limited data. The critical questions raised about the feasibility and limits of learning will be addressed in the subsequent lectures, starting with the next lecture focusing on whether learning is indeed feasible.