UNIT I

Fundamentals of Deep Learning: Artificial Intelligence, History of Machine learning: Probabilistic Modeling, Early Neural Networks, Kernel Methods, Decision Trees, Random forests and Gradient Boosting Machines

Fundamentals of Machine Learning: Four Branches of Machine Learning, Evaluating Machine learning Models, Overfitting and Underfitting

....

1. Artificial Intelligence

- ➢ The topic of whether computers might be taught to "think" was first posed in the 1950s by a small group of pioneers in the developing discipline of computer science. The implications of this question are still being researched today.
- ➢ The endeavour to automate intellectual processes typically carried out by humans would serve as a succinct explanation of the area. As a result, AI is a broad area that comprises a variety of methods that include learning as well as machine learning and deep learning. For instance, early chess programmes used just hardcoded rules created by programmers and were not machine learning applications.
- ➢ For many years, experts thought that AI could be achieved by having programmers write a lot of rules for computers to follow.
- ➢ This approach is called symbolic AI, and it was the main way of doing AI from the 1950s to the late 1980s.
- ➢ Symbolic AI reached its peak popularity in the 1980s, when expert systems were very popular.

1.1. Artificial intelligence, machine learning, and deep learning

Artificial intelligence (AI), machine learning (ML), and deep learning (DL) are all terms that are often used interchangeably, but they actually have different meanings.

- **Artificial intelligence** is a broad term that refers to the ability of machines to perform tasks that are typically associated with human intelligence, such as learning, reasoning, and problem-solving.
- **Machine learning** is a subset of AI that involves the development of algorithms that can learn from data without being explicitly programmed. Machine learning algorithms are trained on large datasets, and they can then be used to make predictions or decisions about new data.
- **Deep learning** is a subset of machine learning that uses artificial neural networks to learn from data. Neural networks are inspired by the human brain, and they can be used to solve complex problems that would be difficult or impossible to solve with traditional machine learning algorithms.

In other words, AI is the umbrella term, ML is a subset of AI, and DL is a subset of ML.

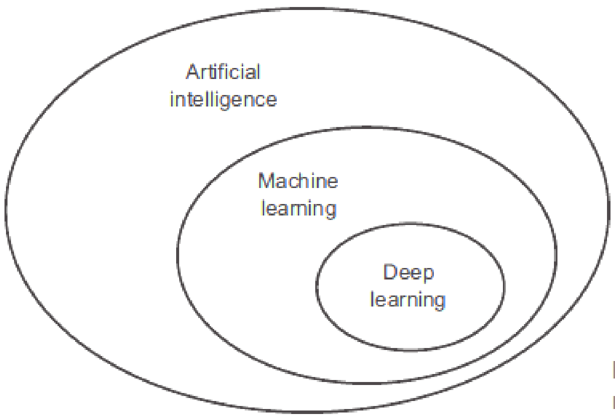


Figure 1.1 Artificial intelligence, machine learning, and deep learning

Here are some examples of how AI, ML, and DL are being used today:

- **AI** is being used to develop self-driving cars, facial recognition software, and spam filters.
- **ML** is being used to predict customer behaviour, optimize product recommendations, and personalize marketing campaigns.
- **DL** is being used to develop natural language processing (NLP) models, image recognition algorithms, and medical diagnosis tools.

AI, ML, and DL are all rapidly growing fields, and they are having a major impact on our lives. As these technologies continue to develop, we can expect to see even more innovative and groundbreaking applications in the years to come.

1.2 Machine Learning

Machine learning is a type of artificial intelligence (AI) that allows software applications to become more accurate in predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

In other words, machine learning algorithms can learn from data and improve their performance over time. This is in contrast to traditional programming, where software applications are explicitly programmed to perform specific tasks.

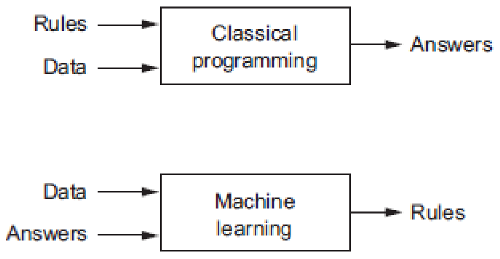


Figure 1.2 Machine learning: a new programming paradigm

Machine learning is used in a wide variety of applications, including:

- **Predictive analytics** - Machine learning can be used to predict future events, such as customer behavior, product demand, and fraud.
- **Personalization** - Machine learning can be used to personalize user experiences, such as product recommendations and advertising.
- **Fraud detection** - Machine learning can be used to detect fraudulent transactions, such as credit card fraud and insurance fraud.
- **Medical diagnosis** - Machine learning can be used to diagnose medical conditions, such as cancer and heart disease.

Here are some examples of how machine learning is used today:

- **Netflix** uses machine learning to recommend movies and TV shows to its users.
- **Amazon** uses machine learning to recommend products to its customers.
- **Google** uses machine learning to power its search engine, translate languages, and recognize objects in images.
- **Spotify** uses machine learning to create personalized playlists for its users.
- **Tesla** uses machine learning to power its self-driving cars.

1.3 The "deep" in deep learning

Deep learning is a type of machine learning that learns from data by creating successive layers of increasingly meaningful representations. The depth of a deep learning model refers to how many layers it has. Modern deep learning models often have tens or even hundreds of layers.

In contrast, other approaches to machine learning typically only learn one or two layers of representations. These approaches are sometimes called shallow learning.

The main difference between deep learning and shallow learning is that deep learning models can learn more complex representations of data. This makes them more powerful and able to solve more difficult problems.

Here is an analogy that might help you understand deep learning:

Imagine you are trying to understand a book. You could start by reading the first page, then the second page, and so on. This would be like shallow learning. You would only be able to understand the book at a superficial level.

However, you could also read the book by first reading the introduction, then the table of contents, then the chapters in order. This would be like deep learning. You would be able to understand the book at a much deeper level.

Deep learning is a powerful technique that is having a major impact on many different fields. It is used in applications such as image recognition, natural language processing, and machine translation.

Neural networks are a type of mathematical model that is inspired by the human brain. They are made up of layers of interconnected nodes, and they can learn to represent complex patterns in data.

In deep learning, neural networks are used to learn successive layers of increasingly meaningful representations. This is done by feeding the network a large amount of data, and then adjusting the weights of the connections between the nodes until the network is able to correctly classify or predict the data.

The term "neural network" is a reference to neurobiology, but neural networks are not models of the brain. The brain is a complex organ, and we do not yet fully understand how it works. Neural networks are much simpler models, and they are not designed to be a complete representation of the brain.

The idea that deep learning is "just like the brain" is a popular misconception. There are some similarities between neural networks and the brain, but there are also many important differences. For example, the brain uses a different learning mechanism than neural networks, and it is able to learn from much less data.

In the context of deep learning, the term "neural network" is simply a way of referring to a particular type of mathematical model. It is not meant to imply that the model is a complete representation of the brain.
Example:

The representations learned by a deep-learning algorithm are typically **abstract** and **meaningful**. They are abstract in the sense that they are not directly related to the original data. For example, the representations learned by a deep-learning algorithm for image recognition are not simply the pixels of the image. Instead, they are more abstract features, such as the edges, shapes, and textures of the image.

The representations learned by a deep-learning algorithm are also meaningful in the sense that they are related to the task that the algorithm is trying to perform. For example, the representations learned by a deep-learning algorithm for image recognition are related to the identity of the digit in the image.

Let's take a look at how a network several layers deep (see Figure 1.5) transform an image of a digit in order to recognize what digit it is.

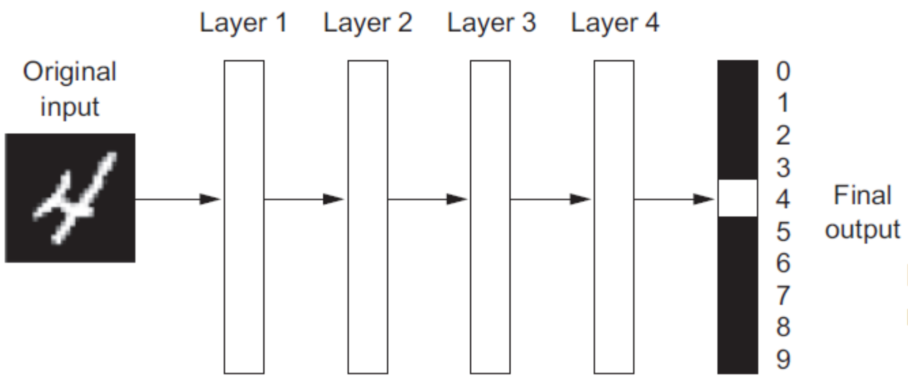


Figure 1.5 A deep neural network for digit classification

Figure 1.6 illustrates how the network alters the digit picture into representations that diverge ever more from the original and reveal even more about the outcome. A deep network can be compared to an information distillation process with multiple stages, where information is passed through progressively purer filters until it is suitable for a given purpose.

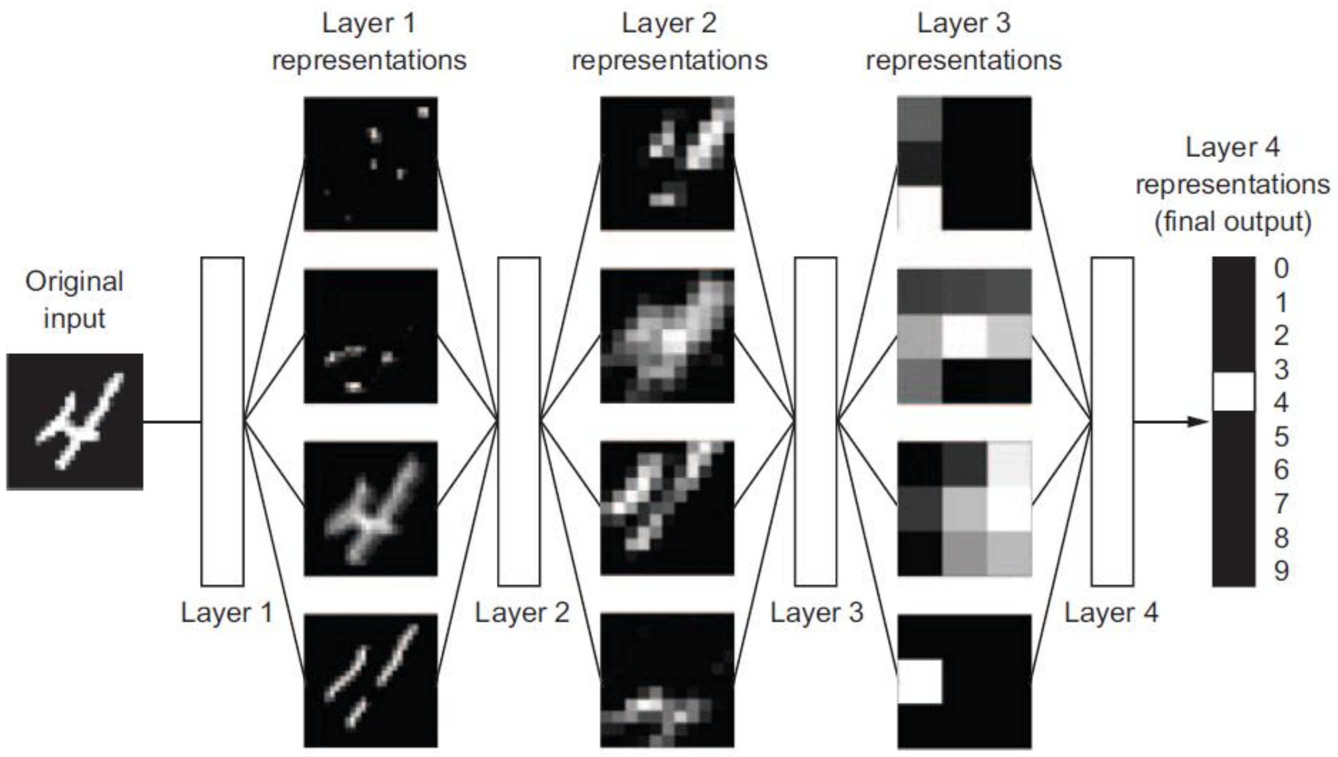


Figure 1.6 Deep representations learned by a digit-classification model

1.4 Understanding how deep learning works, in three figures

Machine learning is about finding a way to map inputs to targets. For example, we could map images of cats to the label "cat". We do this by observing many examples of inputs and targets, and then learning a sequence of simple data transformations that can be used to map new inputs to targets. These data transformations are learned by exposure to examples, which means that we show the machine learning algorithm many examples of inputs and targets, and it learns how to map the inputs to the targets.

Deep neural networks are a type of machine learning algorithm that can do this input-to-target mapping very well. They do this by using a deep sequence of simple data transformations, which allows them to learn very complex mappings.

Now let's look at how this learning happens, concretely.

- The weights of a layer in a deep neural network determine how the layer transforms its input data.
- The weights are essentially a bunch of numbers.
- Finding the correct values for all the weights is a daunting task.
- This is because modifying the value of one weight can affect the behavior of all the other weights in the network.
- The process of finding the correct values for the weights of a deep neural network is called training.
- Training involves feeding the network a large dataset of examples.
- The training process is typically done using an algorithm called backpropagation.

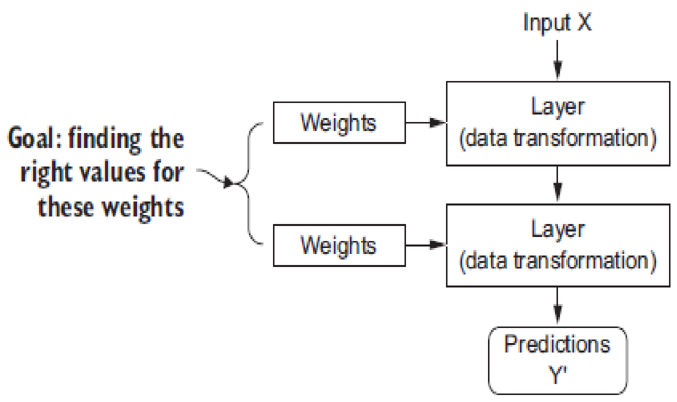

Input X
Goal: finding the right values for these weights
Weights
Layer (data transformation)
Weights
Layer (data transformation)
Predictions Y'

Figure 1.7 A neural network is parameterized by its weights.

To control something, first you need to be able to observe it. To control the output of a neural network, you need to be able to measure how far this output is from what you expected. This is the job of the *loss function* of the network, also called the *objective function*.

The loss function is a measure of how well the network has performed on a particular example. The loss function takes the predictions of the network and the true target, and then calculates a distance score. The distance score is a measure of how far the predictions are from the target.

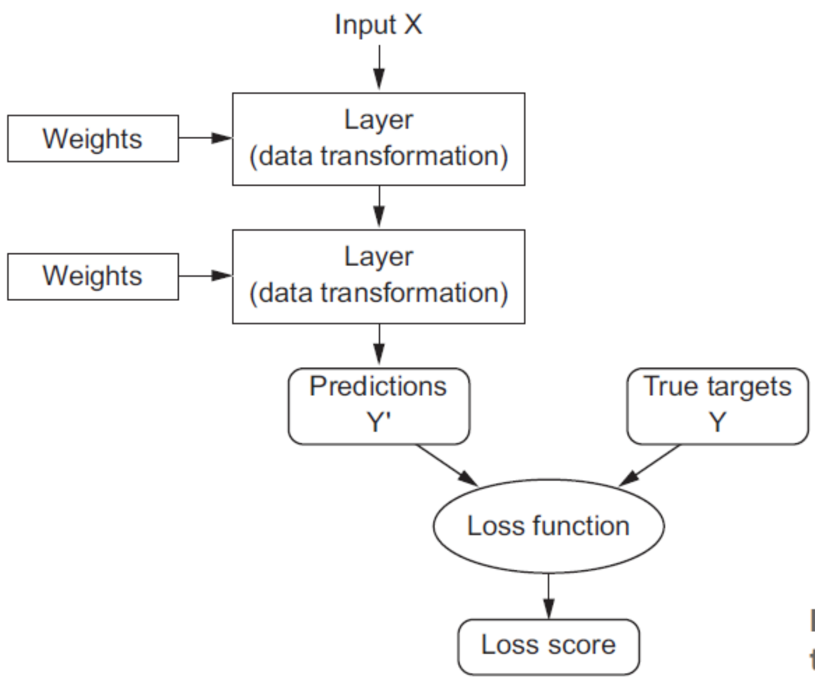


Figure 1.8 A loss function measures the quality of the network's output.

The loss function is used to train the neural network. The goal of training is to minimize the loss function. This means that the goal is to get the predictions of the network as close to the target as possible.

There are many different types of loss functions. Some common loss functions include:

- **Mean squared error (MSE)**: This is a loss function that measures the squared difference between the predictions and the target.
- **Cross-entropy**: This is a loss function that is used for classification problems. It measures the difference between the predicted probabilities and the true probabilities.
- **Huber loss**: This is a loss function that is less sensitive to outliers than MSE.

The choice of loss function depends on the type of problem that the neural network is being trained to solve.

The fundamental trick in deep learning is to use this score as a feedback signal to adjust the value of the weights a little, in a direction that will lower the loss score for the current example. This adjustment is the job of the *optimizer*, which implements what's called the *Backpropagation* algorithm: the central algorithm in deep learning.

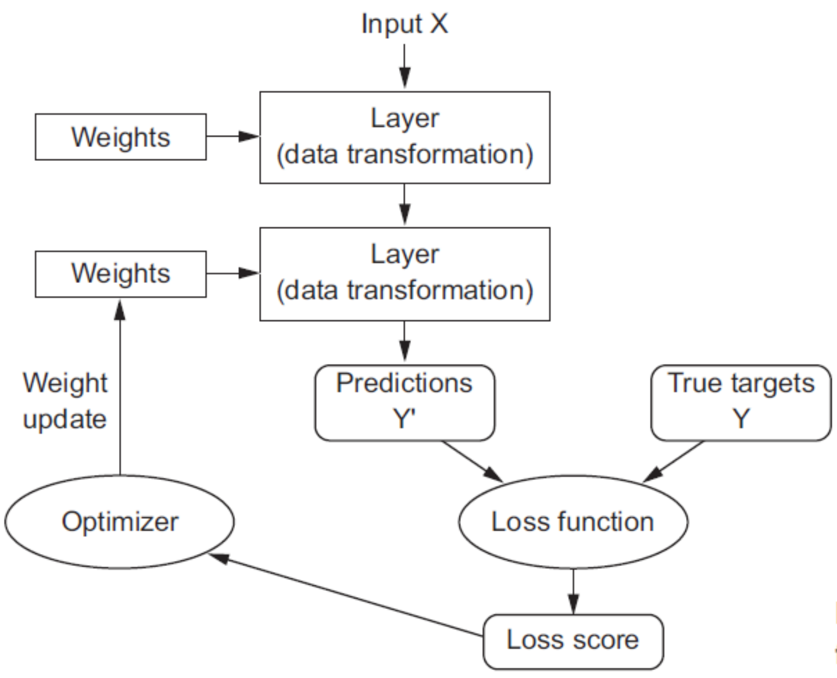


Figure 1.9 The loss score is used as a feedback signal to adjust the weights.

Initially, the weights of the network are assigned random values, so the network merely implements a series of random transformations.

Naturally, its output is far from what it should ideally be, and the loss score is accordingly very high. But with every example the network processes, the weights are adjusted a little in the correct direction, and the loss score decreases. This is the *training loop*, which, repeated a sufficient number of times (typically tens of iterations over thousands of examples), yields weight values that minimize the loss function. A network with a minimal loss is one for which the outputs are as close as they can be to the targets: a trained network.

1.5 *What deep learning has achieved so far*

In particular, deep learning has achieved the following breakthroughs, all in historically difficult areas of machine learning:

- Near-human-level image classification
- Near-human-level speech recognition
- Near-human-level handwriting transcription
- Improved machine translation
- Improved text-to-speech conversion
- Digital assistants such as Google Now and Amazon Alexa
- Near-human-level autonomous driving
- Improved ad targeting, as used by Google, Baidu, and Bing
- Improved search results on the web
- Ability to answer natural-language questions
- Superhuman Go playing

2. Before deep learning: a brief history of machine learning

Deep learning has become very popular and is getting a lot of attention and investment in the AI field. However, it's essential to know that it's not the only successful type of machine learning. Many other algorithms are used in the industry today, and they are not necessarily deep learning algorithms.

Deep learning might not always be the best choice for a particular task. Sometimes there might not be enough data for it to work well, or another algorithm could solve the problem more effectively. If you're new to machine learning and only know about deep learning, you might end up trying to use it for everything and see every problem as a "nail" for your "hammer."

To avoid this trap, it's crucial to learn about other approaches to machine learning and use them when they are more suitable for the task at hand.

2.1. Probabilistic modeling

Probabilistic modeling is a way to use statistics to understand data. It's been around for a long time, and it's still used a lot today. One of the most popular probabilistic models is called **Naive Bayes**.

Naive Bayes is a simple but powerful algorithm that can be used for a variety of tasks, including spam filtering, text classification, and medical diagnosis. It works by assuming that each feature in a dataset is independent of the others. This means that the probability of a certain outcome is only affected by the probability of that outcome for that feature.

Naive Bayes is a type of machine learning algorithm that uses Bayes' theorem to classify data. It assumes that the features in the data are independent of each other, which is why it's called Naive Bayes.

Bayes' theorem is a mathematical formula that can be used to calculate the probability of an event happening, given some prior knowledge. In the case of Naive Bayes, the prior knowledge is the distribution of features in the training data.

Naive Bayes is a simple algorithm, but it can be very effective for classification tasks. It's also fast and scalable, making it a good choice for large datasets.

A closely related model is the *logistic regression* (logreg for short), which is sometimes considered to be the "hello world" of modern machine learning.

2.2. Early neural networks

Early neural networks were inspired by the biological neural networks in the human brain. They were simple models that consisted of a few layers of interconnected neurons. These neurons were typically threshold gates, which means that they would only activate if their input was above a certain threshold.

The first early neural network was the perceptron, which was developed by Frank Rosenblatt in 1957. The perceptron was a simple model that could be used to classify binary data. It was a single-layer neural network with a threshold gate at the output.

Other early neural networks included the ADALINE and MADALINE networks, which were developed by Bernard Widrow and Marcian Hoff in 1959. These networks were also single-layer neural networks, but they used a different type of neuron called a linear unit.

Early neural networks were limited in their capabilities. They could only be used to solve simple problems, and they were not very robust to noise. However, they laid the foundation for the development of more powerful neural networks that are used today.

Here are some of the key limitations of early neural networks:

- They were only able to learn simple linear relationships.
- They were not very robust to noise.
- They were computationally expensive to train.

Despite these limitations, early neural networks were a significant breakthrough in the field of artificial intelligence. They showed that it was possible to build machine learning models that were inspired by the human brain. This led to the development of more powerful neural networks that are used today.

2.3. Kernel Methods

As neural networks started to gain some respect among researchers in the 1990s, thanks to this first success, a new approach to machine learning rose to fame and quickly sent neural nets back to oblivion: kernel methods

Kernel methods are a group of classification algorithms, the best known of which is the *support vector machine* (SVM).

SVMs aim at solving classification problems by finding good *decision boundaries* between two sets of points belonging to two different categories.

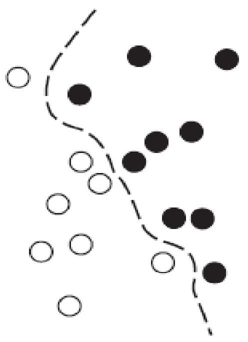


Figure 1.10
A decision boundary

A decision boundary can be thought of as a line or surface separating your training data into two spaces corresponding to two categories.

To classify new data points, you just need to check which side of the decision boundary they fall on.

SVMs proceed to find these boundaries in two steps:

1.The data is mapped to a new high-dimensional representation where the decision boundary can be expressed as a hyperplane (if the data was two dimensional, a hyperplane would be a straight line).

2. A good decision boundary (a separation hyperplane) is computed by trying to maximize the distance between the hyperplane and the closest data points from each class, a step called *maximizing the margin*. This allows the boundary to generalize well to new samples outside of the training dataset.

The technique of mapping data to a high-dimensional representation where a classification problem becomes simpler may look good on paper, but in practice it's often computationally intractable. That's where the *kernel trick* comes in (the key idea that kernel methods are named after).

The kernel trick is a technique used in machine learning that allows kernel methods to operate in a high-dimensional space without explicitly computing the coordinates of the data in that space. This can be computationally advantageous, as it can avoid the need to perform matrix multiplications in high-dimensional space.

2.4. Decision trees, random forests, and gradient boosting machines

Decision trees are flowchart-like structures that let you classify input data points or predict output values given inputs . They're easy to visualize and interpret.

Decisions trees learned from data began to receive significant research interest in the 2000s, and by 2010 they were often preferred to kernel methods.

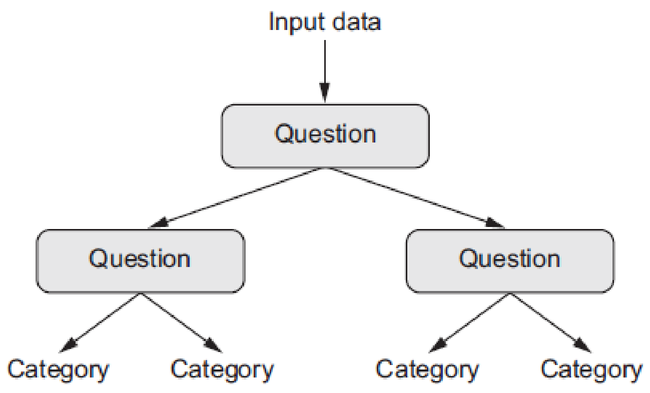


Figure 1.11 A decision tree: the parameters that are learned are the questions about the data. A question could be, for instance, "Is coefficient 2 in the data greater than 3.5?"

In particular, the *Random Forest* algorithm introduced a robust, practical take on decision-tree learning that involves building a large number of specialized decision trees and then ensembling their outputs. Random forests are applicable to a wide range of problems—you could say that they're almost always the second-best algorithm for any shallow machine-learning task.

A gradient boosting machine, much like a random forest, is a machine-learning technique based on ensembling weak prediction models, generally decision trees. It uses *gradient boosting*, a way to improve any machine-learning model by iteratively training new models that specialize in addressing the weak points of the previous models.

Applied to decision trees, the use of the gradient boosting technique results in models that strictly outperform random forests most of the time, while having similar properties. It may be one of the best, if not *the* best, algorithm for dealing with non-perceptual data today.

Chapter-II

Fundamentals of Machine Learning: Four Branches of Machine Learning, Evaluating Machine learning Models, Overfitting and Underfitting

………

1. Four branches of machine learning

Machine learning is a subfield of artificial intelligence that focuses on developing algorithms and statistical models that enable computers to learn and improve their performance on a specific task through experience. There are four main branches of machine learning:

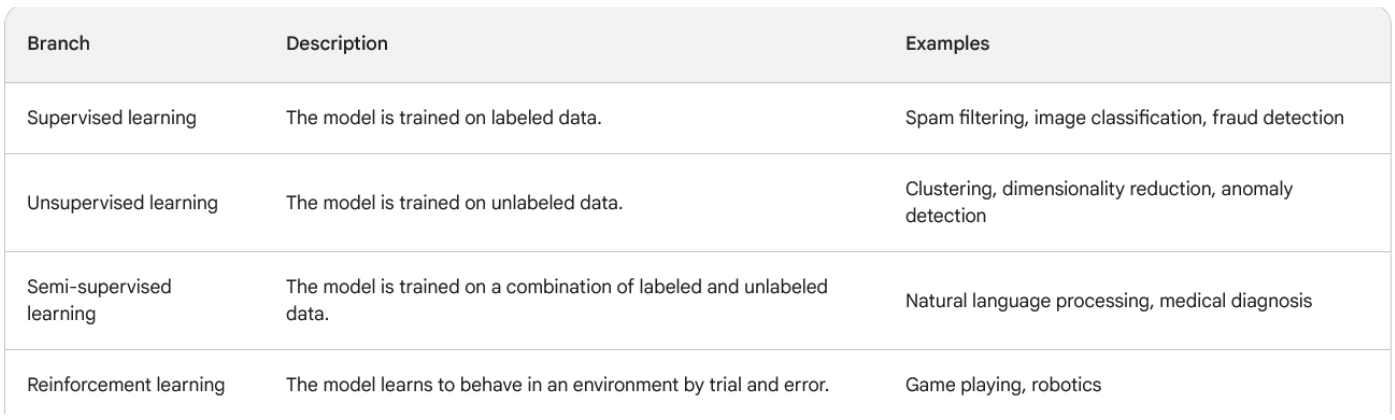
1. **Supervised Learning**: In supervised learning, the algorithm is trained on a labeled dataset, where each input data point is associated with the corresponding target or output label. The goal of the algorithm is to learn a mapping from inputs to outputs, enabling it to make predictions on new, unseen data. Common tasks in supervised learning include classification (assigning labels to input data) and regression (predicting numerical values).

2. **Unsupervised Learning**: Unsupervised learning involves training algorithms on an unlabeled dataset, where the data does not have predefined output labels. The goal of unsupervised learning is to discover patterns, structures, or relationships within the data. Clustering, where the algorithm groups similar data points together, and dimensionality reduction, which aims to simplify data while preserving essential characteristics, are examples of unsupervised learning tasks.

3. **Semi-Supervised Learning**: Semi-supervised learning is a combination of supervised and unsupervised learning. The algorithm is trained on a dataset that contains both labeled and unlabeled data. The labeled data provides some information for guidance, and the unlabeled data helps the algorithm learn more about the underlying structure of the data, often leading to better performance when labeled data is limited or expensive to obtain.

4. **Reinforcement Learning**: Reinforcement learning is different from the previous three types as it involves an agent that interacts with an environment to achieve a goal. The agent takes actions in the environment and receives feedback in the form of rewards or penalties. The goal of the agent is to learn a policy or strategy that maximizes the cumulative reward over time. Reinforcement learning is commonly used in applications such as game playing, robotics, and autonomous systems.

These four branches cover a broad range of machine learning techniques and applications, and they continue to evolve and advance as researchers and practitioners explore new algorithms and approaches.



Branch	Description	Examples
Supervised learning	The model is trained on labeled data.	Spam filtering, image classification, fraud detection
Unsupervised learning	The model is trained on unlabeled data.	Clustering, dimensionality reduction, anomaly detection
Semi-supervised learning	The model is trained on a combination of labeled and unlabeled data.	Natural language processing, medical diagnosis
Reinforcement learning	The model learns to behave in an environment by trial and error.	Game playing, robotics

2. Evaluating Machine learning Models

Evaluating machine learning models is an important step in the machine learning process. It allows you to assess the performance of your model and identify any areas where it can be improved. There are a number of different metrics that can be used to evaluate machine learning models, each with its own strengths and weaknesses.

2.1. Training, validation, and test sets

Evaluating a model always boils down to splitting the available data into three sets: training, validation, and test. You train on the training data and evaluate your model on the validation data. Once your model is ready for prime time, you test it one final time on the test data.

You may ask, why not have two sets: a training set and a test set? You would train on the training data and evaluate on the test data. Much simpler!

- **Overfitting.** If we only train on the training set, and then evaluate on the test set, there is a risk that the model will overfit the training data. This means that the model will learn the specific details of the training data, and will not be able to generalize to new data.

- **Bias.** If we only use the training set to evaluate the model, we may not be able to identify any biases in the model. This is because the training set is the only data that the model has seen, so it is possible that the model will learn the biases that are present in the training data.

- **Unreliability.** If we only use the test set to evaluate the model, then the results of the evaluation may not be reliable. This is because the test set is only a small sample of the data that the model will be used on in the real world.

To address these issues, we often split the data into three sets: a training set, a validation set, and a test set. The training set is used to train the model, the validation set is used to tune the hyperparameters of the model, and the test set is used to evaluate the final model.

This approach helps to prevent overfitting, identify biases, and ensure that the results of the evaluation are reliable.

In addition to the three sets mentioned above, some people also use a fourth set called the **holdout set**. The holdout set is a separate set of data that is not used for training, validation, or testing. **The holdout set is used to get an independent estimate of the model's performance.**

The use of multiple sets can be a bit more complex, but it is generally considered to be a best practice for building reliable machine learning models.

Splitting your data into training, validation, and test sets may seem straightforward, but there are a few advanced ways to do it that can come in handy when little data is available.

Let us review three classic evaluation recipes: simple hold-out validation, K-fold validation, and iterated K-fold validation with shuffling

1. SIMPLE HOLD-OUT VALIDATION

Simple hold-out validation is a method of evaluating a machine learning model by splitting the data into two sets: a training set and a test set. The training set is used to train the model, and the test set is used to evaluate the model's performance.

The following steps are involved in simple hold-out validation:

1. Split the data into a training set and a test set. The training set should typically be about 80% of the data, and the test set should be about 20% of the data.

2. Train the model on the training set.

3. Evaluate the model on the test set.
4. Repeat steps 2 and 3 for different hyperparameters or different models.

The model with the best performance on the test set is the best model.

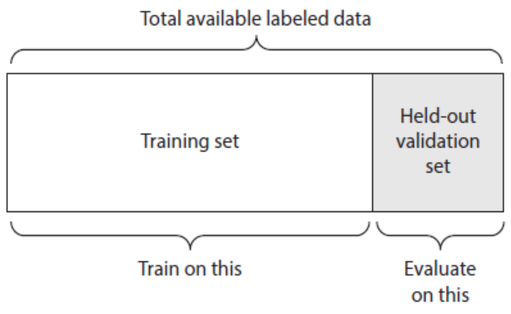


Figure 4.1 Simple hold-out validation split

Here are some of the advantages of simple hold-out validation:

- It is simple to understand and implement.
- It is a good way to evaluate the performance of a model.

Here are some of the disadvantages of simple hold-out validation:

- It can be sensitive to the way that the data is split into training and test sets.
- It can be difficult to get a reliable estimate of the model's performance if the test set is too small.

simple hold-out validation is a good way to evaluate the performance of a machine learning model. However, it is important to be aware of its limitations.

2. K-FOLD VALIDATION

K-fold validation is a method of evaluating a machine learning model by splitting the data into k folds. The model is trained k times, each time using a different fold as the validation set and the remaining folds as the training set. The performance metrics obtained from each fold are averaged to provide a more robust estimate of the model's generalization performance.

The following steps are involved in k-fold validation:

1. Split the data into k folds.
2. For each fold:
   - Train the model on the remaining k-1 folds.
   - Evaluate the model on the current fold.
3. Average the performance metrics from each fold.

The model with the best average performance is the best model.

Here are some of the advantages of k-fold validation:

- It is more robust to overfitting than simple hold-out validation.
- It can provide a more reliable estimate of the model's performance.

Here are some of the disadvantages of k-fold validation:

- It can be more computationally expensive than simple hold-out validation.
- It can be difficult to implement if the data is not evenly distributed.

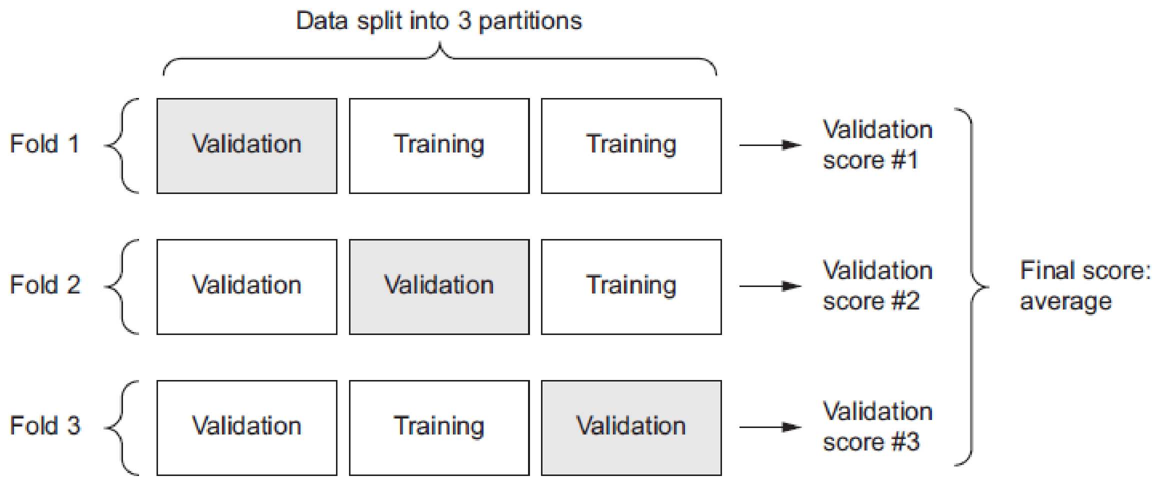


Figure 4.2 Three-fold validation

k-fold validation is a more sophisticated method of evaluating a machine learning model than simple hold-out validation. It can provide a more reliable estimate of the model's performance, but it can also be more computationally expensive.

2.2. *Things to keep in mind*

Keep an eye out for the following when you are choosing an evaluation protocol

- **The purpose of the evaluation.** What do you hope to achieve by evaluating your program or intervention? Are you looking to measure its effectiveness, efficiency, or impact? Once you know the purpose of the evaluation, you can choose a protocol that is designed to answer your specific questions.
- **The scope of the evaluation.** What aspects of your program or intervention will you be evaluating? Will you be looking at the overall impact, or will you be focusing on specific outcomes? The scope of the evaluation will determine the type of data you need to collect and the methods you will use to collect it.
- **The resources available.** How much time, money, and personnel do you have to dedicate to the evaluation? The resources available will affect the type of protocol you can choose. For example, if you have limited resources, you may need to choose a protocol that uses less data or that is less time-consuming to implement.
- **The ethical considerations.** What ethical issues are relevant to your evaluation? For example, if you are collecting personal data about participants, you will need to make sure that you have their consent and that you are protecting their privacy.

Once you have considered these factors, you can start to look for evaluation protocols that are a good fit for your needs. There are many different evaluation protocols available, so you should be able to find one that meets your specific requirements.

3. Overfitting and underfitting

Overfitting and underfitting are two common problems that can occur in machine learning. Overfitting occurs when a model learns the training data too well and starts to memorize the noise and outliers in the data. This can lead to the model performing poorly on new data that it has not seen before. Underfitting occurs when a model does not learn the training data well enough and is unable to make accurate predictions.

Here are some examples of overfitting and underfitting:

- **Overfitting:** A model that is trained to predict whether a patient has cancer might learn to memorize the specific features of the training data that are associated with cancer. This would allow the model to make accurate predictions on the training data, but it would also cause the model to perform poorly on new data that does not have the same features.
- **Underfitting:** A model that is trained to predict the price of a house might not learn the relationship between the features of the house and its price. This would cause the model to make inaccurate predictions on both the training data and new data

Optimization and generalization are two important concepts in machine learning. Optimization refers to the process of finding the best parameters for a model, while generalization refers to the ability of a model to make accurate predictions on new data.

The goal of machine learning is to build models that can generalize well to new data. However, there is a tension between optimization and generalization. If we optimize a model too much, it may become too complex and start to overfit the training data. This means that the model will make accurate predictions on the training data, but it will not generalize well to new data.

On the other hand, if we do not optimize the model enough, it may not learn the training data well enough and will not be able to make accurate predictions on new data.

The challenge in machine learning is to find a balance between optimization and generalization. This can be done by using regularization techniques, which add a penalty to the model's complexity. **Regularization** can help to prevent the model from overfitting the training data and improve its ability to generalize to new data.

Let us review some of the most common regularization techniques

3.1. Reducing the network's size

Reducing the size of a network can help to prevent overfitting by making the model less complex. A less complex model is less likely to memorize the noise and outliers in the training data. As a result, the model will be more likely to generalize well to new data.

3.2. Adding weight regularization

- ☞ Weight regularization is a technique used to prevent neural networks from overfitting the training data.
- ☞ There are two main types of weight regularization: L1 and L2.
- ☞ To add weight regularization to a neural network, you can use the kernel_regularizer argument when creating the layer.
- ☞ The amount of weight regularization to use is a hyperparameter that you will need to tune.

L1 regularization adds a penalty to the loss function that is proportional to the absolute value of the weights. This can help to reduce the number of non-zero weights in the model, which can be useful for feature selection.

L2 regularization adds a penalty to the loss function that is proportional to the square of the weights. This is the most common type of weight regularization and it is generally effective at preventing overfitting.

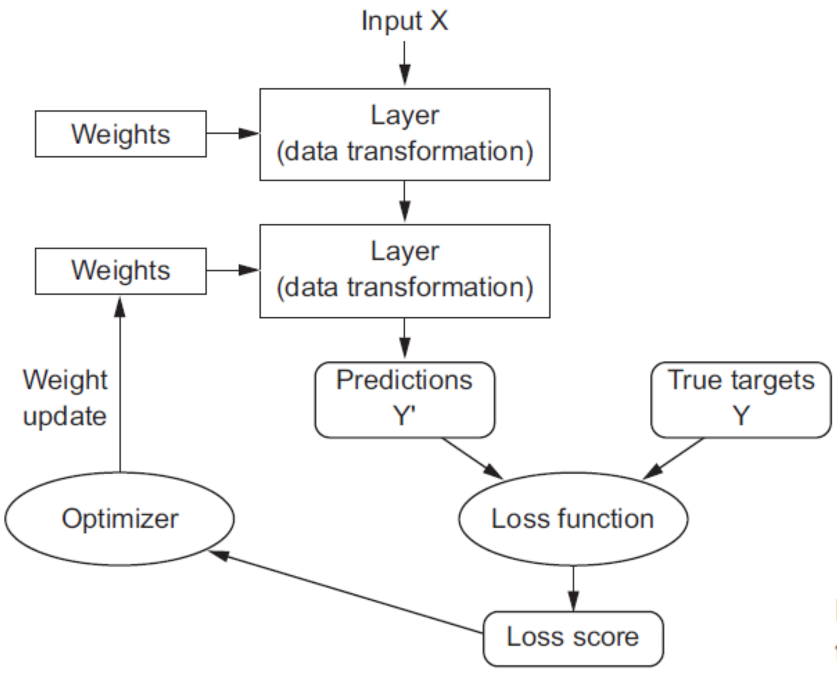


Figure 1.9 The loss score is used as a feedback signal to adjust the weights.

3.3.Adding dropout

Dropout is a regularization technique that can also be used to prevent neural networks from overfitting.

To add dropout to a neural network, you can use the Dropout layer.

For example, the following code adds dropout with a rate of 0.2 to the hidden layer of a neural network:

```
model.add(Dropout(0.2))
```

The rate argument specifies the probability of a neuron being dropped out.

- Dropout is typically used on hidden layers, but it can also be used on the input layer.
- You can use dropout on all hidden layers, or you can only use it on some of the layers.
- You can experiment with different dropout rates to see what works best for your model.
- Dropout is typically used during training, but you can also use it during inference if you want to make your model more robust to noise.